DBCSec: DBC File-Guided Secure Communication Mechanism for CAN-FD Bus

Yun Shen, Jie Cui, *Senior Member, IEEE*, Hong Zhong, Jing Zhang, Qingyang Zhang, Lu Wei, *Member, IEEE*, and Debiao He, *Member, IEEE*

***Abstract*—The network architecture of modern vehicles is composed of multiple communication protocols and electronic control units (ECU). Compared to the widely used protocol of Controller Area Network (CAN), CAN with Flexible Data-Rate (CAN-FD) protocol is suitable for applications requiring higher data throughput. However, the CAN-FD bus is vulnerable to intrusion by external attackers. Nowadays, several secure mechanisms have been proposed to protect the security of in-vehicle data. However, there are still two issues: 1) Most schemes use a centralized controller for key distribution, which can easily lead to a single point of failure; 2) The existing key management modes are not suitable for real-world CAN-FD networks in vehicle manufacturing. To address these issues, we propose a lightweight semi-decentralized scheme based on Database CAN (DBC) files to secure in-vehicle communication. ECUs are grouped on the send-receive relationships set in the DBC file, considering both the communication mode and sending efficiency. Furthermore, the proposed scheme overcomes reliance on long-term keys. Moreover, the security is analyzed by the random oracle model. The performance analysis is evaluated on microcontroller units (MCU) STM32H743IIT and Raspberry Pi 3B. The proposed scheme optimizes the computational costs of authentication, key agreement, and secure communication stages by up to 97.89%, 99.95%, and 82.35%, and optimizes the communication costs by up to 75.52%, 98.42%, and 29.41% compared to existing methods. Simulation experiments demonstrate that the bus load of the scheme increases by up to 9.84% compared to the baseline network.**

***Index Terms*—In-vehicle network, electronic control unit, controller area network, group key management.**

I. Introduction

With the rapid development of the Internet of Vehicles (IoV) and Vehicle-to-Everything (V2X) communication [1], [2], [3], [4], and artificial intelligence technology [5], [6], [7], vehicles are gradually evolving into mobile computing platforms, providing passengers with richer services, thereby significantly improving the passenger's travel experience. The Electronic Control Units (ECU) are core components of modern automobiles, responsible for managing and controlling various electronic systems. Inside vehicles, dozens to hundreds of ECUs are connected to each other through complex network systems to achieve collaborative work of various systems and functions. The in-vehicle electronic and electrical architecture (EEA) [8], [9], [10] built through multiple ECUs and communication protocols realizes the basic functions of the vehicle and is an indispensable component of modern automobiles. For the sake of improving the performance and reliability of the system, the EEA of modern vehicles has also evolved from distributed to highly integrated and networked. The concept of domain controller was introduced [11], which integrates the functions of multiple ECUs into a powerful controller to simplify network architecture. A typical domain-based EEA is shown in Fig. 1.

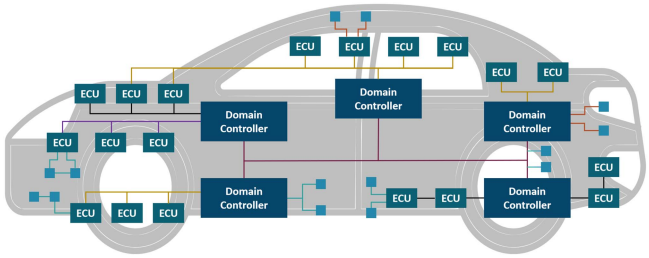


Fig. 1. Domain-based architecture for in-vehicle network.

For in-vehicle networks, Controller Area Network (CAN) [12], [13], [14], [15], [16] and the CAN with Flexible Data Rate (CAN-FD) [17], [18], [19], [20] are the most common in-vehicle communication protocols to address efficient communication issues between ECUs within vehicles. CAN-FD is an upgrade of CAN to meet the needs of modern vehicles for higher baud rates and larger data frames. CAN and CAN-FD buses are multi host broadcast communication mechanisms. All nodes on the network can send and receive messages without any encryption or authentication mechanisms. Unfortunately, in-vehicle CAN and CAN-FD buses are vulnerable to external attacks due to broadcast mode and lack of encryption and authentication mechanisms [21], [22], [23], [24], [25]. Woo et al. [26] developed a malicious smartphone applications, based on the CAN bus data acquisition, analysis, and injection, resulting in distortion of the dashboard, engine stop, and handle

Received 10 September 2024; revised 1 July 2025; accepted 28 July 2025. Date of publication 31 July 2025; date of current version 4 November 2025. The work was supported in part by the National Natural Science Foundation of China under Grant U23A20308, Grant 62472003, Grant 62202008, Grant 62302008, and Grant 62325209, in part by the Fundamental Research Funds for the Central Universities under Grant 2042023KF0203, in part by the Natural Science Foundation of Anhui Province, China under Grant 2408085JX010, and in part by the University Synergy Innovation Program of Anhui Province under Grant GXXT-2022-049. *(Corresponding author: Jie Cui.)*

Yun Shen, Jie Cui, Hong Zhong, Jing Zhang, Qingyang Zhang, and Lu Wei are with the Key Laboratory of Intelligent Computing and Signal Processing of Ministry of Education, School of Computer Science and Technology, Anhui Engineering Laboratory of IoT Security Technologies, Anhui University, Hefei 230039, China (e-mail: cuijie@mail.ustc.edu.cn).

Debiao He is with the School of Cyber Science and Engineering, Wuhan University, Wuhan 430072, China (e-mail: hedebiao@163.com).

Digital Object Identifier 10.1109/TDSC.2025.3594338

control. Koscher et al. [27] implemented attacks such as honking the horn, disable the window relays, and disable the windshield wipers by reverse-engineering and obscuring data packets on the CAN bus. Authentication [28] and encryption mechanisms have been widely used in various fields as means of resisting such attacks. As a result, the design of security mechanism for CAN and CAN-FD buses can effectively resist typical attacks.

However, existing secure communication solutions [29], [30], [31], [32], [33] for in-vehicle networks still have unresolved issues. The first unresolved issue is that most existing frameworks rely on centralized trusted ECUs (such as the gateway ECU) for key distribution. However, during vehicle runtime, once a single point of failure occurs in the gateway ECU [34], the security mechanism will be paralyzed. The second challenge that needs to be addressed is that existing research has not taken into account that in actual vehicle manufacturing, the bus guided by the database CAN (DBC) file is not in broadcast communication mode, but actually in multicast mode [35]. However, the existing key management modes cannot simultaneously balance the multicast communication mode and low latency of the CAN/CAN-FD bus.

A semi-decentralized scheme can alleviate the first issue. The semi-decentralized architecture means achieving real-time key agreement without domain controller participation. During the key agreement stage, the domain controller only needs to participate in the non-real-time phase and does not require real-time involvement. This design can mitigate risks of system paralysis caused by single-point failures to some extent, while also reducing computational overhead during real-time operations. For the second challenge, it is crucial to group ECUs by send-receive relationships in the DBC file and negotiate group keys. As a result, message encryption, authentication, and sending without changing the bus multicast mechanism is realized.

Therefore, designing a semi-decentralized group key management scheme can solve the above problems. However, as far as we know, all existing security schemes for in-vehicle networks rely on original equipment manufacturer (OEM) built-in long-term keys to negotiate the session keys. However, due to the diversity of OEM sources for ECUs, it is difficult to get all OEMs to have the preassemble required long-term keys. Long-term key management is also difficult. Therefore, how to design a security mechanism overcoming the reliance on long-term keys remains a major challenge. Moreover, in-vehicle CAN/CAN-FD networks are characterized by high real-time and low latency, how to design a security mechanism in such a context is also a problem to be solved.

To address the aforementioned issues, this paper proposes a novel semi-decentralized domain controller architecture to enhance the security of the in-vehicle CAN-FD bus. Important contributions of this paper are listed as follows:

1) First, to enhance the security of the in-vehicle CAN-FD network, a novel secure communication scheme oriented to the domain controller EEA is proposed. The proposed scheme groups ECUs on the send-receive relationship in the DBC file, which is adapted to the multicast mode of the CAN-FD network and meets the low latency requirement. The lightweight authenticated encryption with associated data (AEAD) algorithm TinyJAMBU with a short tag is used in the secure communication phase, reducing the computational and communication overhead of the ECU.
2) Second, the proposed scheme is semi-decentralized. We divide the key negotiation phase into centralized non-real-time secret share distribution phase and decentralized real-time key agreement phase. It reduces the computational overhead of real-time operation stage and the risk of single point of failure. By constructing three-layer polynomials, the contributory group key generation is realized.
3) Third, considering that it is difficult to preassemble long-term keys during vehicle manufacturing due to various OEMs of vehicle ECUs, the proposed scheme overcomes the dependence on long-term keys. Through Chebyshev polynomials, the scheme improves the applicability of security schemes in real-world scenarios.

The rest of this paper is organized as follows. Section II introduces related works. Section III presents the CAN-FD bus and the cryptography-related knowledge. The system model, threat model, and objectives are demonstrated in Section IV. Section V describes the proposed scheme. Section VI provides security analysis and security and functionality comparisons. Section VII describes the experimental platform as well as the computational overhead, communication overhead, and bus load. Finally, Section VIII summarizes the work.

II. Related Work

In this section, we will introduce the relevant researches of CAN/CAN-FD security mechanisms and DBC files.

A. CAN/CAN-FD Security Mechanisms

Previous CAN and CAN-FD secure communication schemes can be mainly categorized into security schemes based on asymmetric cryptography and symmetric cryptography. Asymmetric cryptography has been applied to in-vehicle network architectures because it can provide more functionalities [30], [31], [32]. However, due to its high computational and communication overhead, most schemes use symmetric cryptography algorithms for in-vehicle network security. Symmetric cryptography-based schemes can be further categorized into three types based on the key management modes, i.e., pairwise, group key, and global key. The global key-based security framework enables all ECUs within a subnet to share a global key and update the global key after each session [29], [33]. The pairwise security frameworks provide security protection for each pair of communicating ECUs [36], [37], [38]. Secure communication frameworks based on the group key mode tend to target key management between a group of ECUs/messages [39], [40], [41], [42], [43], [44], [45]. The summary and analysis of relevant literature are shown in Table I.

Due to the stronger functionality of asymmetric cryptography, Yu et al. [31] proposed a security protocol for in-vehicle data based on attribute-based encryption (ABE). To forward and process cross-domain messages, Cui et al. [32] introduced an in-vehicle proxy re-encryption (PRE) scheme. Carvajal-Roca

TABLE I
SUMMARY OF RECENT RELATED WORKS

Scheme	Key mode	Architecture	Preset long-term keys	Methodology	Advantage
[31] 2022	Public key	N/A	Yes	Attribute-based encryption	Support fine-grained access control
[32] 2024	Public key	N/A	Yes	Proxy re-encryption	Ensure cross-domain forwarding of confidential messages
[30] 2021	Public key	Semi-centralized	Yes	IBE, IBBE	Provide decentralized and dynamic key generation
[33] 2023	Global key	N/A	Yes	AES, HMAC	Provide data authentication and encryption
[29] 2020	Global key	Centralized	Yes	AES, MAC, ECDH	Enable session key security
[37] 2023	Pairwise key	Centralized	No	MAC, Grain stream cipher, Blom key management	Achieve lightweight encryption and authentication
[38] 2021	Pairwise key	N/A	N/A	MAC	Enhance security of real-time CAN-FD dataset
[36] 2023	Pairwise key	Centralized	N/A	AES, HMAC	Provide trinity-enabled security
[40] 2022	Group key	Centralized	Yes	HMAC, symmetric encryption	Authenticate without protocol modifications or traffic overhead
[45] 2019	Group key	Centralized	No	Physical unclonable function, implicit certificate	Establish trust roots using physical properties of SRAM
[41] 2024	Group key	Centralized	Yes	Chinese Remainder Theorem, AES, MAC	Dynamically group and manage ECUs based on credit value
[39] 2022	Group key	Centralized	Yes	AES, CMAC, ECC	Support intelligent throughput management and hardware signature mechanism
[42] 2021	Group key	Centralized	N/A	Diffie-Hellman protocol	Solve the problem of key exchange
[43] 2024	Group & pairwise key	Centralized	N/A	AES	Implement secure and low-latency CAN-FD
[44] 2023	Group key	Semi-centralized	No	ECQV, GDH	Decentralized negotiation without preset secrets
Ours	Group key	Semi-decentralized	Yes	Chebyshev chaotic mapping, TinyJAMBU	Design security mechanisms based on vehicle DBC files

et al. [30] presented a semi-centralized key management framework for the CAN-FD bus with Identity-Based Encryption (IBE) and Identity-Based Broadcast Encryption (IBBE). This key management mode leads to higher computational overheads, making it hard to apply to in-vehicle messages.

Based on the global key model, de Andrade et al. [33] proposed a cryptographic and authentication protocol for CAN-FD, which divides the frame data field into two blocks. Palaniswamy et al. [29] provided a secure in-vehicle communication protocol suite for key management and secure message flow of remote frames. The global key mode has a lower number of keys and a higher security, but increases the computational overhead per session and reduces the efficiency of message delivery.

For the pairwise key mode, Cui et al. [37] proposed a lightweight cryptographic authentication scheme for autonomous vehicles based on message authentication code (MAC) and Grain stream cipher. Lu et al. [36] introduced a low-overhead security management mechanism to ensure the confidentiality and integrity of CAN-FD messages. Xie et al. [38] proposed a security enhancement technique by maximizing the total MAC size of the independent message set. However, the above schemes suffer from the disadvantages of a large number of keys and high overhead of key distribution and updating. Moreover, pairwise key mode schemes change the broadcast (actually multicast) communication mechanism of the bus, which reduces the efficiency of message transmission.

In group-based protocols, it can be further classified into message-oriented and ECU-oriented group key management modes. For the message-oriented mode, Ying et al. [40] provided secure authentication for ECUs by utilizing covert channels without introducing extra traffic overhead. Püllen et al. [45] proposed to use implicit certificates to derive authenticated message-based group keys for ECUs. However, distributing keys for each CAN/CAN-FD message ID is impractical because the number of IDs can theoretically reach a maximum of 2^{29} in case of CAN/CAN-FD extended frames.

For the ECU-oriented group key management mode, Shen et al. [41] designed a two-layer ECU group management scheme based on the Chinese remainder theorem (CRT) to dynamically group and manage ECUs based on trustworthiness. Oberti et al. [39] proposed a new low-cost CAN-FD architecture to achieve ECU security communication in the field of vehicle regulations. Musuroi et al. [42] utilized the elliptic curve version of the Diffie-Hellman protocol to design a key exchange for the CAN and CAN-FD protocols. They solve the problem of secure exchange of cryptographic keys between ECUs. Lu et al. [43] presented a lightweight authentication protocol for securing automotive networks to seek a critical balance between high security and low latency. Sun et al. [44] innovatively adopted the Elliptic Curve Qu-Vanstone (ECQV) implicit certificates and group Diffie-Hellman (GDH) protocol to generate group keys. The above schemes realize encrypted communication between a group of ECUs, but they do not take into account the multicast mechanism of CAN and CAN-FD buses. When the sender and multiple receivers are not in a group, the sender must encrypt, authenticate, and send the message repeatedly. This also greatly reduces the efficiency of sending messages on the bus.

B. *Researches With DBC Files*

In existing works, researchers decode confidential DBC files using different methods to achieve reverse engineering of CAN/CAN-FD frames and obtain the signal information allocated in the frame payload. Choi et al. [46] proposed an

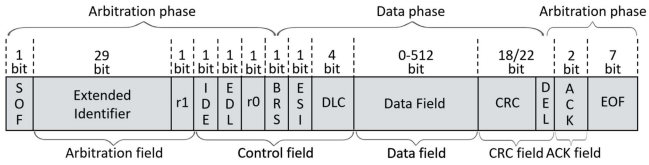


Fig. 2. Typical CAN-FD extended frame structure.

enhanced reverse engineering method for parsing signal boundaries in CAN data loads by analyzing the time series of bit flips. Verma et al. [47] established a vehicle independent CAN decoder through introducing a web of changing dependencies, which can reverse engineer the signal definitions in DBC files. Dos et al. [48] exploited standard DBC files to decode and preprocess OBD data and provided a set of CAN datasets collected under driving conditions.

In addition, in the field of security, researchers also utilized DBC files. Pesé et al. [49] analyzed the DBC files to determine the ECU nodes and CAN IDs in the network, and grouped nodes based on the CAN IDs. Lambert et al. [50] designed a scalable and reconfigurable vehicle network research and testing platform, which can call DBC files for conducting in-vehicle safety research. Bella et al. [51] described how to achieve secure and reliable frame transmission between ECUs through rules defined in DBC files, especially the generation and update mechanisms of freshness values and frame counters. Abd El-Gleel et al. [52] added a 1-byte counter in the application layer of each message defined in the DBC file to identify the continuity of the message sequence. However, existing solutions cannot implement group collaboration mechanisms and do not combine communication logic and security policies to achieve end-to-end intra group message authentication and key management.

III. Preliminaries and Background

This section introduces the CAN-FD bus, and the cryptographic techniques employed in the scheme.

A. CAN-FD Network and DBC File

CAN bus is a multi-master serial communication protocol characterized by priority arbitration and error detection [53]. Multiple nodes can send and receive messages on the bus, and each node decides whether it needs to receive a message based on demand. Traditional CAN has a maximum baud rate of 1 Mbps and a maximum payload of 8 bytes.

CAN-FD [54] is an enhanced version of the CAN protocol with higher data rates and larger data frames. In addition, the CAN-FD bus supports flexible data rates, with lower rates for the arbitration segment and higher rates for the data segment within the same frame. Similar to the CAN network, the frame format on the CAN-FD bus can be divided into standard frames and extended frames. The main differences are the length of the identifier and the structure of the arbitration field. The identifier filed in standard frames is 11-bit, while in extended frames is 29-bit. Fig. 2 shows a typical CAN-FD extended frame structure.

In conventional cognition, the data transmission modes of CAN and CAN-FD buses are considered as broadcast modes. However, in the actual manufacturing process, the receivers, messages, signals, IDs, and other key information of each node on the buses are determined by the DBC file during design [55]. DBC file is a standard format file for describing the bus communication protocol. It is widely used in the automotive industry to define and manage communications. For security and design considerations, the receiver of each message is determined based on functional requirements.

Since the structure and protocol of the CAN bus were developed without cellular or wireless communication technologies and networking of vehicles, the CAN and CAN-FD protocol standards were not designed with resistance to external adversaries. The vulnerability of the CAN-FD bus is mainly demonstrated in the following aspects: 1) Broadcast communication: Messages on the bus can be received by all nodes including external adversaries. 2) No authentication: The lack of identity legitimacy verification allows external nodes to access the bus without any measures. 3) Data transmission in plaintext: Any malicious attacker who can intercept the data frames can access the payload in the frames. Thus, security mechanisms need to be designed for CAN-FD.

B. Shamir's Secret Sharing

In 1979, Shamir and Blakley et al. [56], [57] proposed a basic version of the secret sharing scheme, i.e., (k, n) Shamir's Secret Sharing (SSS) scheme, where k is the threshold and n is the number of group members. SSS is a threshold secret splitting scheme based on the Lagrange interpolation formula. In the (k, n) SSS scheme, the trusted agent generates a $(k-1)$-degree polynomial $f(x) = a_0 + a_1 x + \ldots + a_{k-1}x^{k-1} \ mod \ p$, where the secret $s = a_0$. The agent securely distributes secret share to each member i $(i \in 1, 2, \ldots, n)$. To recover the secret, at least k members are required to exchange shares (i.e., k points on the polynomial) and recover the secret s via the Lagrange interpolation theorem $f(x) = \sum_{i=1}^{k} x_i \prod_{j=1, j\neq i}^{k} \frac{x-x_j}{x_i-x_j} \ mod \ p$. Moreover, the Lagrange interpolation theorem possesses a special property $(x, f(x)) + (x, g(x)) = (x, f(x) + g(x))$, which can be used for splitting and combining secrets.

C. Chebyshev Chaos Mapping

Assuming there are two real numbers $n \in Z^+$ and $x \in [-1, 1]$, the Chebyshev polynomial is represented as $T_n(x) : [-1, 1] \rightarrow [-1, 1]$, defined as $T_n(x) = cos(n \cdot arccos(x))$. The recurrence relationship of the Chebyshev polynomials is denoted as $T_n(x) = 2xT_{n-1}(x) - T_{n-2}(x)$, where $n \geq 2$, $T_0(x) = 1$, and $T_1(x) = x$ [58].

The Chebyshev polynomial is closed under the semigroup property, that is $T_k(T_r(x)) = cos(k \cdot arccos(cos(r \cdot arccos(x)))) = T_{kr}(x)$, where $k, r \in Z^+$, $x \in [-1, 1]$ [59].

In 2008, Zhang et al. [60] proved that the Chebyshev polynomial map still has semigroup properties on the interval $(-\infty, +\infty)$. And the extended Chebyshev polynomial can be represented as $T_n(x) = 2xT_{n-1}(x) - T_{n-2}(x) \ mod \ p$, where p is a large prime number and $x \in (-\infty, +\infty)$.

Definition 1 (Chaotic-maps discrete logarithm problem (CMDLP)): For given a real number $x \in (-\infty, +\infty)$ and an

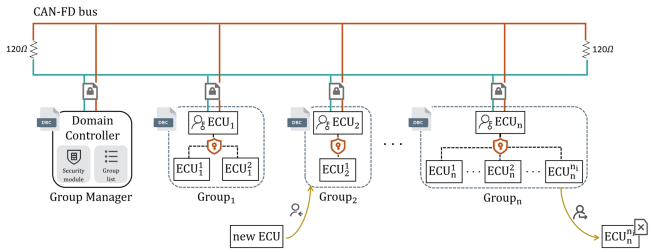


Fig. 3. System model.

extended Chebyshev polynomial $y = T_r(x)\ mod\ p$, CMDLP means that it is hard for a probabilistic polynomial-time (PPT) adversary $\mathcal{A}$ to compute the secret number r. In other words, the probability $Adv_{\mathcal{A}}^{CMDLP}(t)$ of a PPT adversary $\mathcal{A}$ within a time bound t to solve the CMDLP problem is eligible, which can be denoted as $Adv_{\mathcal{A}}^{CMDLP}(t) = Pr[\mathcal{A}(x, y) = r : r \in Z_p^*, y = T_r(x)\ mod\ p]$. Let ϵ be a negligible function, it can be known that $Adv_{\mathcal{A}}^{CMDLP}(t) \leq \epsilon$.

Definition 2 (Chaotic-maps computational Diffie-Hellman problem (CMCDHP)): For given a real number $x \in (-\infty, +\infty)$ and two extended Chebyshev polynomials $T_k(x)\ mod\ p$, and $T_r(x)\ mod\ p$, CMCDHP implies that it is hard for a PPT adversary $\mathcal{A}$ to compute $T_{kr}(x)\ mod\ p$. In other words, the probability $Adv_{\mathcal{A}}^{CMCDHP}(t)$ of a PPT adversary $\mathcal{A}$ within a time bound t to solve the CMCDHP problem is eligible, which can be denoted as $Adv_{\mathcal{A}}^{CMCDHP}(t) = Pr[\mathcal{A}(x, T_k(x)\ mod\ p, T_r(x)\ mod\ p) = T_{kr}(x)\ mod\ p : k, r \in Z_p^*]$. Let ϵ be a negligible function, it can be known that

$$Adv_{\mathcal{A}}^{CMCDHP}(t) \leq \epsilon \tag{1}$$

IV. System Model and Goals

In this section, we describe the system model, threat model, security and functional objectives of the scheme.

A. System Model

Before introducing the proposed scheme, the following assumptions are made: 1) Under the domain controller architecture, multiple domain controllers are equipped to manage each functional domain [11]. 2) Each functional domain contains a high-performance domain controller and multiple low-performance ECUs. 3) All ECUs have a piece of tamper-resistant storage for securely storing the keys and secret parameters, such as the embedded security solution Arm TrustZone, supported by STM32L5 and STM32U5 series [61]. 4) The vehicle manufacturer designs a DBC file to describe the database of the in-vehicle communication network.

By abstracting the CAN-FD network under the domain EEA architecture, the system model of the proposed scheme is shown in Fig. 3. Three kinds of entities are involved: domain controller, group leader ECU, and group member ECU.

1) *Domain Controller (DC):* It is considered to be a fully trusted controller with powerful computing and communication capabilities. It can manage, authenticate, and distribute secret keys to all ECUs within the domain.
2) *Group Leader:* Each ECU with receivers is regarded as the group leader of a group. The group is composed of the ECU and all its message receivers. The group leader is responsible for managing the group members and generating and distributing the parameters.
3) *Group Member:* As long as the ECU can receive messages, it is a group member with the sender as the group leader. Group members are required to receive secret parameters and securely store them.

In the proposed scheme, ECUs are grouped based on the sending and receiving relationships defined in DBC. Each sender ECU and its corresponding recipients of all messages will be divided into a group, with the sending ECU defined as the group leader and the receiving ECU defined as the group member. In a typical and effective CAN-FD network, each ECU is the leader of a group. The purpose of grouping is to facilitate the sharing of a group key between a group of ECUs with sending and receiving relationships and to facilitate the encryption and authentication of messages during communication. Although each ECU needs to manage multiple group keys, it is worthwhile to improve the efficiency of message transmission by increasing limited storage space.

During vehicle manufacturing, manufacturers embed the sending and receiving relationships in the DBC file in the form of a list into DC. DC first authenticates the legitimacy of all ECUs in the list. After successful authentication, key negotiation will be executed. It can divided into a centralized non-real-time secret share distribution stage and a decentralized real-time key agreement stage. In the centralized stage, DC generates partial secret polynomials for all group members and distributes secret shares to them. By accumulating all the partial polynomials, DC generates the group base polynomial and sends it to the group leader. In the decentralization stage, the group leader selects parameter k and generates the group full polynomial based on the base polynomial. All group members calculate the group key by receiving a confidential k. By offloading high computational and communication overhead operations to the non-real-time stage, delays in real-time vehicle communication are avoided and the single point of failure is avoided as much as possible. Each time the vehicle starts, group members only need to perform one symmetric decryption and multiplication operation to obtain the group key. The same applies to group key updates. The design of three-layer polynomials (partial, base, and full polynomials) makes DC, group leader and all group members participate in the group key generation process and realize the contributory group key generation.

After the key distribution is finished, all ECUs will encrypt and send messages with their own group keys. The proposed scheme integrates the encryption and authentication operations into a single algorithm, i.e., the TinyJAMBU algorithm. TinyJAMBU is a lightweight and highly efficient AEAD algorithm, which can provide confidentiality, integrity, and authentication at the same time. It is highly secure and can effectively resist various typical attacks [62]. Since the TinyJAMBU algorithm has low memory and energy requirements, it is widely used in Internet of Things, wireless sensor networks, and embedded systems [63], [64], [65]. The TinyJAMBU-128 algorithm

generates ciphertexts of the same length as the plaintexts, and the authentication tag is 64 b, which requires an extra authentication frame to send. Despite the need for an additional frame, for a CAN-FD network with a maximum payload of 512 b, the additional 64 b only accounts for $\frac{1}{8}$, so the impact on the bus load and communication overhead is quite limited.

In addition, due to the fact that the lifespan of vehicles can reach 8-35 years [66], the installation, disassembly, and replacement of vehicle ECUs are highly likely to occur [44]. The proposed solution also considers group members joining and leaving. The group key update methods for joining and leaving scenarios enhance the scalability of the scheme and improve the flexibility of real-world applications.

B. Threat Model

The adversary is considered to be from outside the vehicle. We consider the widely used Dolev-Yao (DY) threat model [67]. The attacker has the ability to access all messages on the public network, initiate sessions with nodes on the network, and send messages by impersonating legitimate entities. Therefore, the adversary can eavesdrop, forge, impersonate, tamper, replay, and delete messages in the network. However, the attacker is unable to obtain keys and secret parameters that are securely stored inside ECUs. It is worth noting that although the DY model presents an idealized abstraction of an adversary's capabilities, the rapid advancement of automotive cybersecurity technologies and regulatory frameworks has made its key assumptions increasingly feasible in real-world settings. For instance, the widespread adoption of Hardware Security Modules (HSMs) [68], TrustZone-based isolated execution environments, and compliance with standards such as ISO/SAE 21434 and the UN R155 [69] regulation provide reliable safeguards for both key confidentiality and the physical protection of high-entropy random values.

C. Security and Functional Objectives

The proposed scheme needs to satisfy the following security objectives: 1) Message confidentiality: Messages must be encrypted before being sent. 2) Message integrity: The integrity of the message should be able to be verified by the receiver. 3) Authentication: Only legitimate ECUs can send messages on the bus. 4) Forward security: The group key will be updated when a member leaves the group. 5) Backward security: When a member joins the group, the group key will be updated. 6) Resistance to replay attacks: Each message is appended with a timestamp or counter to ensure freshness.

The following functional objectives are needed to enhance the adaptability and practicality in real-world applications: 1) Decentralized key agreement: To avoid a single point of failure as much as possible, the participation of DC is avoided in the real-time key agreement. 2) Overcoming the reliance on long-term keys: There is more than one supplier of ECUs in automobile manufacturing, it is vital to design a security scheme without preassembling long-term keys. 3) Scalability: Considering the installation and removal of ECUs, designing algorithms for ECUs to join and leave the network is crucial.

TABLE II
NOTATIONS AND DESCRIPTIONS

Notation	Description
DC, ECU_i/GL_i	Domain controller, i^{th} ECU/Group leader of G_i
G_i	Group composed of ECU_i and its recipients
GM_i^j	j^{th} group member of group G_i
ID_{DC}, ID_i	Identities of DC and ECU_i
PK_{DC}, PK_i	Public keys for DC and ECU_i
S_{DC}, S_i	Private keys for DC and ECU_i
SK_i, SK_i^j	Session keys between DC and ECU_i, GM_i^j
TK, TGK	Temporary global key, temporary group key
$IniGrpList, RtGrpList$	Initial group list, real-time group list
n	Total number of ECUs in a CAN-FD subnet
n_i	Number of ECU_i's recipients
$f_j(x)$	Partial secret sharing polynomial of GM_i^j
$ts, \Delta T$	Current timestamp, validity period of message
msg_i, ad_i	Message, associated data sent by ECU_i
c_i	Ciphertext sent by ECU_i
$E_k(\cdot)/D_k(\cdot)$	Symmetric encryption/decryption using key k
$H_k(\cdot)$	Message authentication code using key k
$T(\cdot), KDF(\cdot)$	Chebyshev polynomial, key derivation function

4) Minimize the computation and communication overhead: The CAN-FD network is high real-time and low latency, it is essential to minimize the computation and communication overhead and reduce the impact on the real-time operation.

V. The Proposed Scheme

In this section, we propose a semi-decentralized secure communication mechanism based on DBC files for CAN-FD bus. Table II lists the notations.

A. System Initialization

When the vehicle first starts (usually during the vehicle testing process), all ECUs and DC generate keys and public parameters. The process is only executed once throughout the lifecycle of the vehicle. The process contains several steps.

1) DC selects a large prime number p and a random integer $x \in Z_n^*$ as the seed of Chebyshev polynomial.
2) DC selects $S_{DC} \in Z_p$, where Z_p is a finite field of order p. DC calculates its own public key $PK_{DC} = T_{S_{DC}}(x)\ mod\ p$ and makes it public. The public and private keys can be updated during OTA upgrades, and the frequency can be determined by each vehicle company.
3) ECU_i selects $S_i \in Z_p$ $(i = 1, 2, \ldots, n)$, calculates its public key $PK_i = T_{S_i}(x)\ mod\ p$, and makes it public.
4) DC groups ECUs based on the DBC file of the vehicle. DC lists the group formed by ECU_i and all its receivers as G_i $(i = 1, 2, \ldots, n)$.
5) DC denotes ECU_i as the group leader GL_i and the number of members in group G_i as n_i. As a consequence, the total number of nodes in group G_i is $n_i + 1$. Then DC maintains $IniGrpList$ in the form of $\langle G_i, GL_i, GM_i^j \rangle$ $(i = 1, 2, \ldots, n, j = 1, 2, \ldots, n_i)$.
6) Group leader GL_i stores $IniGrpList$.

B. ECU Authentication

Every time the vehicle starts, DC authenticates all ECUs. ECUs that do not pass the authentication will be eliminated

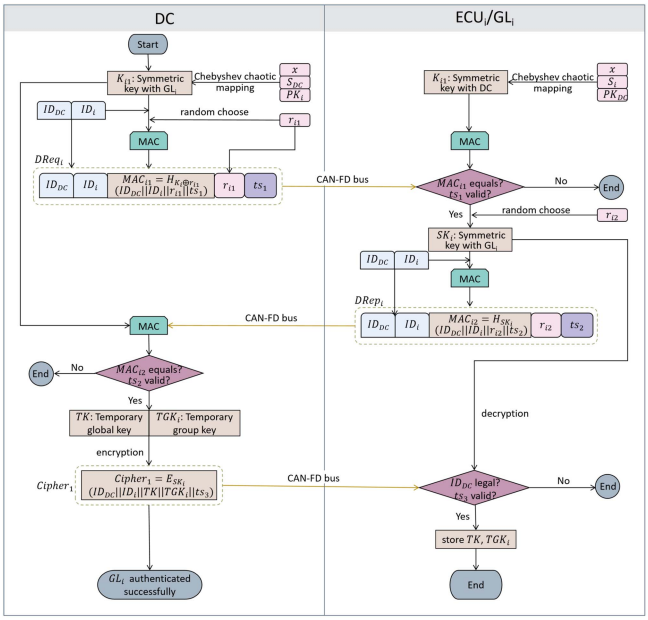


Fig. 4. ECU authentication.

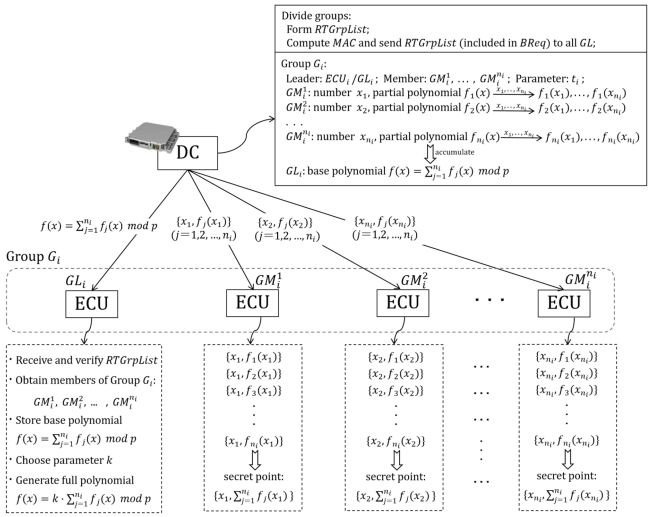


Fig. 5. Group secret share distribution.

from $IniGrpList$, and then form the $RtGrpList$. The process is shown in Fig. 4.

1) DC uses GL_i's public key to pre-compute $K_i = T_{S_{DC}}(T_{S_i}(x)) \bmod p$ and generates a random number r_{i1} and current timestamp ts_1. Based on K_i, DC computes $MAC_{i1} = H_{K_i \oplus r_{i1}}(ID_{DC}||ID_i||r_{i1}||ts_1)$ and sends $DReq_i = ID_{DC}||ID_i||r_{i1}||ts_1||MAC_{i1}$ to GL_i.
2) Once GL_i receives $DReq_i$, it computes the symmetric session key $K_i = T_{S_i}(T_{S_{DC}}(x)) \ mod \ p$ with DC and verifies MAC_{i1}. The key can also be pre-computed to reduce the computational overhead. If MAC_{i1} and ts_1 verified successfully, GL_i chooses a random number r_{i2} and computes $SK_i = K_i \oplus r_{i1} \oplus r_{i2}$ to generate $MAC_{i2} = H_{SK_i}(ID_{DC}||ID_i||r_{i2}||ts_2)$. GL_i sends $DRep_i = ID_{DC}||ID_i||r_{i2}||ts_2||MAC_{i2}$ to DC.
3) GL_i computes $SK_i = K_i \oplus r_{i1} \oplus r_{i2}$ and verifies whether MAC_{i2} equals $H_{SK_i}(ID_{DC}||ID_i||r_{i2}||ts_2)$. Afterward, DC generates a temporary global key TK and a temporary group key TGK_i for group G_i and encrypts keys through computes $Cipher_1 = E_{SK_i}(ID_{DC}||ID_i||TK||TGK_i||ts_3)$. Ultimately, $Cipher_1$ is transmitted to GL_i.
4) GL_i decrypts $Cipher_1$ to obtain TK, TGK_i, and ts_3. GL_i verifies ts_3, and if it is within the valid time interval, securely stores the keys TK and TGK_i.

C. Group Secret Share Distribution

After the authentication of all ECUs, DC is responsible for transmitting the grouping result $RtGrpList$ to all group leaders based on the authentication result and $IniGrpList$ and distributing the group secret share to members of each group. The group leader will be informed of the secret polynomial for each group. This step is performed after the first ignition of the vehicle, after which DC uses each ECU's symmetric key to update each group's secret share irregularly and in non-real-time situations. Before executing this step, the ECU authentication phase must be carried out. The secret share of each group can be updated separately to reduce the computational and communication overhead. Fig. 5 displays the process of group secret share distribution.

1) DC publishes $RTGrpList$ and generates a MAC value $MAC = H_{TK}(ID_{DC}||RTGrpList||ts_4)$. Then DC broadcasts $BReq = ID_{DC}||RTGrpList||ts_4||MAC$ to all ECUs.
2) GL_i receives requests and verifies MAC and ts_4. After successful verification, GL_i saves the $RTGrpList$.
3) For group G_i $(i = 1, 2, \ldots, n)$, DC determines the threshold t_i first. Then, DC selects random numbers X_l and x_l of for each member $(l = 1, 2, \ldots, n_i)$ and generates a partial secret share polynomial of $t_i - 1$ degree. For each partial secret sharing polynomial, DC inputs x_l and outputs $\{x_l, f_j(x_l)\}$ $(j = 1, 2, \ldots, n_i, l = 1, 2, \ldots, n_i)$.
4) For each member of group G_i, DC sends n_i points to GM_i^j through the symmetric key SK_i^j. As a result, each group member in group G_i obtains n_i points according to the partial secret-sharing polynomial. Each member eventually securely stores n_i secret shares.
5) DC obtains the base polynomial of group G_i by accumulating the partial polynomials of all members. It then generates $t_i - 1$ public points based on the base polynomial and distributes the $t_i - 1$ public points to all members in group G_i based on TGK_i.
6) By accumulating the n_i points obtained based on the partial polynomial, GM_i^j can obtain the secret point $\{x_l, \sum_{j=1}^{n_i} f_j(x_l)\}$. It can compute $seed_i$ by the Lagrange interpolation theorem with $t_i - 1$ public points and a secret point.
7) DC sends the base secret polynomial of each group to the group leader using the key SK_i.

D. Group Key Agreement

GL_i distributes materials for negotiating group keys to all members of its group, and group members can calculate the

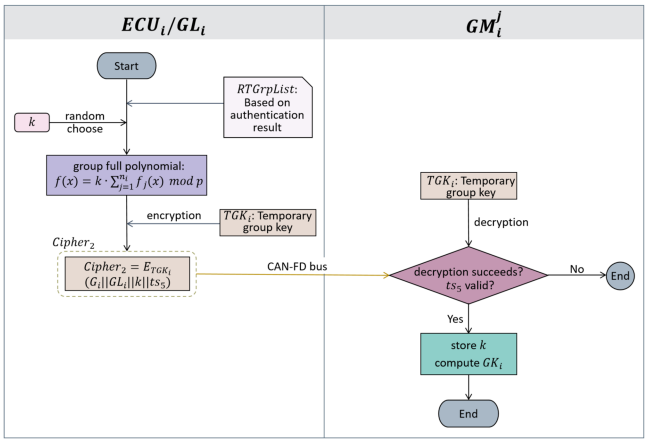


Fig. 6. Group key agreement.

group keys. This process is required to be performed every time the vehicle is ignited. The group key agreement is shown in Fig. 6.

1) GL_i randomly selects k to generate the full secret-sharing polynomial for group G_i. The coefficient k is encrypted with TGK_i and $Cipher_2 = E_{TGK_i}(G_i, GL_i, k, ts_5)$ is sent by GL_i to all members.
2) All members decrypt $Cipher_2$ to obtain the coefficient k of the full secret-sharing polynomial. By calculating $k * seed_i \bmod p$, the group key GK_i is obtained.

E. Secure Communication

Upon group key negotiation is completed, all members of each group jointly manage a message counter ctr_i for synchronization. As long as the receiver successfully obtains data, all members will make the counter add once. According to the characteristics of a short authentication tag (the shorter the better), operation efficiency (as lightweight as possible), and security (high-security level), we determine to employ TinyJAMBU-128 to realize encryption and authentication. TinyJAMBU-128 requires only a 96 b random number and a 64 b tag for encryption, where the random number is filled by the message counter ctr_i in the proposed scheme.

1) Before the message is sent, the sender ECU makes use of TinyJAMBU-128 to encrypt and authenticate the message msg_i with $E_{GK_i}(ctr_i, msg_i, ad_i) \rightarrow (c_i, tag_i)$. (c_i, tag_i) is sent on the bus. The additional data ad_i contains the sender's ECU identity information ID_i. ad_i is also used to generate the authentication tag.
2) Since the receiver gets the ciphertext, it authenticates the tag and decrypts the ciphertext $D_{GK_i}(c_i, tag_i) \rightarrow msg_i/\perp$. If the authentication passes and the decryption is successful, the counter will be added once. Otherwise, it will output $\perp$ and the counter remains unchanged.

F. Group Key Update

Group keys need to be updated periodically. Group keys can be refreshed separately to reduce the impact of computational and communication costs on real-time networks.

1) GL_i uses the current group key GK_i to derive two keys $KDF(GK_i) = GK_{i1}||GK_{i2}$.
2) GL_i selects k' and generates a new full secret sharing polynomial and a new group key $GK_i' = seed_i * k'$.
3) GL_i encrypts k' with GK_{i1} to generate $Cipher_3 = E_{GK_{i1}}(ID_i, G_i, GL_i, k', ts_5)$ and $MAC_4 = H_{GK_{i2}}(Cipher_3)$. Then it broadcasts $UReq = Cipher_3||MAC_4$ to all members.
4) GM_i^j decrypts $Cipher_3$ to get k', the new group key GK_i' can be computed by multiplying k' with $seed_i$.

G. Member Join

Replacing or installing ECUs is done at a 4S store when the vehicle is being serviced. Due to unknown security, it is safer to have the system automatically executed without human intervention. Supposing ECU_{join} is installed on the bus.

1) DC authenticates ECU_{join} as shown in the ECU Authentication phase. If verification is successful, the session key SK_{join} with DC is saved.
2) For the group with ECU_{join} as the sender, DC helps create a new group G_{join}. For this group, it needs to execute the group secret share distribution and group key agreement phases to obtain the group key GK_{join}.
3) For the group with ECU_{join} as the recipient, the group key needs to be updated. Assuming that the group leader is GL_i, the specific process is shown as follows.
   - DC generates the partial polynomial, points $\{x_i, f_{join}(x_i)\}(i \in 1, 2, \ldots, n_i)$ and $\{x_{join}, f_{join}(x_{join})\}$. Points are sent to all group members and ECU_{join} respectively.
   - DC forms a new group base polynomial and computes and publishes $t_i - 1$ public points.
   - Group members and ECU_{join} use the Lagrange interpolation theorem to compute $seed_{join}$.
   - DC encrypts the new group base polynomial and sends it to GL_i. GL_i decrypts and obtains the group base polynomial.
   - DC publishes the MAC value of the real-time grouping result. GL_i verifies it and obtains the new list.
   - GL_i randomly selects k_{join} and generates a new full polynomial for the group and a new group key $GK_{join} = k_{join} * seed_{join} \bmod p$.
   - GL_i encrypts k_{join} and sends it to all group members. Group members use k_{join} to generate GK_{join}.

H. Member Leave

For the ECU removing scenario, similar to the ECU joining, the group key update is also performed in the form of system auto-execution. The specific process is as follows.

1) For the group with ECU_{leave} as the sender, the group is automatically dissolved.
2) For the group with ECU_{leave} as the receiver, the group key needs to be updated. Assuming that ECU_{leave} performs the role of GM_i^j and the group leader is GL_i.
   - DC removes ECU_{leave}'s partial polynomial from the base polynomial. DC computes and distributes $t_i - 1$

public points to members. DC sends the new base polynomial to GL_i.

- Group members (except ECU_{leave}) delete the points belonging to ECU_{leave} and re-accumulate the remaining points to generate a new secret point. Using $t_i - 1$ public points and a secret point of their own, the group members compute a new $seed_{leave}$ according to the Lagrange interpolation theorem.
- GL_i deletes ECU_{leave}'s partial polynomial from the old base polynomial, selects k_{leave}, and generates a new full polynomial. It encrypts k_{leave} and sends it to DC, which distributes it to all group members (except ECU_{leave}) using SK.
- Group members calculate $k_{leave} * seed_{leave} \ mod \ p$ to get the final group key GK_{leave}.

VI. Security Proof and Analysis

In this section, the widely used Random Oracle Model (ROM) [58], [59], [60] is employed to prove the security, followed by a security analysis. Finally, we compare the proposed scheme with existing related frameworks.

A. Security Model

First, we define the primitives related to the security model.

- *Participants:* We denote participants as Λ, $\Lambda \in \{DC, \ GL_i\}$. Each participant has multiple independent instances, i.e., oracles. The instances are denoted as Π^t_Λ, and the specific instances are denoted as Π^u_{DC}, $\Pi^v_{GL_i}$. Let the proposed scheme be P.
- *Partnership:* Instances Π^u_Λ and Π^v_Λ are partners of each other if they fulfill the following three requirements: 1) Π^u_Λ and Π^v_Λ interact during the same session cycle; 2) Π^u_Λ and Π^v_Λ share the same session key SK; 3) Π^u_Λ and Π^v_Λ are mutual partners of each other.
- *Queries:* A series of games between challenger $\mathcal{C}$ and PPT adversary $\mathcal{A}$ are used to define a security model. In this series of games, the interactions between $\mathcal{A}$ and the protocol participants will take place by means of oracle queries. This approach models the adversary's capabilities in real attacks. The following are the types of queries that adversary $\mathcal{A}$ can execute:
  -$Execute(\Pi^u_{DC}, \Pi^v_{GL_i})$: This query models the ability of $\mathcal{A}$ to perform channel eavesdropping. Upon receiving this query, $\mathcal{C}$ returns all messages that instances Π^u_{DC} and $\Pi^v_{GL_i}$ interacted with during the authentication phase.
  -$Reveal(\Pi^t_\Lambda)$: This query simulates $\mathcal{A}$ obtaining a negotiated session key between Π^t_Λ and its partners. When this query is received, $\mathcal{C}$ returns the negotiated session key between the instance and its partners.
  -$Corrupt(\Pi^t_\Lambda)$: This query simulates the ability of $\mathcal{A}$ to corrupt a user. If $\mathcal{A}$ employs the identity of Π^t_Λ to execute the query, $\mathcal{C}$ sends the private key corresponding to the instance to $\mathcal{A}$.
  -$Send(\Pi^t_\Lambda, m)$: This query simulates $\mathcal{A}$'s ability to perform an active attack. $\mathcal{A}$ sends a message and $\mathcal{C}$ returns a response message according to the protocol.
  -$Test(\Pi^t_\Lambda)$: This query models the semantic security of the session key. When the query is received, $\mathcal{A}$ executes the query to challenge, and $\mathcal{C}$ flips a coin to obtain a random bit $b \in \{0, 1\}$. If $b = 0$, $\mathcal{C}$ returns the real session key to $\mathcal{A}$. Otherwise, $\mathcal{C}$ returns a random string of the same length as the real session key to $\mathcal{A}$. If the session key has not yet been generated or the instance is not fresh, $\perp$ is returned.
- *Freshness.* Instance Π^t_Λ is claimed to be fresh if it meets the following requirements: 1) Instance Π^t_Λ has computed a session key SK; 2) Π^t_Λ and its partner have not made a *Reveal-query*.

Definition 3 (AKA Security): In the ultimate phase of games, $\mathcal{A}$ needs to execute a query on a fresh instance Π^t_Λ and send a guessing bit b' to $\mathcal{C}$. Assuming that $Succ(\mathcal{A})$ represents the event where $\mathcal{A}$ initiates a query and correctly reveals b, that is, $\mathcal{A}$ wins the game, $b' = b$. The advantage of $\mathcal{A}$ breaking the proposed scheme is $Adv^{AKA}_P(\mathcal{A}) = 2|Pr(Succ(\mathcal{A}) - 1)|$. If the advantage can be ignored for any probability polynomial adversary, then the proposed scheme satisfies AKA security.

B. Formal Security Proof

Next, we will prove that the proposed scheme satisfies AKA security under the above security model.

Theorem 1: Supposing $Adv^{AKA}_P(\mathcal{A})$ is the probability that a PPT adversary $\mathcal{A}$ successfully breaks the semantic security of scheme in polynomial time t. If $\mathcal{A}$ can break scheme with an advantage $Adv^{AKA}_P(\mathcal{A})$, then a PPT adversary $\mathcal{B}$ can solve CMCDHP with an advantage $Adv^{CMCDHP}_P(\mathcal{B})$. More precisely, $Adv^{CMCDHP}_P(\mathcal{B})$ can respond $T_{\theta\gamma}(x)$ against instance $(x, T_\theta(x), T_\gamma(x))$, where θ and γ are two integers. Let q_s, q_h, and q_e represent times of *Hash-query*, *Send-query*, and *Execute-query*, respectively. Hence, we have

$$Adv^{AKA}_P(\mathcal{A}) \le q_h Adv^{CMCDHP}_P(\mathcal{B}) + \frac{3q_h^2 + q_s^2}{2^l} + \frac{q_h^2 + (q_s + q_e)^2}{2p} \quad (2)$$

Proof: Multiple games $G_0, G_1, \ldots, G_5$ are set up between challenger $\mathcal{C}$ and adversary $\mathcal{A}$ to model the interaction of the scheme. Let E_i denote the event that $\mathcal{A}$ outputs the correct bit b in G_i, where $i = 0, 1, \ldots, 5$.

Game G_0: This game uses a random oracle to simulate an attack on the real protocol. $\mathcal{A}$ can launch queries to all oracles and $\mathcal{C}$ will interact with $\mathcal{A}$ according to the real protocol. As a consequence, the probability that $\mathcal{A}$ wins the game is equal to the probability that the real protocol P is compromised. According to the definition of semantic security [70], it can be obtained that $Adv^{AKA}_P(\mathcal{A}) \le |2Pr[E_0] - 1|$.

Game G_1: This game has the same oracles as G_0 except that it maintains a list of hash oracle answers. Before the game starts, $\mathcal{C}$ will maintain a list that stores messages and corresponding hash values. Upon receiving a query from $\mathcal{A}$ about message m, $\mathcal{C}$ first checks the list. If a hash value corresponding to m exists, $\mathcal{C}$ returns the value to $\mathcal{A}$. If there is no hash value corresponding to m, $\mathcal{C}$ chooses a random string $h(m) \in Z^*_q$, records $(m, h(m))$

and returns $h(m)$ to $\mathcal{A}$. The queries that $\mathcal{A}$ can execute are shown below:

1) $Hash-query$: For this query, $\mathcal{C}$ will look up the corresponding record in the list. If $\mathcal{C}$ finds it, it will return the corresponding hash value to $\mathcal{A}$. Otherwise, $\mathcal{C}$ will randomly select a string equal to the length of the hash value, and add the value to the list. The value is returned to $\mathcal{A}$.
2) $Send(\Pi^u_{DC},\ start)-query$: This query simulates an active attack launched by $\mathcal{A}$. For this query, $\mathcal{C}$ will randomly select the number r_{i1}, compute $MAC_{i1}=H_{K_i\oplus r_{i1}}(ID_{DC}||ID_i||r_{i1}||ts_1)$, and then send $DReq_i=ID_{DC}||ID_i||r_{i1}||ts_1||MAC_{i1}$ to $\mathcal{A}$.
3) $Send(\Pi^v_{GL_i},DReq_i)-query$: This query simulates an active attack launched by $\mathcal{A}$. $\mathcal{C}$ will choose a random number r_{i2}, compute $SK_i=K_i\oplus r_{i1}\oplus r_{i2}$, generate $MAC_{i2}=H_{SK_i}(ID_{DC}||ID_i||r_{i2}||ts_2)$, and then send $DRep_i=ID_{DC}||ID_i||r_{i2}||ts_2||MAC_{i2}$ to $\mathcal{A}$.
4) $Send(\Pi^u_{DC},DRep_i)-query$: This query simulates an active attack launched by $\mathcal{A}$. For this query, $\mathcal{C}$ will randomly select TK and TGK_i, generate $Cipher_1=E_{SK_i}(ID_{DC}||ID_i|||TK||TGK_i||ts_3)$, and send it to $\mathcal{A}$.
5) $Execute(\Pi^u_{DC},\Pi^v_{GL_i})-query$: This query simulates the eavesdropping attack initiated by $\mathcal{A}$. In other words, $\mathcal{A}$ can intercept all messages during the communication between DC and GL_i, i.e., $DReq_i,\ DReq_i, Cipher_1$.
6) $Reveal(\Pi^u_{DC},\Pi^v_{GL_i})-query$: $\mathcal{C}$ returns the session key SK established between DC and GL_i.
7) $Corrupt(\Pi^u_{\Lambda})-query$: This query simulates the ability of $\mathcal{A}$ to disrupt the user. For this query, $\mathcal{C}$ will return the private key of the instance.
8) $Test(\Pi^u_{\Lambda})-query$: This query obtains the session key SK from $Reveal-query$, then $\mathcal{C}$ will flip a coin b. If $b=0$, $\mathcal{C}$ generates a random string of length equal to SK and returns it to $\mathcal{A}$. Otherwise, $\mathcal{C}$ returns SK to $\mathcal{A}$.

Since all the oracles in the game are modeled as real attacks, the game G_0 and the actual execution of the protocol are indistinguishable, i.e., G_0 and G_1 are indistinguishable. It follows that the advantage of $\mathcal{A}$ to break the protocol in G_0 is equal to that in G_1, i.e., $Pr[E_1]=Pr[E_0]$.

Game G_2: G_2 adds the condition that there is no collision based on G_1. In this game, the simulation will be aborted if: (1) The outputs of the *Hash-query* collide; (2) The message copies $DReq_i$, $DRep_i$, and $Cipher_1$ collide. Assuming that p is all possible outputs of the hash function, the maximum probability of collision of the hash function h is $\frac{q_h^2}{2p}$ according to the birthday paradox theorem. The maximum probability of a message collision is $\frac{q_h^2+(q_s+q_e)^2}{2p}$, which is related to the number of *Send* and *Execute-queries*. Thus we have

$$|Pr\left[E_2\right]-Pr\left[E_1\right]|\leq\frac{q_h^2+\left(q_s+q_e\right)^2}{2p} \tag{3}$$

Game G_3: G_3 adds a protocol abort condition to G_2. If $\mathcal{A}$ forges the correct MAC_{i1} and MAC_{i2} without querying the random oracle, the interaction is aborted. This condition occurs only when $\mathcal{A}$ initiates *Send-queries*. The game G_3 is indistinguishable from G_2 unless: GL_i refuses to receive the correct element MAC_{i1}, and DC refuses to receive the correct MAC_{i2}. Consequently, it is possible to obtain

$$|Pr\left[E_3\right]-Pr\left[E_2\right]|\leq\frac{q_s^2}{2^l} \tag{4}$$

Game G_4: In G_4, if $\mathcal{A}$ guesses the session key SK without querying the random oracle, it is aborted. In other words, the session key is independent of H. Unless $\mathcal{A}$ queries the random oracle H for $ID_{DC}||ID_i||r_{i1}||ts_1$ and $ID_{DC}||ID_i||r_{i2}||ts_2$, then $\mathcal{A}$ can obtain $T_{\theta\gamma}(x)$. Thus, we obtain

$$|Pr\left[E_4\right]-Pr\left[E_3\right]|\leq q_h Adv_P^{CMCDHP}(\mathcal{B})+\frac{q_h^2}{2^l} \tag{5}$$

Game G_5: The difference between this game and G_4 is that if $\mathcal{A}$ initiates an $Hash-query$ on $ID_{DC}||ID_i||r_{i1}||ts_1$ and $ID_{DC}||ID_i||r_{i2}||ts_2$, then the game is aborted. In other words, by initiating an $Hash-query$, $\mathcal{A}$ is able to obtain SK with maximum probability. As a result, it can be obtained that

$$|Pr\left[E_5\right]-Pr\left[E_4\right]|\leq\frac{q_h^2}{2^{l+1}} \tag{6}$$

Furthermore, if $\mathcal{A}$ initiates a *Hash-query* based on valid inputs, then the probability of success of this experiment is the same as the probability of distinguishing between SK and random numbers, i.e., $|Pr[E_5]|=\frac{1}{2}$.

Therefore, based on G_0-G_5, we can obtain formula (2).

C. Informal Security Analysis

In this subsection, we analyze the security properties of the proposed scheme.

1) *Confidentiality and integrity:* The proposed scheme employs the TinyJAMBU algorithm and group key GK_i for authenticated encryption of each message. And the sender's identity information is included in the associated data. The confidentiality of the message is guaranteed since GK_i is known only to GL_i and its receivers. Besides, TinyJAMBU generates an authentication tag according to GK_i, nonce, and the processed ad_i, which is used to ensure the integrity and authenticity of the message. According to [62], the probability of success of a differential forgery attack against nonce and associated data in TinyJAMBU is at most 2^{-73}. It is infeasible for attackers to decrypt the ciphertext and tamper with the authentication tag.
2) *Authentication:* In the proposed scheme, DC authenticates all ECUs every time the vehicle is started. DC and ECU_i negotiate an initial session key SK_i. Through the ciphertext and the message authentication code, both communicating parties can obtain the random number and key generated by another party. They can ensure that the messages are generated by the legitimate identity claimed by another party.
3) *Forward security:* In the proposed scheme, after a member leaves the group, DC will delete ECU_{leave}'s partial polynomial from the base polynomial and form a new base

TABLE III
SECURITY AND FUNCTIONALITY COMPARISON

	[36]	[44]	[29]	[43]	Ours
Confidentiality	**Yes**	N/A	**Yes**	**Yes**	**Yes**
Integrity	**Yes**	N/A	**Yes**	N/A	**Yes**
Authentication	**Yes**	**Yes**	N/A	**Yes**	**Yes**
Forward security	N/A	**Yes**	N/A	N/A	**Yes**
Backward security	N/A	**Yes**	**Yes**	N/A	**Yes**
Resistance to replay attacks	**Yes**	N/A	**Yes**	**Yes**	**Yes**
Scalability	N/A	**Yes**	N/A	N/A	**Yes**
Semi-decentralized	N/A	**Yes**	N/A	N/A	**Yes**
Without long-term keys	N/A	**Yes**	N/A	N/A	**Yes**
Robustness of single point failure	N/A	**Yes**	N/A	N/A	**Yes**
Key type	Pairwise key	Group key	Global key	Group & pairwise key	Group key
Key number	$\frac{n(n-1)}{2}$	2	2	n & $\frac{n(n-1)}{2}$	n

secret polynomial. Furthermore, the group leader will select a new k_{leave} to generate a new group full polynomial, which will be distributed by DC to all members. Therefore, leaving members cannot obtain the latest coefficient k_{leave} generated by the group leader, and they cannot compute the latest group key GK_{leave}.

4) *Backward security:* For the group where the new ECU is a group member, DC will distribute the group secret share to it. The group leader generates a new k_{join} for the new base polynomial and encrypts k_{join} to ECU_{join} so that the new ECU can compute the latest group key. Since the previous group secret share and key factor k were distributed using the temporary group key TGK_i distributed by DC, the new ECU does not have access to the key. Without access to the old key factor k, the new ECU cannot compute the old group key.
5) *Resistance to replay attacks:* In ECU authentication, group secret share distribution, and group key agreement phases, each message contains the timestamp ts. Once the timestamp is found to be duplicated or expired while decrypting or authenticating, the message will be discarded. In the secure communication phase, the message sender and receivers will jointly maintain a counter, and the communicating parties will add the counter when the message is successfully sent and received, so the value of the counter is fresh at each communication.

D. Comparison of Security and Functionality

Table III compares the proposed scheme with the security and functional features of related works [29], [36], [43], [44]. In the table, "Yes" indicates that the solution meets the requirements, and "N/A" indicates that the attribute is not provided. Results show that compared with other schemes listed in the table, the proposed scheme provides better security and more comprehensive functional features.

VII. EXPERIMENTAL RESULTS AND ANALYSIS

This section implements the proposed scheme and analyses the computational cost, communication overhead and bus load.

TABLE IV
NOTATIONS AND DEFINITIONS OF RUNTIME FOR CRYPTOGRAPHIC OPERATIONS

Symbol	Definition	Symbol	Definition
T_k	KDF	T_{ea}, T_{em}	Elliptic curve point addition, point multiplication
T_m, T_h	MAC, hash	T_{ee}, T_{ed}	Elliptic curve encryption, decryption
T_a	AES encryption/decryption	T_{ss}	Schnorr signature
T_t	TinyJAMBU authenticated encryption/decryption	T_{sv}	Schnorr signature verification
T_c	Chebyshev chaotic mapping	T_{re}	RSA encryption/ decryption
T_l	Lagrange interpolation theorem	T_{rs}	RSA signature/ signature verification

A. Experimental Platform

We built the CAN/CAN-FD bus experimental platforms based on two kinds of representative automotive-grade circuit boards with different performances to evaluate the effectiveness of the proposed solution. The high-performance MCU is Raspberry Pi 3 Model B V1.2 (RPI3B for short), which has a Quad Core Broadcom BCM2837 64-bit CPU, 1 GB RAM, and 32 GB memory card. The operating system is Debian GNU/Linux 11 (bullseye), with a frequency range of 600 MHz to 1.2 GHz. The low-performance MCU is STM32H743IIT6 (STM32 for short), which is the mainstream MCU developed by STMicroelectronics [36]. The STM32H743IIT6 is based on the ARM Cortex M7 core, with a maximum main frequency of 480 MHz. It is equipped with 2048 KB FLASH, 1060 KB SRAM, and CAN communication interfaces.

To achieve CAN and CAN-FD communication, we installed a dual-channel isolated CAN-FD bus expansion board for Raspberry Pi. The expansion board is based on an MCP2518FDT-H/SL CAN controller, equipped with an MCP2562FD-E/SN transceiver. It supports CAN2.0 and CAN-FD protocols, with an SPI control interface and onboard 120Ω terminal resistors. Based on the expansion board and SocketCAN software package [71], we implemented CAN and CAN-FD communication on Raspberry Pi. STM32H743IIT6 microcontroller unit is equipped with 2 FDCAN controllers and a TPT1051 V transceiver, compliant with ISO 11898-1, supporting AUTOSAR and J1939. On STM32H743IIT6, we implemented CAN and CAN-FD communication protocols using the HAL library [72] and C language.

We employ one RPI3B and three STM32H743IIT6 MCUs to simulate a scenario where one DC manages three ECUs for the in-vehicle bus. The network topology and prototype platform are shown in Figs. 7 and 8. For both CAN and CAN-FD buses, we model them as 500 Kbps.

B. Evaluation and Analysis of Cryptographic Operations

Based on MIRACL (Multiprecision Integer and Rational Arithmetic Cryptographic) [73] library and C language, we implemented the underlying cryptographic operations on RPI3B and STM32H743IIT6 and measured the execution time.

We achieved a security strength of 128 b. The MAC is HMAC-SHA256, and the KDF function is also based on SHA-256. Except for the secure communication phase, the encryption method used is the AES-128 method in OFB mode. TinyJAMBU is

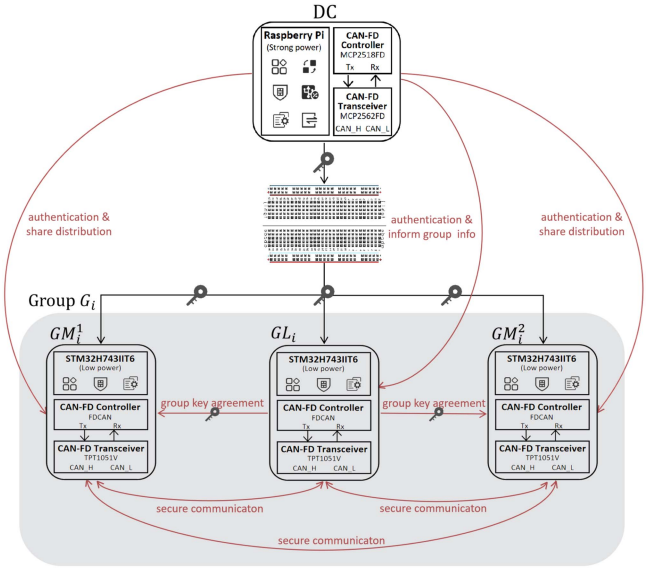


Fig. 7. Network topology.

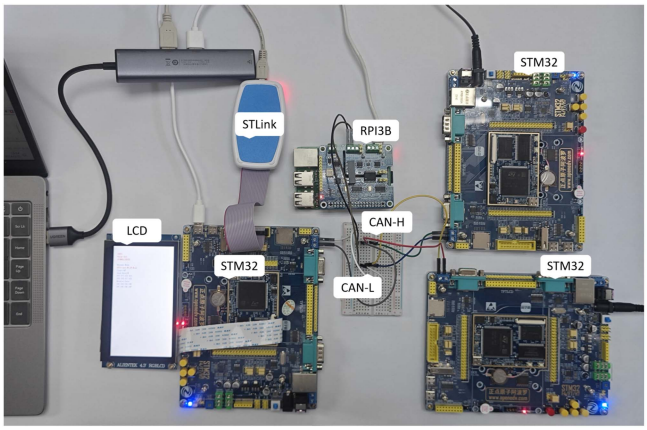


Fig. 8. Prototype platform.

implemented as TinyJAMBU-128. For SSS, we set n to 30 and the threshold t_i to 3.

For the elliptic curve cryptosystem mentioned by Sun et al. [44], we realized the 256-bit prime field Weierstrass curve BrainpoolP256r1 [74], which is $y^2 \equiv x^3 + ax + b \ mod \ p$. A cyclic group G based on an element of order q on this curve is generated, where both p and q are 256-bit prime numbers. BrainpoolP256r1 is widely supported by chip hardware [61]. Schnorr Signature mentioned in Sun et al. [44] is also realized on the BrainpoolP256r1 curve. The RSA algorithm mentioned in Lu et al.'s [36] is implemented as RSA-1024.

Taking account of applications to the CAN-FD bus, the message input is set to the maximum of 512 bits. ECU IDs are set to 8 b. Since there are mostly dozens to hundreds of existing ECUs, 2^8 types may be far more than the number of ECUs. For AES, the length of key and plaintext is 128 b and 512 b. For TinyJAMBU, the length of key, plaintext, nonce, and associated data is 128 b, 512 b, 96 b, and 64 b. For MAC, the length of key is 128 b. For KDF, the length of key input and output are 128 b and 256 b. For

TABLE V
RUNTIME OF CRYPTOGRAPHIC OPERATIONS ON RPI3B

Symbol	Runtime on RPI3B (ms)			
	600Mhz	800Mhz	1Ghz	1.2Ghz
T_m^r	0.027	0.020	0.016	0.014
T_k^r	0.006	0.005	0.003	0.003
T_h^r	0.012	0.009	0.007	0.006
T_a^r	0.031	0.022	0.018	0.013
T_t^r	0.009	0.007	0.005	0.004
T_c^r	7.313	5.730	4.700	4.142
T_l^r	0.073	0.055	0.044	0.038
T_{ea}^r	0.091	0.068	0.054	0.045
T_{em}^r	19.105	14.279	11.468	9.575
T_{re}^r	65.002	48.537	38.773	32.457

TABLE VI
RUNTIME OF CRYPTOGRAPHIC OPERATIONS ON STM32

Symbol	Runtime on STM32 (ms)					
	80Mhz	160Mhz	240Mhz	320Mhz	400Mhz	480Mhz
T_m^s	0.50	0.25	0.17	0.13	0.10	0.08
T_k^s	0.10	0.05	0.03	0.03	0.02	0.02
T_h^s	0.81	0.40	0.27	0.20	0.16	0.13
T_a^s	0.52	0.26	0.17	0.13	0.10	0.09
T_t^s	0.20	0.10	0.07	0.049	0.04	0.03
T_c^s	0.63	0.32	0.21	0.16	0.13	0.11
T_l^s	37.64	18.81	12.56	9.40	7.52	6.27
T_{ea}^s	0.97	0.48	0.32	0.24	0.19	0.16
T_{em}^s	201.23	100.61	67.06	50.30	40.23	33.54
T_{re}^s	637.84	318.84	212.54	159.40	127.51	106.26

Chebyshev chaotic mapping, the length of S, x, p are 128 b. For Lagrange interpolation theorem, the length of prime number is 128 b. For Elliptic curve cryptography, the length of private and public key are 256 b and 512 b. For RSA, the length of key and plaintext is 1024 b and 512 b. The length of timestamp is 32 b. Symbols and definitions are shown in Table IV.

By adjusting the clock frequencies of RPI3B and STM32H743IIT6 to 600 Mhz–1.2 Ghz and 80 Mhz–480 Mhz respectively, we measured the runtime of cryptography operations as shown in Tables V and VI. In the table, to distinguish whether the execution platform is RPI3B or STM32H743IIT6, we mark the RPI3B-based operation as "r" and the STM32-based operation as "s". For facilitating the analysis of computational overhead, elliptic curve encryption, elliptic curve decryption, Schnorr signature, Schnorr signature verification, RSA-1024 signature, and RSA-1024 signature verification are modeled as $T_{ee} = 2T_{em} + T_{ea}$, $T_{ed} = T_{em} + T_{ea}$, $T_{ss} = T_{em} + T_h$, $T_{sv} = 2T_{em} + T_{ea} + T_h$, and $T_{rs} = T_h + T_{re}$.

C. Computation Overhead Analysis

The computational overhead of the proposed scheme is compared with the related works [29], [36], [43], [44], which are denoted as EAS, TSEC, SLDC, and WGKP. The phases compared involve ECU authentication, group key agreement, secure communication, member join, and member leave phases (denoted as EAUT, KAGA, SCOM, JOIN, and LEAV respectively). Assuming that there are n ECUs and one DC in a CAN-FD subnet. For all schemes, EAUT and KAGA compare the computational costs incurred by all nodes in the subnet.

To obtain a realistic ECU send-receive relationship, we downloaded OpenPilot [75], an open-source advanced driver

TABLE VII
COMPARISON OF COMPUTATIONAL COSTS

Step	Role	Scheme				
		[36]	[44]	[29]	[43]	Proposed
EAUT	DC/Gateway	$2T^r_{rs}+T^r_{re}$ $+T^r_a$	$T^r_{em}+T^r_{ea}$ $+T^r_h$	N/A	N/A	$2T^r_c+2T^r_m$ $+T^r_a$
	ECU_i	$2T^s_{rs}+T^s_{re}$ $+T^s_a$	$3T^s_{em}+T^s_{ea}$ $+T^s_h$	N/A	N/A	$2T^s_c+2T^s_m$ $+T^s_a$
	Total (ms) (n ECUs)	$416.526n$ $=12495.78$	$110.536n$ $=3316.08$	N/A	N/A	$8.795n=263.85$
KAGA	DC/Gateway	$2T^r_a$	–	$2T^r_m$	$(n_i+1)T^r_a$	–
	ECU_i	$2T^s_a$	$i=1, T^s_h$; $1<i<n, (n-i)(T^s_{ea}+T^s_{em})$ $+(n-i+1)T^s_k$ $+(2n-2i+2)T^s_{em}$; $i=n, 2T^s_{em}+T^s_k$	$2T^s_m+T^s_k$	First receiver: $2T^s_a$ Other receiver: T^s_a	$GL_i: T^s_m+T^s_a$ $GM^j_i: T^s_a$
	Total (ms) (n ECUs)	$0.155(n^2-n)$ $=134.415$	$33.695n^2-33.985n$ $+33.83=29339.78$	$0.208n=6.24$	$0.103n(n_i+1)=$ $0.515n=15.45$	$(0.09n_i+0.17)n$ $=0.53n=15.9$
SCOM	Sender	$T^s_a+T^s_m$	N/A	$T^s_a+T^s_m$	T^s_a	T^s_t
	Receiver	$T^s_a+T^s_m$	N/A	$T^s_a+T^s_m$	T^s_a	T^s_t
	Total (ms)	$2T^s_a+2T^s_m=0.34$	N/A	$2T^s_a+2T^s_m=0.34$	$2T^s_a=0.18$	$2T^s_t=0.06$
JOIN	DC	N/A	–	N/A	N/A	$2T^r_c+3T^r_m$ $+(n_i+4)T^r_a$
	GL	N/A	$2T^s_{em}+2T^s_{rs}$	N/A	N/A	$3T^s_a+T^s_m$
	New ECU	N/A	$2T^s_{em}+2T^s_{rs}$	N/A	N/A	$2T^s_c+2T^s_m$ $+4T^s_a+T^s_l$
	Existing member	N/A	T^s_{em}	N/A	N/A	$3T^s_a+T^s_l$
	Total (ms)	N/A	$33.54n+526.18=1532.38$	N/A	N/A	$6.553n_i+15.738$ $=41.950$
LEAV	DC	N/A	–	N/A	N/A	$(n_i+2)T^r_a$
	GL	N/A	–	N/A	N/A	$2T^s_a$
	Existing member	N/A	$i=1: 10T^s_{em}+5T^s_{ea}$; $2\le i\le L-1: 11T^s_{em}+6T^s_{ea}$; $i=L: T^s_{em}+T^s_{ea}$	N/A	N/A	$2T^s_a+T^s_l$
	Total (ms)	N/A	$36.99\cdot 5^L\cdot L^2-38.675\cdot 5^L$ $+33.7n+344.625=38135$	N/A	N/A	$6.463n_i-6.244$ $=19.608$

assistance system released by Comma Corporation. The project supplies automatic lane centering and adaptive cruise control functions for more than 250 supported vehicle brands and models, including DBC files for Acura, Chrysler, and others. Using Python, cantools [76] and Vector CANdb++ Editor [77], we analyzed the sending and receiving relationships of ECUs in DBC files based on the Volkswagen Group's Modularer Querbaukasten (MQB) platform and obtained the average number of receivers for all ECUs is 3.73. It should be noted that ECUs that do not send or receive any messages are excluded from our consideration. For ease of computation, we set the number of receivers per ECU to 4 $n_i=4$. Moreover, the ISO 11898 standard specifies that the maximum number of nodes in a subnet is 30 [78]. Therefore, we set $n=30$. The clock frequency of RPI3B and STM32H743IIT6 are fixed to be 1.2 GHz and 480 Mhz, respectively.

The computational overhead of EAUT, KAGA, SCOM, JOIN, and LEAV is shown in Table VII. Fig. 9 shows the comparison between EAUT and SCOM stages at different main frequencies. Based on the above analysis, it can be seen that for the EAUT stage, the computational cost of the proposed scheme is lower than [36] and [44] 97.89% and 92.04%, respectively. For the KAGA stage, the computational cost of the proposed scheme is lower than that of [29], [36], [43], [44] 88.17%, 99.95%, −154.81%, and −2.91%, respectively. Although the proposed scheme incurs higher computational costs during the KAGA stage compared to [29] and [43], these two schemes require a centralized controller to distribute keys, making them difficult

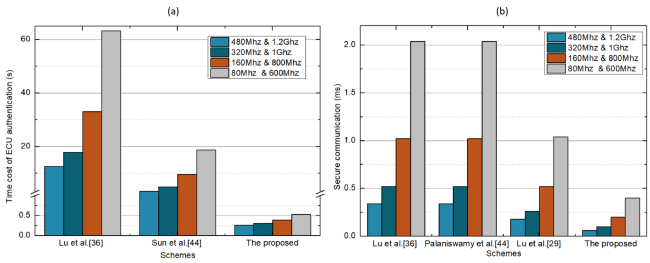


Fig. 9. Computational overhead comparison.

to handle single point of failure. For the SCOM stage, the computational cost of the proposed scheme is lower than that of [29], [36], and [43] 82.35%, 82.35%, and 66.67%, respectively. For the JOIN stage, the computational cost of the proposed scheme is lower than that of [44] 97.26%. For the LEAV stage, the computational cost of the proposed scheme is lower than that of [44] 99.95%.

D. Evaluation of Communication Overhead

In this subsection, we analyze the communication overhead. According to the STM32H7 reference manual [79], the maximum frame format for a CAN-FD extended frame is 583 b (1-bit SOF field, 30-bit arbitration field, 9-bit control field, 512-bit data field, 22-bit CRC field, 2-bit ACK field, and 7-bit EOF field). Moreover, CAN-FD has the nature of variable bus baud rate. It means that the rate of the arbitration phase (ID and ACK

TABLE VIII
THE TRANSMISSION TIME OF A FRAME UNDER DIFFERENT BAUD RATE COMBINATIONS

	Arbitration phase	Data phase	Time (ms)
Baudrate1	125Kbps	125Kbps	4.664
Baudrate2	500Kbps	500Kbps	1.166
Baudrate3	1Mbps	1Mbps	0.583
Baudrate4	1Mbps	2Mbps	0.3135
Baudrate5	1Mbps	5Mbps	0.1518

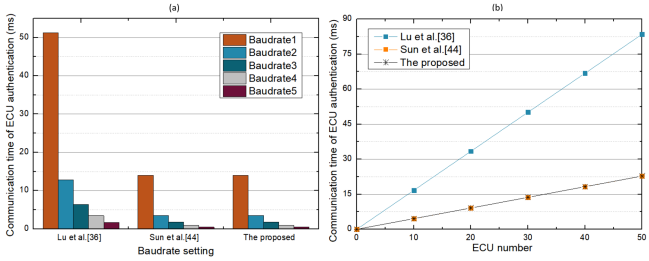


Fig. 10. Communication time during the EAUT phase.

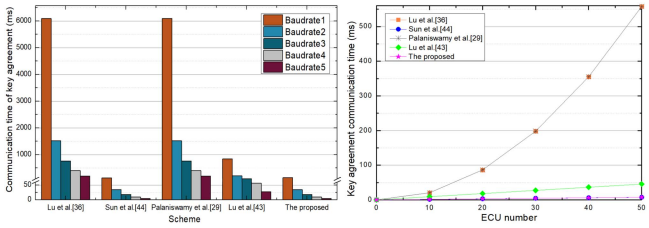


Fig. 11. Communication time for the key agreement phase.

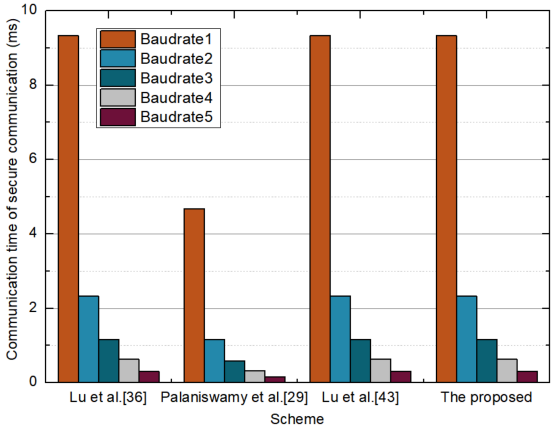


Fig. 12. Communication time for the SCOM phase.

fields) can be up to 1 Mbps, while the data phase can be up to 8 Mbps. For CAN-FD data frames, there are two arbitration phases, where the first arbitration phase contains 35 b, the second arbitration phase contains 9 b and the data phase contains 539 b. As a consequence, we set the CAN-FD bus baud rate from 125 Kbps to 5 Mbps and calculate the corresponding data transmission time at different baud rates, as shown in Table VIII.

The communication overhead of schemes is shown in Figs. 10, 11, and 12. The comparison of communication costs is shown in Table IX. To summarize, in the EAUT stage, the

TABLE IX
COMPARISON OF COMMUNICATION COSTS

Phase	Scheme	Message number	Message length (byte)	Frame number
EAUT	[36]	$3n = 90$	$531n = 15930$	$11n = 330$
	[44]	$3n = 90$	$129n = 3870$	$3n = 90$
	[29]	–	–	–
	[43]	–	–	–
	Ours	$3n = 90$	$146n = 4380$	$3n = 90$
KAGA	[36]	$\frac{3n(n-1)}{2} = 1305$	$\frac{49n(n-1)}{2} = 21315$	$\frac{3n(n-1)}{2} = 1305$
	[44]	$n = 30$	$64n = 1920$	$n = 30$
	[29]	$\frac{3n(n-1)}{2} = 1305$	$\frac{96n(n-1)}{2} = 41760$	$\frac{3n(n-1)}{2} = 1305$
	[43]	$3n = 90$	$101n = 3030$	$(n_i + 2)n = 180$
	Ours	$n = 30$	$22n = 660$	$n = 30$
SCOM	Original message	1	64	1
	[36]	1	102	2
	[44]	–	–	–
	[29]	1	68	2
	[43]	1	86	2
	Ours	1	72	2

communication overhead of the proposed scheme is lower than that of [36] and [44] 75.52% and -0.78%, only slightly lower than [44]. For the KAGA stage, the computational cost of the proposed scheme is lower than that of [29], [36], [44], and [43] 96.90%, 65.63%, 98.42%, and 78.22%, respectively. For the SCOM stage, the computational cost of the proposed scheme is lower than that of [29], [36], and [43] 29.41%, −5.88%, and 16.28%, respectively. Although the communication overhead of the proposed scheme is slightly higher than that of [29] at the SCOM stage, the computational overhead is lower. Therefore, the proposed scheme has practicality.

E. Evaluation of Bus Load

We adopt CANoe software [80] and Communication Access Programming Language (CAPL) to simulate the bus load of the proposed scheme. For the in-vehicle network, bus load is an important factor affecting the stability of signal transmission. It is mentioned in [15] that the load of most high-speed CAN buses in the vehicle is only about 30%–33%. It is rare to use a utilization rate of over 60% to offset the uncertainty in extreme situations.

We built a baseline network simulation environment with four virtual ECUs. The network structure consists of a domain controller, a group leader, and two group members. The baud rate is set to Baudrate1-Baudrate5 as mentioned above. The bus load of the baseline network is maintained near 30% by adjusting ECUs to send messages. Based on the baseline network, we simulated the data flow of EAUT, KAGA, and SCOM phases, and tested the bus load.

For EAUT phase, based on the baseline network, DC authenticates GL, GM^1, and GM^2. Three messages need to be sent are set to be 64, 64, and 48 bytes (the length of messages in the CANdb++ editor can only be set to a fixed length [81]). For KAGA, $BReq$ and $Cipher_2$ are set to be 48 and 24bytes. During SCOM, for each message (64 bytes) sent in the baseline network, a message of 1 B is appended as the authentication tag. The bus

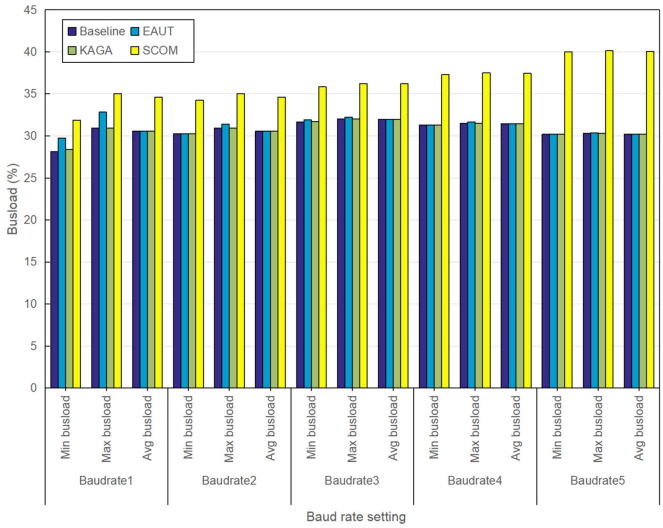


Fig. 13. Bus load analysis at each phase.

TABLE X
ANALYSIS OF MEMORY OVERHEAD

Role	Stage	Memory overhead
DC	EAUT	852KB
	KAGA	—
	Total	852KB
GL	EAUT	RW-data:1024; ZI-data:462712; Total:463736 Bytes
	KAGA	RW-data:1024; ZI-data:462712; Total:463736 Bytes
	SCOM	RW-data:36; ZI-data:457180; Total:457216 Bytes
	Total	RW-data=1024; ZI-data=462712; Total:452 KB
GM	EAUT	RW-data:1024; ZI-data:462712; Total:463736 Bytes
	KAGA	RW-data:1020; ZI-data:462708; Total:463728 Bytes
	SCOM	RW-data:36; ZI-data:457180; Total:457216 Bytes
	Total	RW-data=1024; ZI-data=462712; Total:452 KB

load for the three phases is shown in Fig. 13. It can be seen that the busload during the EAUT, KAGA, and SCOM phases can increase by up to 1.88%, 0.25%, and 9.84%, respectively compared to the baseline. And the busload in the SCOM phase is higher compared to the baseline, with a range of 3.72% to 9.84%, and the range of the increase increases as the baud rate rises. However, the maximum bus load in this phase is 40.07%, which is still far below the upper limit of 60%.

F. Memory Overhead Analysis

On RPi3B (1 GB SDRAM) and STM32 (1060 KB SRAM), we implemented the EAUT, KAGA, and SCOM stages respectively and measured the memory overhead. Memory overhead refers to the memory space occupied by program data during the execution of security algorithms. On STM32, memory overhead consists of RW-data (Read-Write Data) and ZI-data (Zero Initialized Data). The memory consumption of different roles across phases is shown in Table X. As revealed in the table, the memory overhead generated by security algorithms on RPI3B is only 852 KB, accounting for 0.08% of RAM. On the STM32 platform, the memory overhead is 452 KB, occupying 46.2% of SRAM. Notably, if using ST's official lightweight cryptographic library X-CUBE-CRYPTOLIB (optimized with streamlined data structures and strict memory management), the memory footprint of security algorithms on STM32 could be further reduced. In contrast, MIRACL, as a general-purpose big number/cryptographic library supporting multiple platforms and algorithms, incurs higher memory overhead on embedded platforms due to its more complex data structures and generality-oriented design.

Furthermore, on STM32, the memory overheads of EAUT and KAGA phases are remarkably similar. This stems from their shared dependency on MIRACL's big number library data structures (e.g., big and miracl objects, memory pools). During linking, both phases automatically incorporate all MIRACL-related global variables and auxiliary structures into the final executable. Since MIRACL's big number structures and global auxiliary objects are statically allocated and their memory space is determined at the linking stage, the actual RW-data and ZI-data consumption shows minimal difference between EAUT and KAGA phases. The SCOM phase does not involve the MIRACL library, resulting in significantly lower memory usage. When integrating all three phases simultaneously, the RW-data and ZI-data does not simply accumulate the total memory of each phase. Instead, the linker optimizes memory allocation by reusing static variables with identical names or purposes, allocating only the maximum required space for shared components. This static memory allocation strategy ensures optimal memory utilization for the entire system.

G. Robustness Analysis

This section evaluates the robustness of single point fault scenarios in domain controllers. From the analysis in Section VI-I-C (KAGA row of Table VII), it can be seen that [29], [36], and [43] require DC participation in the key agreement stage, thus lacking robustness for DC single point failure scenarios. The proposed scheme and [44] do not require DC participation and have robustness. This is because in the architecture design of this study, DC is only responsible for real-time authentication (which needs to be online) and non-real-time secret share distribution (which can be offline or completed in advance). This stage is entirely independently completed by the ECUs within the group, without the need for DC involvement.

VIII. CONCLUSION

In this paper, we design an efficient key agreement and secure communication mechanism for the CAN-FD network to address the problem that existing security frameworks rely on centralized gateways. The proposed scheme groups ECUs according to their send-receive relationships at the time of manufacturing to avoid destroying the intrinsic communication mechanism of the CAN-FD bus. The security analysis shows that the scheme proposed in this paper is resistant to various typical attacks. Performance analysis shows that the proposed scheme performs well in terms of computation overhead and communication overhead.

Although the proposed scheme enhances the security of the CAN-FD network, there are still limitations, such as complex entity interactions, dependence on DC in some stages, and increased bus load, which makes real-world applications difficult.

Therefore, in the future, we plan to design a fully decentralized, practical, and efficient security mechanism for in-vehicle networks.

ACKNOWLEDGMENT

The authors are very grateful to the anonymous referees for their detailed comments and suggestions regarding this paper.

REFERENCES

[1] J. Zhang, S. Su, H. Zhong, J. Cui, and D. He, "Identity-based broadcast proxy re-encryption for flexible data sharing in VANETs," *IEEE Trans. Inf. Forensics Security*, vol. 18, pp. 4830–4842, 2023.

[2] L. Wei, J. Cui, H. Zhong, I. Bolodurina, C. Gu, and D. He, "A decentralized authenticated key agreement scheme based on smart contract for securing vehicular ad-hoc networks," *IEEE Trans. Mobile Comput.*, vol. 23, no. 5, pp. 4318–4333, May 2024.

[3] C. Chen, W. Chenyu, C. Li, X. Ming, and P. Qingqi, "A V2V emergent message dissemination scheme for 6G-oriented vehicular networks," *Chin. J. Electron.*, vol. 32, no. 6, pp. 1179–1191, 2023.

[4] Y. Li, J. Zhao, J. Liao, and F. Hu, "Cellular V2X-based integrated sensing and communication system: Feasibility and performance analysis," *Chin. J. Electron.*, vol. 33, no. 4, pp. 1104–1116, 2024.

[5] C. Wu et al., "TokenScout: Early detection of ethereum scam tokens via temporal graph learning," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, New York, NY, USA, 2024, pp. 956–970, doi: 10.1145/3658644.3690234.

[6] C. Wu et al., "Rethinking membership inference attacks against transfer learning," *IEEE Trans. Inf. Forensics Security*, vol. 19, pp. 6441–6454, 2024.

[7] C. Wu, J. Chen, Z. Wang, R. Liang, and R. Du, "Semantic sleuth: Identifying ponzi contracts via large language models," in *Proc. 39th IEEE/ACM Int. Conf. Autom. Softw. Eng.*, New York, NY, USA, 2024, pp. 582–593, doi: 10.1145/3691620.3695055.

[8] L. Popa, B. Groza, C. Jichici, and P.-S. Murvay, "ECUPrint-physical fingerprinting electronic control units on CAN buses inside cars and SAE J1939 compliant vehicles," *IEEE Trans. Inf. Forensics Security*, vol. 17, pp. 1185–1200, 2022.

[9] K.-T. Cho and K. G. Shin, "Fingerprinting electronic control units for vehicle intrusion detection," in *Proc. 25th USENIX Secur. Symp.*, Austin, TX: USENIX Association, 2016, pp. 911–927.

[10] H. Stoll, E. Koch, and E. Sax, "Integration of ROS communication interfaces in a model-based tool for the description of AUTOSAR-compliant electrical/electronic architectures (E/E-A) in vehicle development," in *Proc. IEEE 23rd Int. Conf. Intell. Transp. Syst.*, 2020, pp. 1–6.

[11] Y. Peng, B. Shi, T. Jiang, X. Tu, D. Xu, and K. Hua, "A survey on in-vehicle time-sensitive networking," *IEEE Internet Things J.*, vol. 10, no. 16, pp. 14375–14396, Aug. 2023.

[12] L. Xiao, X. Lu, T. Xu, W. Zhuang, and H. Dai, "Reinforcement learning-based physical-layer authentication for controller area networks," *IEEE Trans. Inf. Forensics Security*, vol. 16, pp. 2535–2547, 2021.

[13] H. J. Jo and W. Choi, "A survey of attacks on controller area networks and corresponding countermeasures," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 7, pp. 6123–6141, Jul. 2022.

[14] H. Zhang, K. Zeng, and S. Lin, "Federated graph neural network for fast anomaly detection in controller area networks," *IEEE Trans. Inf. Forensics Security*, vol. 18, pp. 1566–1579, 2023.

[15] W. Zeng, M. A. S. Khalid, and S. Chowdhury, "In-vehicle networks outlook: Achievements and challenges," *IEEE Commun. Surveys Tuts.*, vol. 18, no. 3, pp. 1552–1571, Third Quarter 2016.

[16] X. Duan, H. Yan, D. Tian, J. Zhou, J. Su, and W. Hao, "In-vehicle CAN bus tampering attacks detection for connected and autonomous vehicles using an improved isolation forest method," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 2, pp. 2122–2134, Feb. 2023.

[17] Y. Xie, G. Zeng, R. Kurachi, F. Xiao, and H. Takada, "Optimizing extensibility of CAN FD for automotive cyber-physical systems," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 12, pp. 7875–7886, Dec. 2021.

[18] G. Xie, L. T. Yang, W. Wu, K. Zeng, X. Xiao, and R. Li, "Security enhancement for real-time parallel in-vehicle applications by CAN FD message authentication," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 8, pp. 5038–5049, Aug. 2021.

[19] G. Xie, R. Li, and S. Hu, "Security-aware obfuscated priority assignment for CAN FD messages in real-time parallel automotive applications," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 39, no. 12, pp. 4413–4425, Dec. 2020.

[20] W. Ma, G. Xie, R. Li, and W. Chang, "Optimality-guaranteed design space pruning for CAN-FD frame packing," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 43, no. 1, pp. 44–56, Jan. 2024.

[21] B. Lampe and W. Meng, "IDS for CAN: A practical intrusion detection system for CAN bus security," in *Proc. IEEE Glob. Commun. Conf.*, 2022, pp. 1782–1787.

[22] P. Biondi, G. Bella, G. Costantino, and I. Matteucci, "Implementing CAN bus security by toucan," in *Proc. 20th ACM Int. Symp. Mobile Ad Hoc Netw. Comput.*, New York, NY, USA, 2019, pp. 399–400.

[23] Y. Xie, G. Zeng, R. Kurachi, F. Xiao, H. Takada, and S. Hu, "Timing analysis of CAN FD for security-aware automotive cyber-physical systems," *IEEE Trans. Dependable Secure Comput.*, vol. 20, no. 4, pp. 3064–3078, Jul./Aug. 2023.

[24] J. Tang, S. Shao, J. Song, and A. Gupta, "Nash equilibrium control policy against bus-off attacks in CAN networks," *IEEE Trans. Inf. Forensics Security*, vol. 18, pp. 980–990, 2023.

[25] M. Jedh, L. Ben Othmane, N. Ahmed, and B. Bhargava, "Detection of message injection attacks onto the CAN bus using similarities of successive messages-sequence graphs," *IEEE Trans. Inf. Forensics Security*, vol. 16, pp. 4133–4146, 2021.

[26] 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.

[27] K. Koscher et al., "Experimental security analysis of a modern automobile," in *Proc. 2010 IEEE Symp. Secur. Privacy*, 2010, pp. 447–462.

[28] C. Wu, J. Chen, K. He, Z. Zhao, R. Du, and C. Zhang, "EchoHand: High accuracy and presentation attack resistant hand authentication on commodity mobile devices," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, New York, NY, USA, 2022, pp. 2931–2945, doi: 10.1145/3548606.3560553.

[29] B. Palaniswamy, S. Camtepe, E. Foo, and J. Pieprzyk, "An efficient authentication scheme for intra-vehicular controller area network," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 3107–3122, 2020.

[30] I. E. Carvajal-Roca, J. Wang, J. Du, and S. Wei, "A semi-centralized dynamic key management framework for in-vehicle networks," *IEEE Trans. Veh. Technol.*, vol. 70, no. 10, pp. 10864–10879, Oct. 2021.

[31] D. Yu, R.-H. Hsu, J. Lee, and S. Lee, "EC-SVC: Secure CAN bus in-vehicle communications with fine-grained access control based on edge computing," *IEEE Trans. Inf. Forensics Security*, vol. 17, pp. 1388–1403, 2022.

[32] J. Cui, Y. Shen, H. Zhong, J. Zhang, and L. Liu, "A multilevel electronic control unit re-encryption scheme for autonomous vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 25, no. 1, pp. 104–119, Jan. 2024.

[33] R. de Andrade, M. M. D. Santos, J. F. Justo, L. R. Yoshioka, H.-J. Hof, and J. H. Kleinschmidt, "Security architecture for automotive communication networks with CAN FD," *Comput. Secur.*, vol. 129, 2023, Art. no. 103203.

[34] A. Gonzalez Mariño, F. Fons, and J. M. Moreno Arostegui, "Elastic gateway SoC proof of concept: Experiments design and performance evaluation," *Veh. Commun.*, vol. 43, 2023, Art. no. 100636.

[35] A. Buscemi, I. Turcanu, G. Castignani, R. Crunelle, and T. Engel, "Canmatch: A fully automated tool for CAN bus reverse engineering based on frame matching," *IEEE Trans. Veh. Technol.*, vol. 70, no. 12, pp. 12358–12373, Dec. 2021.

[36] R. Lu, G. Xie, R. Li, W. Xu, and J. Lei, "TrinitySec: Trinity-enabled and lightweight security framework for CAN-FD communication," *IEEE Trans. Dependable Secure Comput.*, vol. 21, no. 4, pp. 2704–2719, Jul./Aug. 2024.

[37] J. Cui et al., "Lightweight encryption and authentication for controller area network of autonomous vehicles," *IEEE Trans. Veh. Technol.*, vol. 72, no. 11, pp. 14756–14770, Nov. 2023.

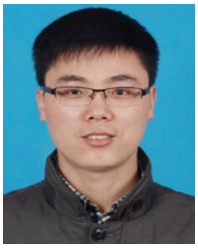
[38] G. Xie, L. T. Yang, Y. Liu, H. Luo, X. Peng, and R. Li, "Security enhancement for real-time independent in-vehicle CAN-FD messages in vehicular networks," *IEEE Trans. Veh. Technol.*, vol. 70, no. 6, pp. 5244–5253, Jun. 2021.

[39] F. Oberti, A. Savino, E. Sanchez, F. Parisi, and S. Di Carlo, "EXTTAURUM P2T: An extended secure CAN-FD architecture for road vehicles," *IEEE Trans. Device Mater. Rel.*, vol. 22, no. 2, pp. 98–110, Jun. 2022.

[40] X. Ying, G. Bernieri, M. Conti, L. Bushnell, and R. Poovendran, "Covert channel-based transmitter authentication in controller area networks," *IEEE Trans. Dependable Secure Comput.*, vol. 19, no. 4, pp. 2665–2679, Jul./Aug. 2022.

[41] Y. Shen, J. Cui, H. Zhong, J. Zhang, I. Bolodurina, and D. He, "A two-layer dynamic ECU group management scheme for in-vehicle CAN bus," *IEEE Trans. Intell. Transp. Syst.*, vol. 25, no. 8, pp. 10431–10445, Aug. 2024.
[42] A. Musuroi, B. Groza, L. Popa, and P.-S. Murvay, "Fast and efficient group key exchange in controller area networks (CAN)," *IEEE Trans. Veh. Technol.*, vol. 70, no. 9, pp. 9385–9399, Sep. 2021.
[43] R. Lu et al., "Secure and low-delay CAN-FD communication in embedded microcontroller: A cooperative swapping approach," *IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst.*, vol. 43, no. 8, pp. 2312–2325, Aug. 2024.
[44] H. Sun, W. Luo, J. Weng, Z. Liu, and M. Li, "ECQV-GDH-based group key exchange protocol for CAN bus," *IEEE Trans. Veh. Technol.*, vol. 72, no. 10, pp. 12857–12872, Oct. 2023.
[45] D. Püllen, N. A. Anagnostopoulos, T. Arul, and S. Katzenbeisser, "Using implicit certification to efficiently establish authenticated group keys for in-vehicle networks," in *Proc. 2019 IEEE Veh. Netw. Conf.*, 2019, pp. 1–8.
[46] W. Choi, S. Lee, K. Joo, H. J. Jo, and D. H. Lee, "An enhanced method for reverse engineering CAN data payload," *IEEE Trans. Veh. Technol.*, vol. 70, no. 4, pp. 3371–3381, Apr. 2021.
[47] M. E. Verma, R. A. Bridges, J. J. Sosnowski, S. C. Hollifield, and M. D. Iannacone, "CAN-D: A modular four-step pipeline for comprehensively decoding controller area network data," *IEEE Trans. Veh. Technol.*, vol. 70, no. 10, pp. 9685–9700, Oct. 2021.
[48] A. Dos Santos Roque, L. M. Da Silva Alves, and E. P. de Freitas, "CAN-modes: In-vehicle datasets generation and analysis in different driving situations," in *Proc. Workshop Commun. Netw. Power Syst.*, 2024, pp. 1–7.
[49] M. D. Pesé, J. W. Schauer, J. Li, and K. G. Shin, "S2-CAN: Sufficiently secure controller area network," in *Proc. Annu. Comput. Secur. Appl. Conf.*, New York, NY, USA, 2021, pp. 425–438, doi: 10.1145/3485832.3485883.
[50] W. L. Lambert, S. Ghafoor, H. Burnell, B. Huber, F. Kandah, and A. Skjellum, "Design and development of XiveNet: A hybrid CAN research testbed," in *Proc. 23rd Int. Symp. Parallel Distrib. Comput.*, 2024, pp. 1–8.
[51] G. Bella, P. Biondi, G. Costantino, and I. Matteucci, "Designing and implementing an AUTOSAR-based basic software module for enhanced security," *Comput. Netw.*, vol. 218, 2022, Art. no. 109377. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S138912862200411X
[52] M. A. El-Gleel, A. M. O. El-Zawawi, and M. El-Habrouk, "Secure lightweight CAN protocol handling message loss for electric vehicles," in *Proc. Int. Conf. Commun. Control Inf. Sci.*, 2021, pp. 1–6.
[53] M. Roeschlin, G. Camurati, P. Brunner, S. Mridula, and C. Srdjan, "EdgeTDC: On the security of time difference of arrival measurements in CAN bus systems," in *Proc. Netw. Distrib. Syst. Secur. Symp.*, Mar. 2023.
[54] W. Ma, G. Xie, R. Li, W. Liu, H. H. Li, and W. Chang, "Efficient AUTOSAR-compliant CAN-FD frame packing with observed optimality," in *Proc. Des. Autom. Test Europe Conf. Exhib.*, 2021, pp. 1899–1904.
[55] M. A. Maruf and A. Azim, "Anomaly detection and functional testing for automotive CAN communication," in *Proc. IEEE Int. Syst. Conf.*, 2024, pp. 1–8.
[56] A. Shamir, "How to share a secret," *Commun. ACM*, vol. 22, no. 11, pp. 612–613, Nov. 1979.
[57] G. R. Blakley, "Safeguarding cryptographic keys," in *Proc. Int. Workshop Manag. Requirements Knowl.*, Los Alamitos, CA, USA, 1979, pp. 313–318.
[58] Z. Zhang et al., "TAGKA: Threshold authenticated group key agreement protocol against member disconnect for UANET," *IEEE Trans. Veh. Technol.*, vol. 72, no. 11, pp. 14987–15001, Nov. 2023.
[59] S. Qiu, D. Wang, G. Xu, and S. Kumari, "Practical and provably secure three-factor authentication protocol based on extended chaotic-maps for mobile lightweight devices," *IEEE Trans. Dependable Secure Comput.*, vol. 19, no. 2, pp. 1338–1351, Mar./Apr. 2022.
[60] L. Zhang, "Cryptanalysis of the public key encryption based on multiple chaotic systems," *Chaos Solitons Fractals*, vol. 37, no. 3, pp. 669–674, 2008.
[61] Infineon, "Hardware security module," 2020. Accessed: Aug. 13, 2020. [Online]. Available: https://www.infineon.com/dgdl/Infineon-AURIX_TC3xx_Hardware_Security_Module_Quick-Training-v01_00-EN.pdf?fileId=5546d46274cf54d50174da4ebc3f2265
[62] H. Wu and T. Huang, "TinyJAMBU: A family of lightweight authenticated encryption algorithms," 2019. Accessed: Mar. 29, 2019. [Online]. Available: https://csrc.nist.gov/CSRC/media/Projects/Lightweight-Cryptography/documents/round-1/spec-doc/TinyJAMBU-spec.pdf
[63] M. El-Hadedy, R. Hua, S. Saqib, K. Yoshii, W.-M. Hwu, and M. Margala, "BLTESTI: Benchmarking lightweight TinyJAMBU on embedded systems for trusted IoT," in *Proc. IEEE 36th Int. Syst.-on-Chip Conf.*, 2023, pp. 1–6.
[64] A. Abdulgadir, S. Lin, F. Farahmand, J.-P. Kaps, and K. Gaj, "Side-channel resistant implementations of a novel lightweight authenticated cipher with application to hardware security," in *Proc. Great Lakes Symp. VLSI*, New York, NY, USA, 2021, pp. 229–234.
[65] P. Rosa, A. Souto, and J. Cecílio, "Light-SAE: A lightweight authentication protocol for large-scale IoT environments made with constrained devices," *IEEE Trans. Netw. Service Manag.*, vol. 20, no. 3, pp. 2428–2441, Sep. 2023.
[66] M. Held, N. Rosat, G. Georges, H. Pengg, and K. Boulouchos, "Lifespans of passenger cars in Europe: Empirical modelling of fleet turnover dynamics," *Eur. Transport Res. Rev.*, vol. 13, pp. 1–13, 2021.
[67] D. Dolev and A. Yao, "On the security of public key protocols," *IEEE Trans. Inf. Theory*, vol. 29, no. 2, pp. 198–208, Mar. 1983.
[68] I. T. AG, "HSM - Hardware security module," 2019. Accessed: Nov. 12, 2019. (n.d.) [Online]. Available: https://www.infineon.com/dgdl/Infineon-AURIX_Hardware_Security_Module-TR-v01_00-EN.pdf?fileId=5546d46269bda8df0169ca6e34c62549&intc=0560030
[69] UNECE, "UN regulation no 155 - Cyber security and cyber security management system," 2021. Accessed: Apr. 03, 2021. [Online]. Available: https://unece.org/transport/documents/2021/03/standards/un-regulation-no-155-cyber-security-and-cyber-security
[70] M. Abdalla, P.-A. Fouque, and D. Pointcheval, "Password-based authenticated key exchange in the three-party setting," in *Proc. Int. Workshop Public Key Cryptography*, Berlin, Heidelberg:Springer, 2005, pp. 65–84.
[71] S. Choochotkaew, T. Chiba, S. Trent, and M. Amaral, "Bypass container overlay networks with transparent BPF-driven socket replacement," in *Proc. IEEE 15th Int. Conf. Cloud Comput.*, 2022, pp. 134–143.
[72] STMicroelectronics, "STM32CubeH7-STM32Cube MCU package for STM32H7 series with HAL and dedicated middleware," 2017. Accessed: Dec. 1, 2017. [Online]. Available: https://www.st.com/en/embedded-software/stm32cubeh7.html#documentation/
[73] MIRACL, "Miracl cryptographic sdk," 2019. Accessed: Nov. 29, 2019. [Online]. Available: https://github.com/miracl/MIRACL/
[74] L. Bruckert, J. Merkle, and M. Lochter, "RFC 8734: Elliptic curve cryptography (ECC) brainpool curves for transport layer security (TLS) version 1.3," 2020. Accessed: Feb. 2020. [Online]. Available: https://www.hjp.at/(en,st_a)/doc/rfc/rfc8734.pdf
[75] Comma, "Openpilot," 2024. Accessed: Jun. 14, 2024. [Online]. Available: https://github.com/commaai/openpilot/tree/devel#what-is-openpilot
[76] E. Moqvis, "Cantools documentation," 2024. Accessed: Jan. 15, 2024. [Online]. Available: https://cantools.readthedocs.io/_/downloads/en/latest/pdf/
[77] Vector, "CANdb admin," 2019. Accessed: Nov. 2019. [Online]. Available: https://cdn.vector.com/cms/content/products/candb/Docs/CANdb_Admin_FactSheet_EN.pdf
[78] T. INSTRUMENTS, "ISO1050 isolated CAN transceiver," 2023. Accessed: Oct. 2023. [Online]. Available: https://www.ti.com/lit/ds/symlink/iso1050.pdf
[79] STMicroelectronics, "RM0433 reference manual," 2023. Accessed: Jan. 19, 2023. [Online]. Available: https://www.st.com.cn/resource/en/reference_manual/rm0433-stm32h742-stm32h743753-and-stm32h750-value-line-advanced-armbased-32bit-mcus-stmicroelectronics.pdf
[80] Vector, "Product information CANoe," 2023. Accessed: Sep. 22, 2023. (n.d.) [Online]. Available: https://cdn.vector.com/cms/content/products/canoe/canoe/docs/Product%20Informations/CANoe_ProductInformation_EN.pdf
[81] VECTOR, "Length of the data field," 2021. Accessed: Sep. 22, 2021. [Online]. Available: https://elearning.vector.com/mod/page/view.php?id=368

Yun Shen is now working toward the PhD degree with the School of Computer Science and Technology, Anhui University. Her research focuses on in-vehicle network security, CAN bus security, and ECU authentication.



Jie Cui (Senior Member, IEEE) received the PhD degree from the University of Science and Technology of China, in 2012. He is currently a professor and PhD supervisor of the School of Computer Science and Technology, Anhui University. His current research interests include applied cryptography, IoT security, vehicular ad hoc network, cloud computing security and software-defined networking (SDN). He has more than 150 scientific publications in reputable journals (e.g., *IEEE Transactions on Dependable and Secure Computing*, *IEEE Transactions on Information Forensics and Security*, *IEEE Journal on Selected Areas in Communications*, *IEEE Transactions on Mobile Computing*, *IEEE Transactions on Parallel and Distributed Systems*, *IEEE Transactions on Computers*, *IEEE Transactions on Vehicular Technology*, *IEEE Transactions on Intelligent Transportation Systems*, *IEEE Transactions on Network and Service Management*, *IEEE Transactions on Industrial Informatics*, *IEEE Transactions on Industrial Electronics*, *IEEE Transactions on Cloud Computing* and *IEEE Transactions on Multimedia*), academic books and international conferences.

Hong Zhong received the PhD degree in computer science from the University of Science and Technology of China, in 2005. She is currently a professor and PhD supervisor of the School of Computer Science and Technology, Anhui University. Her research interests include applied cryptography, IoT security, vehicular ad hoc network, cloud computing security and software-defined networking (SDN). She has more than 200 scientific publications in reputable journals (e.g., *IEEE Journal on Selected Areas in Communications*, *IEEE Transactions on Parallel and Distributed Systems*, *IEEE Transactions on Mobile Computing*, *IEEE Transactions on Dependable and Secure Computing*, *IEEE Transactions on Information Forensics and Security*, *IEEE Transactions on Intelligent Transportation Systems*, *IEEE Transactions on Multimedia*, *IEEE Transactions on Vehicular Technology*, *IEEE Transactions on Network and Service Management*, *IEEE Transactions on Cloud Computing*, *IEEE Transactions on Industrial Informatics*, *IEEE Transactions on Industrial Electronics* and *IEEE Transactions on Big Data*), academic books and international conferences.

Jing Zhang received the MA Eng and PhD degrees in computer science from Anhui University, in 2021. She is currently an associate professor of the School of Computer Science and Technology, Anhui University. Her research interests include vehicular ad hoc network, IoT security and applied cryptography. She has nearly 20 scientific publications in reputable journals (e.g., *IEEE Transactions on Dependable and Secure Computing*, *IEEE Transactions on Information Forensics and Security*, *IEEE Journal on Selected Areas in Communications*, *IEEE Transactions on Vehicular Technology*, *IEEE Transactions on Intelligent Transportation Systems*, *Information Sciences*, *Science China Information Sciences* and *Vehicular Communications*) and international conferences.

Qingyang Zhang received the PhD degree in computer science from Anhui University, in 2022. He is currently a lecture of the School of Computer Science and Technology, Anhui University. His research interests include vehicular ad hoc networks and applied cryptography. He has more than 10 scientific publications in reputable journals (e.g., *IEEE Transactions on Information Forensics and Security*, *IEEE Journal on Selected Areas in Communications*, *IEEE Transactions on Mobile Computing*, *IEEE Transactions on Intelligent Transportation Systems*).

Lu Wei (Member, IEEE) received the PhD degree in computer science from Anhui University, in 2022. He is currently a lecture of the School of Computer Science and Technology, Anhui University. His research interests include vehicular ad hoc networks and applied cryptography. He has more than 10 scientific publications in reputable journals (e.g., *IEEE Transactions on Information Forensics and Security*, *IEEE Journal on Selected Areas in Communications*, *IEEE Transactions on Mobile Computing*, *IEEE Transactions on Intelligent Transportation Systems*).

Debiao He (Member, IEEE) received the PhD degree in applied mathematics from the School of Mathematics and Statistics, Wuhan University, Wuhan, China, in 2009. He is currently a professor of the School of Cyber Science and Engineering, Wuhan University, Wuhan, China. His main research interests include cryptography and information security, in particular, cryptographic protocols. He has published more than 100 research papers in refereed international journals and conferences, such as *IEEE Transactions on Dependable and Secure Computing*, *IEEE Transactions on Information Security and Forensic*, and Usenix Security Symposium. He is the recipient of the 2018 IEEE Sysems Journal Best Paper Award and the 2019 IET Information Security Best Paper Award. His work has been cited more than 10000 times with Google Scholar. He is in the editorial board of several international journals, such as *Journal of Information Security and Applications*, *Frontiers of Computer Science*, and *Human-centric Computing & Information Sciences*.