Lightweight Encryption and Authentication for Controller Area Network of Autonomous Vehicles

Jie Cui, Yaning Chen, Hong Zhong, Debiao He, Lu Wei, Irina Bolodurina, and Lu Liu

***Abstract*—With the rapid development of autonomous vehicles (AVs), vehicle safety has attracted attention. Controller area network (CAN) bus without authentication and encryption mechanisms is a weakness of the in-vehicle network (IVN), and attackers always primarily target the CAN bus. Attacks on the AVs' CAN bus are more frequent because AVs have more external interfaces and sensors than ordinary vehicles. To address the abovementioned issues, this paper proposes a lightweight encryption and authentication scheme for the CAN bus of AVs using message authentication codes and Grain stream cipher. The scheme realizes efficient authentication between electronic control units and a counter re-synchronization protocol is used to ensure that authentication failure owing to counter inconsistency is prevented. Blom key management scheme is employed to rapidly distribute and update session keys, avoiding the complicated process of updating pairwise keys. In addition, considering the scalability of the IVN, a key distribution protocol for new nodes was provided. The security proof using Burrows-Abadi-Needham logic and security analysis prove that the proposed scheme can satisfy the security requirements of the IVN. In addition, performance analysis shows that the proposed scheme can meet real-time requirements and has little impact on the busload.**

***Index Terms*—In-vehicle network, CAN bus security, autonomous vehicles, authentication, Blom key management scheme.**

Manuscript received 26 March 2022; revised 3 January 2023 and 30 March 2023; accepted 6 May 2023. Date of publication 30 May 2023; date of current version 14 November 2023. The work was supported in part by the National Natural Science Foundation of China under Grants 62202008, and 62272002, in part by the Excellent Youth Foundation of Anhui Scientific Committee under Grant 2108085J31, in part by the Natural Science Foundation of Anhui Province, China under Grant 2208085QF196, in part by the University Synergy Innovation Program of Anhui Province under Grant GXXT-2022-049, in part by the Foundation of Anhui Educational Committee under Grant KJ2020A0037 and in part by the Open Fund of Key Laboratory of Embedded System and Service Computing (Tongji University), Ministry of Education under Grant ESSCKF2022-04. The review of this article was coordinated by Dr. Ming Li. *(Corresponding author: Hong Zhong.)*

Jie Cui, Yaning Chen, Hong Zhong, and Lu Wei are with the School of Computer Science and Technology, Anhui University, Hefei 230039, China, also with the Anhui Engineering Laboratory of IoT Security Technologies, Anhui University, Hefei 230039, China, and also with the Institute of Physical Science and Information Technology, Anhui University, Hefei 230039, China (e-mail: cuijie@mail.ustc.edu.cn; yaningchen123@outlook.com; zhongh@ahu.edu.cn; dreamer_weilu@163.com).

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

Irina Bolodurina is with the Faculty of Mathematics and Information Technologies, Orenburg State University, 460018 Orenburg, Russia (e-mail: prmat@mail.osu.ru).

Lu Liu is with the School of Informatics, University of Leicester, LE1 7RH Leicester, U.K. (e-mail: l.liu@leicester.ac.uk).

I. Introduction

In the 1970s, countries worldwide began researching autonomous vehicles (AVs). Today, several premium vehicles on the market are equipped with the advanced driver assistance system (ADAS), and fully autonomous vehicles are still in the road test stage because a few problems still exist [1]. The main issue is that AVs are vulnerable to cyber-attack. Various physical and wireless connection methods increase the attack surface [2], and adversaries can easily penetrate in-vehicle communication networks. A large amount of data is transmitted through an in-vehicle network (IVN) and affects the driving decisions of AVs [3]. These data are generated by sensors and cameras throughout the vehicles' body. Security threats arise once driving data are used by an adversary connected to the network. Therefore, if manufacturers want to deploy AVs on a wide range of real roads, secure IVN communication requires to ensure the safety of AVs.

To realize autonomous driving, the number of electronic control units (ECUs) installed in an autonomous vehicle must be greater than that of ordinary vehicles. ECUs are connected to various types of buses to transmit information. The IVN is composed of the bus, ECUs and sensors. The buses used in the IVN include the controller area network (CAN), local interconnect network (LIN), media-oriented system transport (MOST), FlexRay, and Ethernet [4]. Among them, CAN is the most widely used and enduring in-vehicle bus system. Its built-in security features are designed to ensure reliable communication without considering network security issues [5]. Messages on the CAN bus are transmitted in plaintext and there is no authentication mechanism to guarantee message integrity. An adversary can access the CAN bus and steal messages through multiple external interfaces on the AV so the CAN bus is not secure.

The main goal of designing AVs is to improve traffic safety [6], whereas attacks on IVNs increase the risk of car accidents. Recent studies have shown an increasing number of attacks on the in-vehicle bus, especially the CAN bus, and researchers have implemented some of the attack methods in an actual in-vehicle environment. After attackers enter the IVN through the external interface, they can access the status information of the vehicle and even control the vehicle by injecting messages into the bus. The most famous vehicle attack occurred in 2015 when Miller and Valasek hacked a moving Jeep and caused a car accident, eventually leading to the recall of 1.4 million vehicles by Chrysler Corporation [7].

The vulnerabilities of the CAN bus enable adversaries to control the vehicle by modifying the transmitted message once they access the IVN. The ECU receives malicious messages from the adversary without authentication. However, it does not recognize the sender of the message. Messages transmitted between two nodes are vulnerable to spoofing attacks if there are no authentication procedures. Adversaries can easily eavesdrop on the unencrypted message transmitted on the CAN bus, which is convenient for forging the attack message. To prevent such attacks, it is necessary to apply an encryption and authentication scheme on the CAN bus.

Some researchers have used symmetric or asymmetric cryptography to encrypt messages transmitted on the CAN bus to prevent the leakage of transmitting messages. They also considered lightweight authentication methods to reduce authentication latency [3]. But there are still some unresolved problems in these schemes. For instance, some schemes rely on some long-term symmetric keys to distribute other keys, or only one party is responsible for generating keys. All these approaches result in new security and privacy risks. Considering the above problems, in this study, we present a lightweight encryption and authentication scheme based on the key distribution solution. It is worth mentioning that this scheme overcomes the dependence on long-term symmetric keys.

A. *Our Contribution*

There are several challenges in designing a security scheme that can be applied to the CAN bus environment of AVs. First, the transmission of messages over the CAN bus must be real-time [8]. To ensure the safety of people in the car when driving at high speed, the message must be rapidly transmitted, which requires the time delay of the security protocol to be very low. Second, most ECUs deployed in vehicles have limited computing performance because of cost issues [9]. It is impossible to complete computational operations with the high time complexity while ensuring real-time performance. Third, the load capacity of the CAN bus is limited, and the CAN protocol stipulates that a CAN data frame can carry a maximum of eight bytes of data. This study proposes a lightweight encrypted authentication protocol for secure communication between ECUs based on the abovementioned limitations. The main contributions of this study are as follows:

- The Blom key management scheme was first applied to an IVN environment. A new lightweight encryption and authentication scheme based on the Blom scheme was proposed for the CAN bus of AVs. The proposed scheme overcomes the dependence of other IVN security schemes on long-term session keys. It is difficult to guarantee that the session key stored in the AV in-vehicle environment for an extended time will not be obtained by the adversary. Moreover, there is a pairwise key generation scheme to ensure that new nodes can communicate with old nodes.
- The security proof using Burrows-Abadi-Needham (BAN) logic and informal security analysis prove that the proposed scheme can guarantee the security of the ECU communication. In addition, Raspberry Pi hardware and IVN simulation software CANoe were used to analyze the performance of the proposed scheme. The experimental results of time overhead and bus load show that this scheme is more lightweight than other schemes.

B. *Organization of the Rest Article*

The remainder of this paper is organized as follows. Section II introduces the related work. Section III describes the preparation and related background knowledge. Section IV provides a detailed description of the proposed security protocol. Section V presents a security analysis of the proposed scheme. The performance evaluation of our scheme is presented in Section VI. Finally, Section VII concludes this study.

II. Related Work

AV external interfaces connect the IVN with the outside environment, increasing the vehicle's functionality, but exposing the insecure IVN to hackers. Attacks on vehicles generally fall into two categories: attacks launched from the inside and those launched from the outside. Attacks launched from the inside are through interfaces inside the car such as onboard diagnostics-2 (OBD-2) and CD players. Koscher et al. [10] injected fake CAN messages into the bus by connecting a self-written injection tool through the OBD-2 port to control the key components of the vehicles. Hoppe et al. [11] implemented ECU reprogramming via code injection in a test rig comprised of real automotive hardware. Typically, such attacks require physical contact with the attacked vehicle, which is difficult to implement.

The external attack on the vehicle can be carried out far from the vehicle by using a wireless interface in which the vehicle communicates with the outside world to attack the vehicle, such as Wi-Fi, Bluetooth, and sensors. The Keen Security Lab of Tencent [12] used Tesla Model S as the test object and fully accessed the CAN bus from external sources through Wi-Fi. Woo et al. [13] successfully implemented a replay attack from the outside through a wireless channel connected to an IVN. Checkoway et al. [14] used a telematics unit in a vehicle to run remote malicious code via Bluetooth and a long-range wireless connection. In addition to the attack method of injecting malicious messages into the network, Cho and Shin [15] used a denial of service (DOS) attack called the bus-off attack to force the ECU not to receive and send messages or even shut down the entire CAN network. A bus-off attack can significantly affect vehicle operation.

At present, the methods for attacking IVN are gradually increasing. To ensure the safety of drivers, security measures must be taken to protect IVN. One of the main methods is an intrusion detection system (IDS). An IDS detects forged malicious data or unauthorized access in a network [16]. The main research direction of current IDSs is based on traffic or physical layer anomalies. Song et al. [17] proposed a lightweight IDS for IVNs based on CAN message time-interval analysis. The IDS can detect all message injection attacks. Cho et al. [18] designed an IDS based on clock feature anomalies to check the sender's ECU based on the estimated clock skew. Subsequently, Ying et al. [19] proposed a new masquerading attack and formally analyzed

Cho's scheme. The voltage anomaly based IDS designed by Choi et al. [20] identifies fingerprints using the voltage signal features.

An IDS, as a defense method in an IVN and can effectively detect a few known attacks. Its disadvantage is that the problem can only be detected after the attack has continued for a while, and the occurrence of the attack cannot be prevented. The use of cryptography to protect the IVN environment is another primary research direction for IVN security. Because the data in the CAN bus are transmitted in plaintext, the adversary can launch an eavesdropping attack with ease, thereby collecting the data frames transmitted in the CAN bus. Introducing the encryption mechanism into the CAN bus can prevent that situation. The message frame is broadcast by an ECU or adversary over the CAN bus, and all the nodes attached to the bus can choose to receive the message. The remainder of this subsection provides an overview of these security schemes.

Woo et al. [13] conducted a remote wireless attack on a connected vehicle using a malicious smartphone application. Combined with the proposed wireless attack experiment, a security protocol was designed to compensate for the vulnerability of the CAN bus and prevent the occurrence of attacks. The protocol not only considers secure communication between ECUs but also designs a program to establish a key for the external device connected to the CAN bus so that the message transmitted by the external device can be verified. The disadvantage of this scheme is that the session keys of all ECUs are the same. Once the key of a compromised ECU is leaked, the other ECUs on the bus are not secure.

Palaniswamy et al. [21] conducted an informal security analysis of the protocol suite proposed by Woo. They found several defects in the protocol suite and proposed a new protocol to mitigate these identified weaknesses. Finally, researchers used professional security protocol analysis tools to analyze the security of the new scheme.

Jo et al. [22] designed a node-based centralized authentication method to prevent masquerading attacks initiated by compromised nodes. Each node on the bus only shares the key with an authentication entity called $Auth$. All the messages transmitted by the ECU must be verified by $Auth$. If the verification fails, $Auth$ broadcasts a warning message over the bus. Because all protocols must rely on $Auth$, once $Auth$ has a problem, it is easy to cause a single point of failure. The time required for the ECU to receive the message and wait for $Auth$ to send the error message is also a critical factor.

The choice of key type is a problem that needs to be considered in encryption and authentication mechanisms [23]. The key types used in IVN security schemes can be classified into group, global and pairwise. A scheme using group keys [10] classifies the ECUs into different groups. ECUs in the same group share a unique key, so ECUs in different groups cannot decrypt and authenticate messages from each other. If an ECU is compromised and an adversary obtains the key, all ECUs of the same group are at risk. If all ECUs in the IVN are in the same group, the shared group key is called the global key. The advantage of using a global key is that it is easy to manage; however, it has the same drawbacks as using a group key, and the scope of the impact is greater. The pairwise scheme [24] establishes a key between each pair of communicating ECUs. A compromised pairwise key only affects the communication of a pair of ECUs. Each ECU stores many pairwise keys, so the key management process is complicated. Our scheme uses paired keys as encryption and authentication keys and the Blom key management scheme [25] simplifies the key distribution and updating process, which compensates for the shortcomings of using paired keys to a certain extent. Specific steps of the Blom scheme are introduced in the next section.

III. Preliminaries and Background

This section briefly presents a background on cryptography including Blom key management scheme and Grain stream cipher. The rest of this section describes in detail the application environment CAN bus, CAN frame format, system model, and security requirements of the encrypted authentication scheme.

A. Blom Key Management Scheme

Blom scheme is a matrix-based pairwise key distribution solution [25]. The scheme can ensure that each node accessing the network can securely obtain the key to communicate with other nodes while avoiding excessive communication overhead. Two matrices are defined over a finite field $GF(q)$ in the Blom scheme, where q is a large prime number. The two matrices are the $(\lambda + 1) \times N$ public matrix G (any λ+1 column of G is linearly independent) and the $(\lambda + 1) \times (\lambda + 1)$ symmetric secret matrix D, where N is the number of nodes and λ is the security parameter. As long as the number of compromised ECUs does not exceed λ, the entire network is secure. The pairwise keys for communication between nodes can be established using matrices D and G. The specific process is as follows:

$$A = (D \cdot G)^T \tag{1}$$

$$K = A \cdot G \tag{2}$$

Matrix A is calculated by multiplying matrix D and G and then transposing. The matrix obtained by multiplying matrix A and G is the symmetric key-matrix K. Elements of matrix K are pairwise keys used by nodes. For example, the session key K_{ij} of node i and node j is the value of the ith row and the jth column of the matrix K or the value of the jth row and the ith column of the matrix K. The generation process of the key-matrix K is shown in Fig. 1.

Node i stores the ith row vector α_i of matrix A as private information and the ith column vector β_i of matrix G as public information. Afterward, node i obtains β_j from node j and calculates the pairwise key K_{ij} by the following formula. Node j can calculate the key K_{ij} in the same mode so that both nodes i and j have obtained the uniform session key.

$$K_{ij} = \alpha_i \cdot \beta_j = \alpha_j \cdot \beta_i = K_{ji} \tag{3}$$

Considering the overhead of storage and transmission, the public matrix G is generally in the form of a Vandermonde matrix [26] in practical applications. Since each column of the

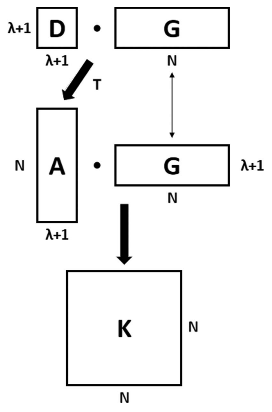


Fig. 1. The generation process of the key-matrix.

Vandermonde matrix is generated by a generator g, nodes only need to store and exchange g.

B. Grain Stream Cipher

Due to the performance limitations of the ECU and real-time requirements of the IVN, it is hard to use the public key encryption algorithm to encrypt messages on the CAN bus. Symmetric encryption algorithms are more suitable for performance-constrained hardware than the asymmetric cryptographic algorithm. Stream cipher and block cipher are the main two types of symmetric ciphers. The 64-bit payload of the CAN frame cannot satisfy the requirements of the block cipher for the length of the plaintext, so the stream cipher algorithm is the most suitable encryption algorithm for the CAN bus environment.

The Grain encryption algorithm is a well-known lightweight stream cipher algorithm based on the nonlinear feedback shift register, including Grain v1 [27], Grain-128 [28], and Grain-128a [29]. Grain v1 is the winning algorithm of the eStream project of the European Serial Cryptography Project, and Grain-128a is the encryption authentication standard of the international radio frequency identification (RFID) technology.

Since Grain v1 only supports 80-bit keys and 64-bit initial vectors (IVs), Hell et al. [28] proposed Grain-128, which supports 128-bit keys and 96-bit IVs. However, Grain-128 is now no longer recommended owing to a key recovery attack on Grain-128 that is simpler than the brute-force attack. Agren et al. [29] described a new version of Grain-128 named Grain-128a. The Grain-128a algorithm also supports 128-bit keys and 96-bit IVs, and has the function of authentication. Due to the lack of effective attack methods, Grain v1 and Grain-128a still have high security.

Our scheme uses the algorithm of the Grain-128a encryption part to encrypt the transmitted CAN message. Due to the limited space of the article, this part does not introduce the process of the Grain-128a encryption algorithm in detail.

C. Controller Area Network

The controller area network was designed by the German BOSCH company in 1986 as a medium for transmitting information between automotive ECUs or sensors. The functions supported by the CAN bus include power train, engine management, anti-brake system, and transmission [30]. Although the CAN bus has security deficiencies as mentioned above and its maximum transmission speed is only 1 Mb/s, the CAN bus is still indispensable for many vehicles, including autonomous vehicles. Improved versions of CAN protocols such as CAN+ protocol [31] and CAN with flexible data-rate protocol (CAN-FD) [32] have also appeared in recent years but they all require expensive hardware.

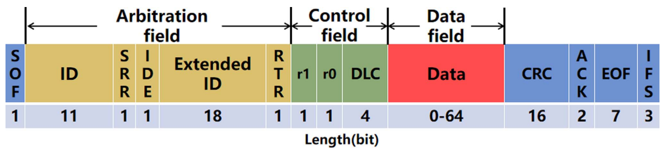


Fig. 2. Extended CAN frame format.

The CAN bus is a serial data communication protocol, and the CAN data frame is composed of several fields such as the arbitration field, control field, and data field. The arbitration field contains an 11-bit identifier field. The identifier (ID) represents the type of transmission message, which is generally determined by the ECU that sent the message frame, and other ECUs choose whether to receive the CAN message according to the identifier. As the number of ECUs increases, the number of identifiers also increases. In order to prevent the identifier field from being insufficient, a new 18-bit extended identifier field is set in the extended CAN frame. The identifier extension (IDE) bit in the arbitration field determines whether the frame is standard or extended. The specific format is shown in Fig. 2. If the remote transmission request (RTR) bit of a CAN frame is recessive, it indicates that the frame is an RTR frame. The task of the RTR frame is to request a node to send a message [33].

The data length code (DLC) in the control field represents the length of the data field. The transmitted data is stored in the data field of the CAN frame, and the maximum length of the data does not exceed eight bytes. The function of the cyclic redundancy check (CRC) field in the CAN frame is to check the frame to prevent errors during transmission, and this security mechanism is far from enough in the CAN bus.

In addition to indicating the message type, the identifier also represents the priority of the message. Only one message is broadcast on the CAN bus at the same time. If more than two ECUs simultaneously transmit messages, the arbitration mechanism will determine which CAN message to transmit first through the message ID. The arbitration mechanism compares the identifiers of the frames bit by bit. The smaller identifier of the frame, the higher priority of the frame, and the message with the higher priority will be transmitted first.

D. Network Model

Our scheme is applied to the hierarchical CAN bus structure model, as shown in Fig. 3.

The IVN is comprised of several ordinary ECUs and a key ECU (KECU) with relatively high costs and excellent performance. In fact, there are nodes with outstanding performance

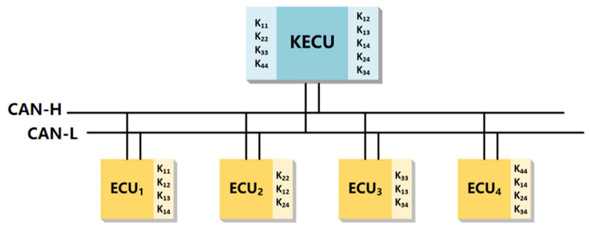

K_{11} K_{22} K_{33} K_{44}
KECU
K_{12} K_{13} K_{14} K_{24} K_{34}
CAN-H
CAN-L
ECU_1 K_{11} K_{12} K_{13} K_{14}
ECU_2 K_{22} K_{12} K_{24}
ECU_3 K_{33} K_{13} K_{34}
ECU_4 K_{44} K_{14} K_{24} K_{34}

Fig. 3. Network Model.

like KECU in the actual IVN, such as gateway ECU (GECU). KECU differs from GECU in that there is no remote interface for communication with external networks, which can reduce the risk of remote intrusion. Assuming that KECU is absolutely secure, the secret information stored in it cannot be leaked. Both KECU and ECU send messages on the CAN bus.

KECU only participates in the key distribution and updating process of ECUs, which stores the correlation matrix required by the Blom key management scheme and the keys for communicating with other ECUs. Each ordinary ECU has a unique serial number, which is convenient for transmitting corresponding messages during the key distribution process. An ECU stores the key communicated with the KECU and the session key between the ECUs that need to communicate.

E. *Adversary Model*

We consider an adversary that can access the CAN bus through an external interface on the vehicle or a compromised ECU. Among the external interfaces, the OBD-2 port is the physical interface often used by adversaries. An adversary can damage the safety of a driving car by controlling the CAN bus. The adversary attached to the CAN bus can eavesdrop on messages broadcast transmitted on the bus and be hard to detect. In addition, the adversary can also launch the forgery attack, disguised as other ECUs sending messages. Replay attacks are also frequently used attacks.

Although the adversary can compromise the common ECUs but cannot destroy the KECU. There are no direct remote interfaces between KECU and the outside world, and other nodes must be carefully checked before connecting to KECU through physical interfaces. A tamper-proof device (TPD) can be installed on the KECU to enhance security. TPD prevents adversaries from extracting and modifying stored data at short notice [34]. Additionally, DOS attacks can occur on the bus but we do not discuss DOS attacks in this article. Cryptographic schemes are not the best solution to DOS attacks, and methods such as intrusion detection can better resist DOS attacks.

F. *Security Requirements*

As proposed in [10], [11], [12], [13], [14], [15], there have been a variety of attack methods on vehicles. Many of these attacks stem from the flaw of the CAN bus [35]. The CAN bus of the autonomous vehicle is an open environment in which nodes can send and receive any messages, so an adversary accessing the CAN bus can easily achieve the purpose of the attack. In this case, applying an authentication protocol to the CAN bus is a way to improve bus security. The authentication protocol for CAN bus should satisfy the following security requirements.

1) Communicating Entity Authentication: Each transmitted CAN data frame has an ID indicating the type and source of the frame. A normal ECU only sends messages with its ID but an adversary can simply fabricate and send messages with any ID. Because of the lack of authentication, none of the ECUs can determine the genuine source of the messages on the CAN bus. If the entity connecting the bus can authenticate, the integrity and authenticity of the message can be guaranteed.
2) Resistance to Message Eavesdropping Attack: Message frames are broadcast in the form of plaintext over the CAN bus, and all entities connecting, including the adversary, can receive the data. The adversary can analyze leaked CAN messages to determine the meaning of each data frame. The ECU encrypts a message before sending it to ensure the confidentiality of the message and can resist eavesdropping attacks from adversaries.
3) Resistance to Message Replay Attack: The adversary saves some messages transmitted on the CAN bus and resends them later. In this way, it is possible to interfere with the regular operation of the vehicle. The replay attack is an attack that is easy to implement on the CAN bus, and it is necessary to prevent the replay attack to ensure the safety of the vehicle.
4) Resistance to Message Forgery Attack: The data on the CAN frame have a definite meaning, and the compromised ECU can send messages with forged ID and data to achieve the purpose of destruction. An adversary can fabricate meaningless data as well as meaningful data, all of which have an impact on the vehicle. The CAN bus should be able to withstand the message forgery attack.
5) Session Key Freshness: Even though the session key is stored in the TPD, the adversary may still be able to obtain the correct key by brute force cracking. The key used in the security scheme cannot remain unchanged for a long time, otherwise the key will be at risk of being cracked by brute force. The scheme should specify a starting condition for key updating process to ensure key freshness. In this way, the risk of adversaries obtaining keys by brute force is reduced.
6) Session Key Secrecy: When a session key is comprised, the adversary cannot infer the past session key because of the comprised key, nor can they obtain future session keys. The proposed protocol assures forward and backward secrecy of the session key [36].

IV. The Proposed Scheme

This section details our proposed encryption and authentication scheme for the CAN bus in the IVN. The keys used in the scheme are generated by the Blom key management scheme. The proposed protocol has five main phases, including system initialization, key updating, message encryption and authentication, message counter re-synchronization, session key

TABLE I
NOTATIONS AND DEFINITIONS USED

Notation	Definition
KECU	High-performance ECU
ECU_i	Sender ECU
ECU_j	Receiver ECU
ID_i	Identifier of ECU_i
α_i	Row vector of ECU_i in matrix A
β_i	Column vector of ECU_i in matrix K
K_{ii}, K_{ij}	Generation key of corresponding session key
EK_{ii}	Encryption key for ECU_i to communicate with KECU
AK_{ii}	Authentication key for ECU_i to communicate with KECU
EK_{ij}	Encryption key for ECU_i to communicate with ECU_j
AK_{ij}	Authentication key for ECU_i to communicate with ECU_j
$mCTR_{ij}$	Message counter shared by ECU_i and ECU_j
$f(k)$	Key derivation function using key k
$H_k()$	Keyed hash function using key k
MAC	Generated message authentication code
$C_k()$	Grain encryption function using key k
$D_k()$	Grain decryption function using key k
IV_i	Initial vector generated by ECU_i
PM_i	Plaintext generated by ECU_i
CM_i	Ciphertext generated by ECU_i

distribution with new nodes. The notations used in the proposed scheme are shown in Table I.

Above all, the system initializes public parameters and distributes them to all ECUs. During system initialization, each ECU broadcasts messages related to key generation on the CAN bus. After that, all ECUs communicate with KECU in priority order to receive the message and calculate the key. In order to ensure the security of the key, the ECU regularly updates the key through the KECU. All ECUs send CAN encrypted messages, enclosing the corresponding authentication values generated using their session keys. The ECU receives and verifies the message using the corresponding key. When message validation fails, the message counter re-synchronization phase is performed. In this way, it can ensure that the counters maintained by the pair of ECUs are consistent. When a new node is connected to the CAN bus, a new key can be assigned to it through the proposed scheme to ensure that the new node can communicate with other nodes normally.

A. System Initialization

Initial session keys between KECU and ECUs are established at this phase. At the same time, the ECUs also obtain keys to communicate with other ECUs. The following steps present the session key derivation procedure of ECUs with KECU.

1) ECUs broadcast their generators g according to ECU priority which is based on ID priority. If a ECU has two or more IDs, the highest priority ID is used for comparison. The generator g can be calculated to generate a vector $(g^0, g^1, g^2, \ldots, g^\lambda)$ as a column of the public matrix G. In other words, each ECU broadcasts its column vector β on the CAN bus. In this way, all ECUs on the bus can receive the required column vector messages.
2) After receiving all generators from all ECUs, KECU generates a random secret symmetric matrix D and saves it. KECU then uses all generators received to compute the corresponding column vectors, which can form a complete public matrix G.
3) By multiplying matrix D and G, KECU obtains a new matrix. The new matrix becomes matrix A after transposing. KECU transmits the row vector of matrix A to the corresponding ECU in order. KECU then computes the key-matrix $K = A \cdot G$. The communication key between ECU_i and ECU_j is K_{ij}, which is the ith row and the jth column of K. The ith row and the ith column of K is the key generation key K_{ii} between the ECU_i and KECU.
4) The ECU receives its row vector α and then calculates session keys to communicate with other ECUs. If ECU_i transmits a message to ECU_j, ECU_i needs to use the generator of ECU_j to calculate β_j. ECU_i multiplies row vector α_i and column vector β_j to get the key generation key K_{ij}. Similarly, ECU_j multiplies row vector α_j and column vector β_i to get K_{ij}. Then ECU_i and ECU_j can obtain the encryption key EK_{ij} and authentication key AK_{ij} through $f(K_{ij}) = (EK_{ij}, AK_{ij})$. ECU_i then computes $f(K_{ii}) = (EK_{ii}, AK_{ii})$ to communicate with the KECU and initializes all message counters mCTR to zero. To prevent key leakage, all session keys and α are stored in the tamper-proof units of the corresponding ECUs.

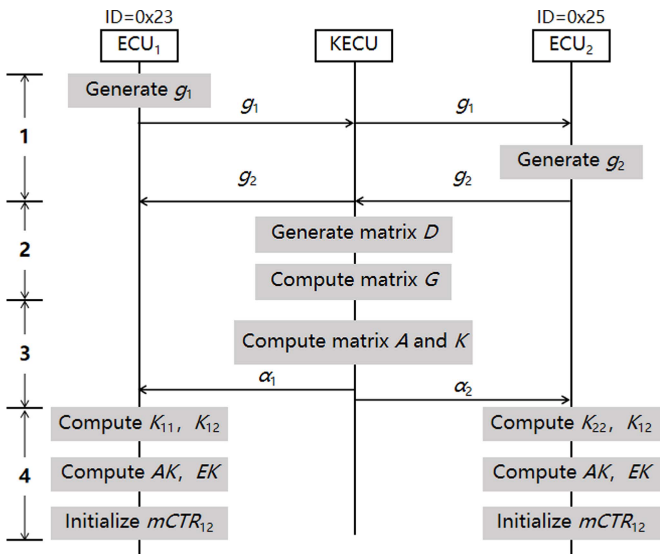


Fig. 4. System Initialization.

Fig. 4 illustrates the system initialization process between a single KECU and two ECUs. The process only occurs during vehicle production. The IVN is secure when the vehicle is manufactured, so there is no need to authenticate messages. After the system initialization process, each pair of communicating ECUs will have the same symmetric session.

B. Key Updating

To ensure secure communication, the session authentication keys should be updated with new values. Our proposed scheme characterizes a simple key updating procedure. KECU updates the old key-matrix K to prevent possible attacks owing to key exposure and transmits the row vector of new matrix A to the corresponding ECU in order, as explained below.

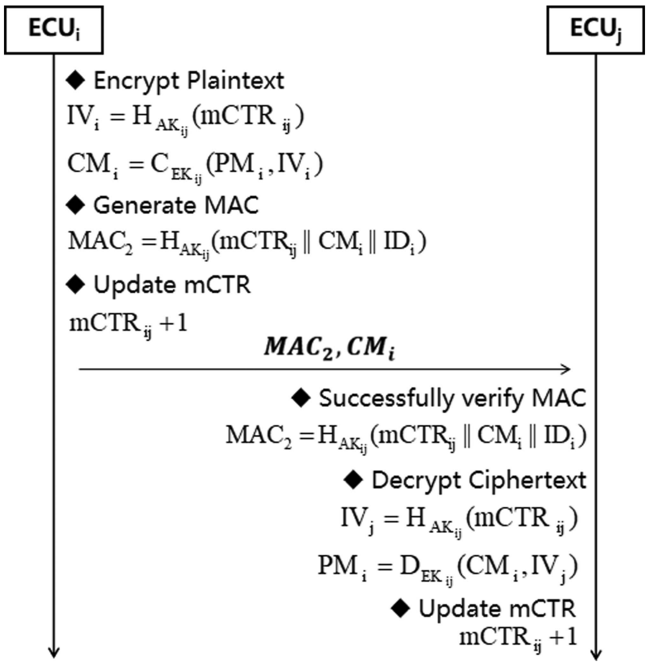
1) KECU regenerates a new secret symmetric matrix D_{new} and computes a new matrix $A_{new} = (D_{new} \cdot G)^T$. In other words, the row vectors corresponding to all ECUs are updated.
2) Before transmits the row vector α_i^{new} of matrix A_{new} to ECU_i, KECU performs an XOR operation between α_i^{new} and α_i to generate α'. KECU computes $MAC_1 = H_{AK_{ii}}(ID_{KECU}||\alpha')$ using AK_{ii}, where AK_{ii} is the authentication key between KECU and ECU_i. α' and MAC_1 are then transmitted to ECU_i.
3) ECU_i verifies the MAC_1. If the verification is successful, ECU_i decrypts α' with $\alpha_i^{new} = \alpha' \oplus \alpha_i$ and α_i^{new} should be multiplied by β_j to form the new session key K_{ij}^{new}. ECU_i can get session keys with different ECUs by using different β.

Key updating operation is performed when the vehicle is started or the value of a message counter exceeds the maximum value. If the ECU's message counter overflows, the ECU should send a key updating message to the KECU. KECU that receives the updating message only needs to transmit a message to each ECU to complete the updating of all keys in the protocol. After updating the session keys, KECU and all ECUs set all message counters to zero and begin to transmit messages with the new keys.

C. Message Encryption and Authentication

In order to ensure the authenticity and confidentiality of messages transmitted on the CAN bus, ECUs must encrypt CAN message and calculate message authentication codes (MACs). So Encrypt-then-MAC (EtM) is used to ensure the security of the transmitted message. Namely, the plaintext is encrypted to get a ciphertext, and then the MAC of the ciphertext is calculated. In the proposed protocol, the ECU encrypts a message using a stream key encryption algorithm called Grain-128a, the details of which are beyond the scope of this article. In addition, we use the keyed-hash MAC to ensure that the received message has not been modified. The sender ECU_i and the receiver ECU_j are a pair of communicating ECUs, and the message encryption and authentication process is described as follows.

1) ECU_i computes initial vector $IV_i = H_{AK_{ij}}(mCTR_{ij})$ using AK_{ij}. The IV_i is an important input value in the Grain encryption algorithm. The initial vector generated by the abovementioned formula can be guaranteed to be difficult to forge.
2) ECU_i generates ciphertext CM_i using EK_{ij} and IV_i from $CM_i = C_{EK_{ij}}(PM_i, IV_i)$. The plaintext PM_i is the message that was originally intended to be transmitted.
3) ECU_i generates a MAC value for the message that includes CM_i and $mCTR_{ij}$ as follows: $MAC_2 = H_{AK_{ij}}(mCTR_{ij}||CM_i||ID_i)$. ECU_i broadcasts the CAN message with CM_i and MAC_2 on the CAN bus and increments $mCTR_{ij}$.
4) The receiver ECU_j verifies the MAC_2 from ECU_i using AK_{ij}. If MAC_2 is valid, ECU_j generates initial vector IV_j. Since the calculation uses the same key and message counter, IV_i and IV_j are equal. After calculating IV_j correctly, ECU_j performs decryption and obtains the right plaintext PM_i using the following: $PM_i = D_{EK_{ij}}(CM_i, IV_j)$. Finally, ECU_j increments the $mCTR_{ij}$ by 1.



ECU$_i$
ECU$_j$
◆ Encrypt Plaintext
$IV_i = H_{AK_{ij}}(mCTR_{ij})$
$CM_i = C_{EK_{ij}}(PM_i, IV_i)$
◆ Generate MAC
$MAC_2 = H_{AK_{ij}}(mCTR_{ij} \| CM_i \| ID_i)$
◆ Update mCTR
$mCTR_{ij} + 1$
MAC_2, CM_i
◆ Successfully verify MAC
$MAC_2 = H_{AK_{ij}}(mCTR_{ij} \| CM_i \| ID_i)$
◆ Decrypt Ciphertext
$IV_j = H_{AK_{ij}}(mCTR_{ij})$
$PM_i = D_{EK_{ij}}(CM_i, IV_j)$
◆ Update mCTR
$mCTR_{ij} + 1$

Fig. 5. Message Encryption and Authentication.

Fig. 5 shows details of the message transmission process. Through the abovementioned process, messages are no longer transferred over the CAN bus as plaintext. The main advantage of using the stream cipher mechanism is that the key stream can be calculated in advance using the session key and the IV. The sender ECU performs an XOR operation between key stream and message. In this way, the time delay caused by the encryption operation can be reduced. This method can also be used when the receiver ECU decrypts messages. It should be noted that we do not transmit all the bits of the MAC. Since the CAN data frame payload is eight bytes, we can truncate the MAC to four bytes for transmission.

D. Message Counter Re-Synchronization

On receiving the ciphertext message and truncated MAC from the sender node, the receiver node decrypts messages directly if the MAC can be verified. If the verification is failed, message counters may be inconsistent and the receiver ECU_j will perform a message counter re-synchronization process to confirm the reason for the authentication failure. The details of the re-synchronization process are described as follows.

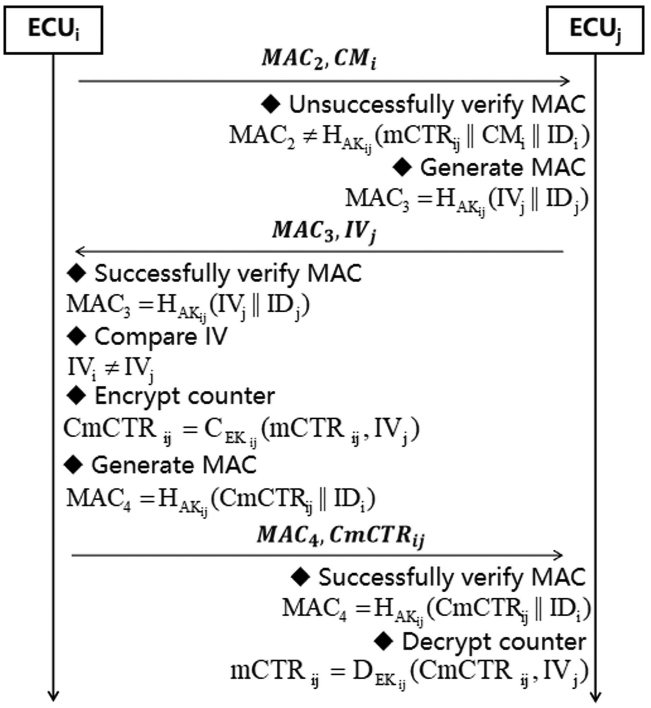
1) When the received MAC_2 cannot be successfully verified, ECU_j sends a counter verification message to the sending node. ECU_j uses its own saved counter and the key AK_{ij} to calculate the initial vector IV_j and generates a MAC value for IV_j as follows: $MAC_3{=}H_{AK_{ij}}(IV_j||ID_j)$. The counter verification message contains the IV_j and MAC_3.
2) The sender ECU_i receives and verifies the MAC_3. If MAC_3 is valid, ECU_i will compare IV_j with IV_i. Assuming IV_j is equal to IV_i, this means that the counter does not need to be re-synchronized. ECU_i only needs to resend the previously sent message. If these two values are different, ECU_i sends the relevant message to keep the counters of the two ECUs consistent.
3) ECU_i generates ciphertext $CmCTR_{ij}$ using EK_{ij} and IV_j from $CmCTR_{ij}{=}C_{EK_{ij}}(mCTR_{ij},\,IV_j)$. The $mCTR_{ij}$ is the message counter stored in ECU_i. ECU_i computes $MAC_4 = H_{AK_{ij}}(CmCTR_{ij}||ID_i)$ using AK_{ij}. ECU_i sends the CAN message with $CmCTR_{ij}$ and MAC_4 to ECU_j to modify its counter.
4) ECU_j verifies the MAC_4 from ECU_i using AK_{ij}. If MAC_4 is valid, ECU_j performs decryption using the following formula $mCTR_{ij}{=}D_{EK_{ij}}(CmCTR_{ij},\,IV_j)$ to obtain plaintext $mCTR_{ij}$. The ECU_j resets its counter to the value of the decrypted message counter.

After the message counter is updated, the ECU_i needs to broadcast the previously unauthenticated message again to avoid losing important information. Through this process, the possible reasons for a MAC to fail verification can be checked. One is that the message counters are not synchronized, or the transmitted message data is wrong. Although this process increases the time delay, authentication failures do not occur frequently in a normal CAN network. So re-synchronization does not have a significant impact on message transmission overall. Fig. 6 shows details of the message counter re-synchronization process.

E. Session Key Distribution With New Nodes

It is common for a new node to connect to the CAN bus through an external interface. In our scheme, if a node does not have a session key, it cannot communicate with other nodes. This protocol allows establishing a secret key between KECU and a new node. Assume that the new node is a reliable device. The session key can be distributed to the new node by following the steps below.

1) Initial data transmission: The new node ECU_{new} and the KECU transmit messages through the trusted channel. The trusted channel is established by the official service provider through a direct wired connection. KECU first generates α_{new} and β_{new}. KECU then transmits the α_{new} of the ECU_{new} and the β of other ECUs that the ECU_{new} wants to communicate with.
2) Counter setting: The ECU_{new} generates the session key through inner product operations. Since each key corresponds to a counter, the ECU_{new} sets the corresponding message counter and initializes it.
3) New parameter broadcasting: The KECU broadcasts the β_{new} of the ECU_{new} on the CAN bus. Any node attached to the bus can receive the message, and calculate the session key with the ECU_{new}. Finally, ECUs set the corresponding message counter.



ECUi
ECUj
MAC2, CMi
◆ Unsuccessfully verify MAC
$MAC_2 \neq H_{AK_{ij}}(mCTR_{ij} \| CM_i \| ID_i)$
◆ Generate MAC
$MAC_3 = H_{AK_{ij}}(IV_j \| ID_j)$
MAC3, IVj
◆ Successfully verify MAC
$MAC_3 = H_{AK_{ij}}(IV_j \| ID_j)$
◆ Compare IV
$IV_i \neq IV_j$
◆ Encrypt counter
$CmCTR_{ij} = C_{EK_{ij}}(mCTR_{ij}, IV_j)$
◆ Generate MAC
$MAC_4 = H_{AK_{ij}}(CmCTR_{ij} \| ID_i)$
MAC4, CmCTRij
◆ Successfully verify MAC
$MAC_4 = H_{AK_{ij}}(CmCTR_{ij} \| ID_i)$
◆ Decrypt counter
$mCTR_{ij} = D_{EK_{ij}}(CmCTR_{ij}, IV_j)$

Fig. 6. Message Counter Re-synchronization.

V. Security Analysis

EtM is a frequently used combination encryption authentication scheme. Bellare et al. [37] proved that EtM achieves indistinguishability under chosen-plaintext attack (IND-CPA) assuming that the given symmetric encryption scheme is secure against chosen-plaintext attack (CPA) and the given MAC is existential unforgeable under chosen-message attack (EU-CMA).

Ban logic, characterized by its simplicity, direction, and strong analytic capability, can find major security loopholes and redundant protocol flows [38]. This section presents a formal security analysis using BAN logic for the proposed security protocol. In addiation, a informal security analysis of the required security objectives is performed.

A. Formal Security Analysis Using BAN Logic

In this subsection, we provide a formal security analysis of our proposed authentication protocol using BAN logic [39]. Through the following rigorous derivation process, we demonstrate the security of messages transmitted between two ECUs. Table II shows the notations used in the proof process.

In order to implement the analysis process well, the basic logical postulates of BAN logic can be described as follows. The formulas above the horizontal line represent hypothetical situations, and the ones below the horizontal line represent the corresponding results.

- R_1(Message-meaning rule):

TABLE II
BASIC NOTATIONS OF THE BAN LOGIC

Notation	Definition
P, Q	Principals
X, Y	Statements
K	Seession key
$P \mid\equiv X$	P believes X
$P \triangleleft X$	P sees X
$P \mid\sim X$	P once said X
$P \Rightarrow X$	P has jurisdiction over X
$\{X\}_K$	X encrypted under the key K
$P \overset{K}{\longleftrightarrow} Q$	P and Q use the shared key K to communicate
$\langle X\rangle_Y$	X combined with the formula Y
$\#(X)$	X is fresh
$P \overset{Y}{\rightleftharpoons} Q$	Y is a secret known only to P and Q

$\frac{P|\equiv P\overset{K}{\longleftrightarrow}Q, P\triangleleft\{X\}_K}{P|\equiv Q|\sim X}$ or $\frac{P|\equiv P\overset{Y}{\rightleftharpoons}Q, P\triangleleft\langle X\rangle_Y}{P|\equiv Q|\sim X}$

- R_2(Nonce-verification rule):
  $\frac{P|\equiv\#(X), P|\equiv Q|\sim X}{P|\equiv Q|\equiv X}$
- R_3(Jurisdiction rule):
  $\frac{P|\equiv Q\Rightarrow X, P|\equiv Q|\equiv X}{P|\equiv X}$
- R_4(Freshness-conjuncatenation rule):
  $\frac{P|\equiv\#(X)}{P|\equiv\#(X,Y)}$ or $\frac{P|\equiv\#(X)}{P|\equiv\#(\{X\}_K)}$
- R_5(Belief rule):
  $\frac{P|\equiv(X,Y)}{P|\equiv X}$ or $\frac{P|\equiv Q|\equiv(X,Y)}{P|\equiv Q|\equiv X}$

We mainly analyze the security of the message encryption and authentication process and the message counter re-synchronization process in the scheme. According to the previously proposed security requirements, these two phases must meet the following goals.

- G_1: $ECU_j| \equiv CM_i$
- G_2: $ECU_j| \equiv PM_i$
- G_3: $ECU_j| \equiv \#(MAC_2)$
- G_4: $ECU_j| \equiv \#(CM_i)$
- G_5: $ECU_i| \equiv \#(IV_j)$
- G_6: $ECU_i| \equiv IV_j$
- G_7: $ECU_i| \equiv \#(MAC_3)$
- G_8: $ECU_j| \equiv \#(CmCTR_{ij})$
- G_9: $ECU_j| \equiv CmCTR_{ij}$
- G_{10}: $ECU_j| \equiv mCTR_{ij}$
- G_{11}: $ECU_j| \equiv \#(MAC_4)$

In the message encryption and authentication phase, ECU_i sends MAC_2 and CM_i to ECU_j. If ECU_j unsuccessfully verifies MAC_2, the message counter re-synchronization process will begin. ECU_j sends MAC_3 and IV_j to ECU_i, then ECU_i responds MAC_4 and $CmCTR_{ij}$ to ECU_j. For the convenience of description, we represent the original protocol with the following idealized protocol.

- M_1: $ECU_i \to ECU_j$: (MAC_2, CM_i): $(\langle mCTR_{ij}, CM_i\rangle_{AK_{ij}}, \{PM_i, mCTR_{ij}\}_{AK_{ij}})$
- M_2: $ECU_j \to ECU_i$: (MAC_3, IV_j): $(\langle IV_j\rangle_{AK_{ij}}, IV_j)$
- M_3: $ECU_i \to ECU_j$: $(MAC_4, CmCTR_{ij})$: $(\langle CmCTR_{ij}\rangle_{AK_{ij}}, \{mCTR_{ij}\}_{AK_{ij}})$

In addition, we need to make the following assumptions about the initial state.

- A_1: $ECU_j| \equiv ECU_j \overset{AK_{ij}}{\rightleftharpoons} ECU_i$
- A_2: $ECU_j| \equiv ECU_j \overset{AK_{ij}}{\longleftrightarrow} ECU_i$
- A_3: $ECU_j| \equiv \#(mCTR_{ij})$
- A_4: $ECU_j| \equiv ECU_i \Rightarrow CM_i$
- A_5: $ECU_j| \equiv ECU_i \Rightarrow PM_i$
- A_6: $ECU_i| \equiv ECU_i \overset{AK_{ij}}{\rightleftharpoons} ECU_j$
- A_7: $ECU_i| \equiv \#(mCTR_{ij})$
- A_8: $ECU_i| \equiv ECU_j \Rightarrow IV_j$
- A_9: $ECU_j| \equiv ECU_i \Rightarrow CmCTR_{ij}$
- A_{10}: $ECU_j| \equiv ECU_i \Rightarrow mCTR_{ij}$

Based on the abovementioned BAN logic rules and assumptions, we can derive the proposed goals.The specific proof process of the message encryption and authentication protocol is as follows.

- S_1: According to M_1, we can obtain: $ECU_j \triangleleft (\langle mCTR_{ij}, CM_i\rangle_{AK_{ij}}, \{PM_i, mCTR_{ij}\}_{AK_{ij}})$
- S_2: According to R_1, A_1 and S_1, we have: $ECU_j| \equiv ECU_i| \sim (mCTR_{ij}, CM_i)$
- S_3: According to R_1, A_2 and S_1, we have: $ECU_j| \equiv ECU_i| \sim (PM_i, mCTR_{ij})$
- S_4: According to R_4 and A_3, we deduce: $ECU_j| \equiv \#(mCTR_{ij}, CM_i)$
- S_5: According to R_2, S_2 and S_4, we can obtain: $ECU_j| \equiv ECU_i| \equiv (mCTR_{ij}, CM_i)$
- S_6: According to R_5 and S_5, we can obtain: $ECU_j| \equiv ECU_i| \equiv CM_i$
- S_7: According to R_3, A_4 and S_6, we can get G_1: $ECU_j| \equiv CM_i$
- S_8: According to R_4 and A_3, we deduce: $ECU_j| \equiv \#(PM_i, mCTR_{ij})$
- S_9: According to R_2, S_3 and S_8, we can obtain: $ECU_j| \equiv ECU_i| \equiv (PM_i, mCTR_{ij})$
- S_{10}: According to R_5 and S_9, we can obtain: $ECU_j| \equiv ECU_i| \equiv PM_i$
- S_{11}: According to R_3, A_5 and S_{10}, we can get G_2: $ECU_j| \equiv PM_i$
- S_{12}: According to R_4 and S_4, we can get G_3: $ECU_j| \equiv \#(MAC_2)$
- S_{13}: According to R_4 and S_8, we can get G_4: $ECU_j| \equiv \#(CM_i)$

Similarly, the steps for the proof of the message counter re-synchronization protocol are as follows.

- S_{14}: According to M_2, we can obtain: $ECU_i \triangleleft (\langle IV_j\rangle_{AK_{ij}}, IV_j)$
- S_{15}: According to R_1, A_6 and S_{14}, we have: $ECU_i| \equiv ECU_j| \sim IV_j$
- S_{16}: According to R_4 and A_7, we can get G_5: $ECU_i| \equiv \#(IV_j)$
- S_{17}: According to R_2, S_{15} and S_{16}, we can obtain: $ECU_i| \equiv ECU_j| \equiv IV_j$
- S_{18}: According to R_3, A_8 and S_{17}, we can get G_6: $ECU_i| \equiv IV_j$

- S_{19}: According to R_4 and S_{16}, we can get G_7: $ECU_i| \equiv \#(MAC_3)$
- S_{20}: According to M_3, we can obtain: $ECU_j \triangleleft (\langle CmCTR_{ij} \rangle_{AK_{ij}}, \{mCTR_{ij}\}_{AK_{ij}})$
- S_{21}: According to R_1, A_1 and S_{20}, we have: $ECU_j| \equiv ECU_i| \sim CmCTR_{ij}$
- S_{22}: According to R_1, A_1 and S_{20}, we have: $ECU_j| \equiv ECU_i| \sim mCTR_{ij}$
- S_{23}: According to R_4 and A_3, we can get G_8: $ECU_j| \equiv \#(CmCTR_{ij})$
- S_{24}: According to R_2, S_{21} and S_{23}, we can obtain: $ECU_j| \equiv ECU_i| \equiv CmCTR_{ij}$
- S_{25}: According to R_3, A_9 and S_{24}, we can get G_9: $ECU_j| \equiv CmCTR_{ij}$
- S_{26}: According to R_2, A_3 and S_{22}, we can obtain: $ECU_j| \equiv ECU_i| \equiv mCTR_{ij}$
- S_{27}: According to R_3, A_{10} and S_{26}, we can get G_{10}: $ECU_j| \equiv mCTR_{ij}$
- S_{28}: According to R_4 and S_{23}, we can get G_{11}: $ECU_j| \equiv \#(MAC_4)$

G_1, G_6, and G_9 indicate that the ECU can authenticate correct messages and prevent message forgery attacks. G_3, G_4, G_5, G_7, G_8, and G_{11} indicate that the freshness of the message can be guaranteed to prevent replay attacks. The formal BAN logic analysis of our authentication scheme shows that the authentication between ECUs on the CAN bus can be successfully achieved.

B. Informal Security Analysis

This subsection details how the proposed scheme is resistant to various attacks. In addition, the analysis is carried out for other security requirements.

1) Communicating Entity Authentication: In the proposed scheme, the ECU uses MAC to verify the transmitted message, which is the authentication method recommended by automotive open system architecture (AUTOSAR) to be used on the CAN bus [24]. The sender ECU calculates the MAC value of data before transmitting the message, and then sends the truncated MAC together with the message. The receiver ECU verifies the message after receiving the MAC. In order to maintain the normal busload, the truncated MAC is generally four bytes. The MAC used in the protocol is the hash-based message authentication code (HMAC), and its value changes depending on the message and the key used. An adversary cannot generate the correct MAC value for a meaningful message without the corresponding key. If the adversary randomly generates a 32-bit MAC value, the probability of generating the correct MAC value is $1/2^{32}$. The attacker can continuously send a randomly generated MAC and the same message on the CAN bus for testing. Calculated at the frequency of sending a message in 10 ms, it takes up to 11,930 hours to determine the result [13], which is quite different from the running time of the vehicle. The proposed scheme stipulates that the key needs to be updated every time the message counter overflows or the vehicle restarts, so it is more difficult for the adversary to implement the brute force attack of the MAC.
2) Resistance to Message Eavesdropping Attack: Data transmitted over the CAN must be in ciphertext form to prevent eavesdropping attacks. In our scheme, the Grain-128a stream cipher algorithm is used to realize the function of transmitting data encryption. The keys used for encryption are distributed during the initialization phase and updated later through the key updating protocol. The IV used in the encryption algorithm is generated using a key and a counter. Grain-128a is a well-known stream cipher that has resisted all types of single-key attacks [40]. Therefore, the adversary cannot decrypt the messages on the bus to obtain relevant information without obtaining the session key.
3) Resistance to Message Replay Attack: As long as the uniqueness of each transmission message is guaranteed, the CAN bus can withstand replay attacks. Our scheme generates distinct messages by setting message counters. The sender ECU and the receiver ECU manage the message counter, and the same counter of both is synchronized. In the message encryption and authentication process, since the IV is generated by the message counter, the ciphertext of the same message after encryption is different. The counter is automatically incremented by one after each message is successfully transmitted, so the receiver ECU cannot successfully decrypt and authenticate the previous message sent by the adversary. We also designed the message counter re-synchronization process to ensure that the counters do not become out of sync.
4) Resistance to Message Forgery Attack: An adversary typically injects data frames into the bus to attack the vehicle. The fabricated data frame has adverse effects on other ECUs. After applying this scheme to the CAN bus, all messages sent are accompanied by the corresponding MAC. If the verification is unsuccessful, the ECU does not accept this message. Since the adversary does not have the corresponding session key, its fabricated messages can not be received by any ECU. Therefore, the proposed scheme can withstand the forgery attack.
5) Session Key Freshness: All keys used in the scheme are updated when the vehicle starts and message counters overflow. The updated keys include session keys between normal ECUs and keys used by each ECU to communicate with the KECU. In the key updating protocol, KECU random generates a new secret matrix D. Since the key-matrix K is calculated from matrix D, the freshness of the key is guaranteed.
6) Session Key Secrecy: The key type used in our security protocol is the pairwise key, and a pairwise key ensures secure communication between a pair of ECUs. If an adversary obtains a pairwise key, it can use that key to communicate but cannot compute the corresponding previous and next round keys. The session key is calculated using the inner product of vectors, and it is difficult to inversely calculate the row vector using the key and the public column vector. In the key updating process, the KECU encrypts the new row vector with the old row

vector and transmits it. If the adversary cannot obtain the row vector corresponding to the current key, then row vectors of the previous round and the next round cannot be obtained either. The adversary cannot calculate the key without the row vector, so our scheme satisfies the forward and backward key secrecy requirement.

VI. Performance Analysis and Comparison

To prove that the proposed lightweight scheme satisfies real-time requirements of AVs, we use Raspberry Pi 3B board and CANoe bus simulation software to evaluate the overall performance. We choose three protocols [13], [21], [22] to compare with ours to illustrate the advantages of our scheme in terms of communication overhead.

A. Experimental Details

The CANoe is a software for bus environment simulation developed by the German company Vector [41]. The software can simulate the transmission of data on the CAN bus of autonomous vehicles. We measure the load of CAN bus by setting up multiple virtual nodes and transmitting data frames. In the simulation experiment, four virtual nodes are arranged on the bus, and a node is used as the KECU. The built-in CANoe scripting language CAPL is used to simulate the behavior of ECU receiving and sending messages. The baud rate of the simulated CAN bus can be set flexibly.

We run the scheme code on a Raspberry Pi 3B and record the time overhead because Raspberry Pi has the same 64-bit ARM core processor as advanced ECUs in the AV [42]. The program of the proposed scheme is written in C programming language, and the cryptography part of the scheme is implemented by Miracl cryptographic library. The protocol parameters in the experiment are set as follows: λ is 4, N is 50, session keys are 128 bits, and the truncated HMAC calcuated by the SHA-256 algorithm is 32 bits.

During the driving process of the AV, a large number of messages are continuously transmitted over the CAN bus. At this time, the message encryption and authentication protocol and the key updating protocol in our scheme are the most frequently executed protocols, so we mainly measure the performance of these two protocols. The time overhead and bus load illustrate that this solution does not cause delays to message transmission on the CAN bus, and does not affect the operation of the AV. The running time of the protocol is mainly the computation delay of the ECU and KECU. The delay time is tested by Raspberry Pi hardware. When using CANoe to measure the busload, we measure the load of the initial CAN bus without any additional security scheme, and use the result as a baseline for comparing with the CAN busload of the applied security scheme to illustrate the impact of schemes on the busload.

B. Time Overhead

a) Key updating time overhead: The KECU performs matrix multiplication to generate a new key-matrix K during the key updating process. The calculation time of KECU is related to the number of ECUs connected to the CAN bus. We use Raspberry Pi as KECU, and Fig. 7 shows the time for KECU to update the key-matrix under different ECU numbers. It can be seen from the experimental results that the time for KECU to generate keys increases with the increase of the number of ECUs. Although KECU spends a lot of time generating the matrix in the updating phase, it does not affect real-time performance. Because the idle KECU can complete the calculation process of the matrix in advance, once the key is updated, the key message can be directly transmitted to each ECU without calculation delay.

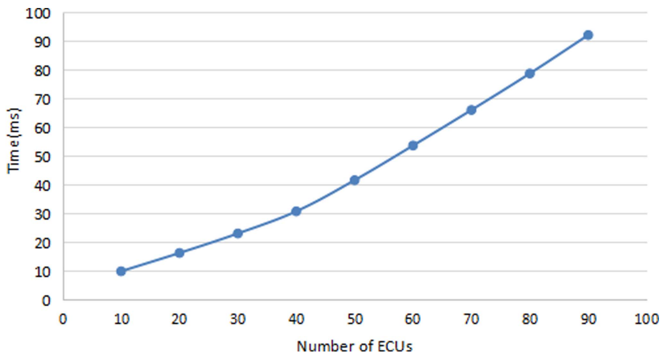


Fig. 7. Time for updating the key matrix.

Before transmitting the key message to an ECU, the KECU performs an XOR operation and generates a MAC value. After receiving the key message transmitted by KECU, the ECU verifies the received MAC and obtains the session key through a vector inner product operation. Table III shows the average time required for a single operation in the key updating process.

b) Message encryption and authentication time overhead: The sender ECU encrypts the transmitted message before transmitting the data on the CAN bus. Generating a pseudo-random keystream is the main time overhead in the stream cipher encryption algorithms, and encryption is an XOR operation. In order to ensure the security of the ciphertext, the IV used when generating the random keystream is generated by the HMAC algorithm. Although these steps consume a lot of time, the encryption operation does not delay the sending of the message. Because computing the keystream and IV is not directly related to the plaintext message to be transmitted, the ECU can calculate keystream and IV ahead of time during idle time, thereby reducing the time overhead when sending messages. This is one of the reasons why we chose stream cipher. After encrypting the message, the ECU needs to calculate the MAC of the ciphertext and send the ciphertext and MAC.

The operations performed by the receiver ECU are different from those performed by the sender only in the sequence of steps, so the description will not be repeated. Table III shows the specific time overhead of the ECU in the message encryption and authentication process.

C. Busload

The busload generally represents the size of the message transmitted on the bus. In the case of a limited bus transmission rate, the busload is also limited within a certain period of time.

TABLE III
Average Time for a Single operation(μs)

Key Generation	Message Encryption	Message Authentication	Vector Multiplication
41630.25	5152.6	111.55	14.48

TABLE IV
Busload for CAN Bus

	Min busload	Max busload	Avg busload
Normal	30.26%	30.65%	30.63%
Applied protocol	36.98%	37.46%	37.44%

The busload rate represents the usage of a bus, its meaning is the ratio of the actual number of bits transmitted on the bus to the maximum number of bits transmitted in the period of time. Under normal circumstances, the load rate of the CAN bus is about 30%, and it does not exceed 70% at most [43]. When the load rate is too high, the communication of the bus may be congested, and real-time performance of the bus cannot be guaranteed. In extreme cases, error frames may appear.

In order to ensure the authenticity of the message, the calculated MAC is sent together with the message. The transmission of MAC can increase the busload, which is the price that must be paid to ensure the security of the in-vehicle network, but we can still use some measures to minimize the additional load and impact. The maximum payload of a CAN frame is eight bytes. Although the MAC has been truncated to four bytes, it is impossible to put the MAC into the data field without changing the message length, so we use the Woo's method [13] to split the MAC into two parts and place them in two fields of the data frame. The first two bytes of the MAC are filled in the extended identifier field of the frame. The last two bytes are filled in the CRC field because the application of the MAC completely replaces the role of the CRC.

We tested the busload rate in two scenarios with the CAN bus baud rate of 500 Kb/s. The first scenario is a normal CAN bus scenario without any security protocol attached, and the experimental results serve as a baseline for the results of the second scenario. The second scenario is the CAN bus with the security protocol applied. The results of the two tests are compared to show the influence of the security protocol on the bus.

Table IV shows the experimental results under the two scenarios. From the results, the application of our scheme in the CAN bus increases the busload rate. However, the added extra load is within the normal range and does not exceed 70% of the dangerous line, and has no serious impact on the message transmission.

D. *Performance Comparison*

This subsection presents the performance comparison of our scheme with Woo et al. [13], Palaniswamy et al. [21] and MAuth [22]. For the comparison, we consider both time overhead and busload. For the convenience of description, we denote the Woo protocol as WSP, the Palaniswamy scheme as EAS, and our scheme as Ours. EAS is an improved solution to WSP. As shown in Table V, we perform a detailed analysis of the compared protocols.

The parameters set by the other three schemes should be the same as our scheme to ensure the accuracy of the experimental results. Specifically, the key used is 128 bits, the random seed value and transmitted message are 64 bits, and the truncated MAC tag is 32 bits.

Tables VI and VII show the time overhead of each entity in the key updating protocol and message authentication protocol of each scheme, respectively. Since there are high-performance entities in the four schemes, such as the GECU in the WSP and EAS schemes, the authenticator in the MAuth scheme, and the KECU in our scheme, they are collectively referred to as high-end ECUs (HECU) in Table VI. Because the description of the key updating process in the MAuth scheme is not detailed enough, there is no MAuth scheme in Table VI.

In Tables VI and VII, T_M represents the time to perform one MAC operation, T_A represents the time to perform one AES encryption operation, and T_v represents the time of one vector multiplication. We use the message authentication time in Table III as the time of the MAC operation. For convenience, we use the AES encryption time measured with Raspberry Pi 3B in [44] as the value of T_A. The time for AES encryption operation is 10 ms. T_w represents the waiting time of the ECU during authentication. There is no specific waiting time given in the MAuth scheme, so we cannot calculate the total time overhead of MAuth.

Although encrypting the message increases the time overhead, the impact on the actual overhead can be avoided by precomputing. From the total time overhead results in Tables VI and VII, our scheme has lower time overhead than the other three schemes.

We apply the four security protocols to the simulated CAN bus respectively and compare the busload results obtained. When there are no adversary attacks, the sender only needs to transmit a CAN frame to the receiver to authenticate the message in our scheme and WSP scheme. The EAS scheme utilizes two RTR frames to transmit a MAC to ensure the integrity and authenticity of the RTR frame. This way can increase the load of the CAN bus. In the MAuth scheme, the corresponding ECU broadcasts the message after the authenticator sends the trigger data frame, which means that at least two CAN frames are transmitted, so the load rate of the bus applying the MAuth scheme is higher.

Fig. 8 shows the busload rates of the CAN bus with four security protocols and the common CAN bus without the security protocol at different baud rates. Because the initialization phase only occurs when the vehicle is manufactured or started and does not have a continuous impact on the CAN busload, the communication overhead of the initialization phase of all

TABLE V
COMPARISON OF SECURITY PROTOCOLS FOR CAN BUS

	WSP [13]	EAS [21]	MAuth [22]	Ours
Authentication key type	Global key	Global key	Pairwise key	Pairwise key
Long term key	Yes	Yes	Yes	No
Authentication mode	Distribution	Distribution	Centralization	Distribution
Message encryption	Yes	Yes	No	Yes
Network scalability	Yes	Yes	No	Yes

TABLE VI
TIME OVERHEAD OF KEY UPDATING PROTOCOL

	WSP [13]	EAS [21]	Ours
HECU	$2T_M$+T_A	$2T_M$	T_M
ECU	$2T_M$+T_A	$2T_M$	T_M+T_v
Total time overhead	$4T_M$+$2T_A \approx$ 20.45ms	$4T_M \approx$ 0.45ms	$2T_M$+$T_v \approx$ 0.24ms

TABLE VII
TIME OVERHEAD OF MESSAGE AUTHENTICATION PROTOCOL

	WSP [13]	EAS [21]	MAuth [22]	Ours
Sender ECU	T_M+T_A	$2T_M$+T_A	$2T_M$	$2T_M$
Receiver ECU	T_M+T_A	$2T_M$+T_A	T_M+T_w	$2T_M$
Total time overhead	$2T_M$+$2T_A \approx$ 20.22ms	$4T_M$+$2T_A \approx$ 20.45ms	$3T_M$+$T_w \approx$ 0.34+$T_w$ms	$4T_M \approx$ 0.45ms

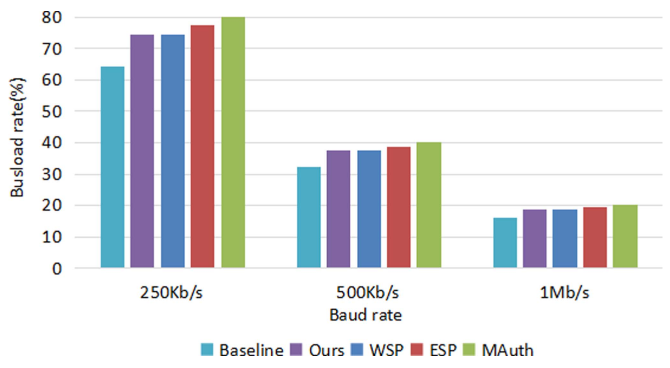


Fig. 8. The busload rate of the CAN bus.

protocols is not considered. Likewise, all parameter updating protocols, including key updating protocols, are not considered. The key updating process with a frequency of 10 minutes does not have adverse consequences for CAN bus communication compared to the average frequency of 10 ms of data message transmission.

Based on the results shown in Fig. 8, the CAN busload of our scheme is lower than that of other schemes, and it is closer to the baseline. In addition, the gap between the busloads of the schemes becomes more and more pronounced as the baud rate decreases.

VII. CONCLUSION

The emergence of AVs has improved people's lives but vehicle safety has become an issue that must be addressed. More external interfaces lead to higher risks and more severe consequences for AVs than regular vehicles. This study proposed a secure and efficient lightweight encryption authentication scheme for an in-vehicle CAN bus. First, to solve the problem that the key distribution of many ECUs in vehicles takes too much time and the process is complicated, we modified the Blom key management algorithm and used it in the key distribution process of ECUs on the CAN bus. The designed key updating scheme can update the session keys of all ECUs within a short time. Second, the Grain stream cipher and the MAC authentication mechanisms are utilized to realize the encryption and authentication in the message transmission process. These two cryptographic methods are acceptable in a limited IVN environment. Furthermore, formal security analysis using BAN logic and informal security analysis show that the proposed scheme is secure and can resist attacks such as forgery and replay. Finally, using hardware and software to test the performance of the scheme, it was proven that the scheme does not affect real-time performance of the IVN nor cause an excessive load on the CAN bus. We chose three schemes for comparison, and the results showed that the proposed scheme is superior. In the future, we plan to enhance security in a more complete and complex hybrid IVN environment for AVs.

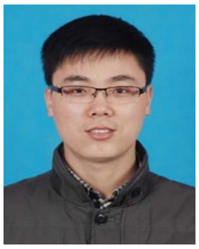
ACKNOWLEDGMENT

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

REFERENCES

[1] J. Takahashi, "An overview of cyber security for connected vehicles," *IEICE Trans. Inf. Syst.*, vol. 101, no. 11, pp. 2561–2575, 2018.
[2] J. Cui, L. S. Liew, G. Sabaliauskaite, and F. Zhou, "A review on safety failures, security attacks, and available countermeasures for autonomous vehicles," *Ad Hoc Netw.*, vol. 90, 2019, Art. no. 101823.
[3] K. Ren, Q. Wang, C. Wang, Z. Qin, and X. Lin, "The security of autonomous driving: Threats, defenses, and future directions," *Proc. IEEE*, vol. 108, no. 2, pp. 357–372, Feb. 2020.
[4] T. Huang, J. Zhou, Y. Wang, and A. Cheng, "On the security of in-vehicle hybrid network: Status and challenges," in *Proc. Int. Conf. Inf. Secur. Pract. Experience*. 2017, pp. 621–637.
[5] M. Bozdal, M. Samie, S. Aslam, and I. Jennions, "Evaluation of CAN bus security challenges," *Sensors*, vol. 20, no. 8, 2020, Art. no. 2364.
[6] X. Sun, F. R. Yu, and P. Zhang, "A survey on cyber-security of connected and autonomous vehicles (CAVs)," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 7, pp. 6240–6259, Jul. 2022.

[7] C. Miller and C. Valasek, "Remote exploitation of an unaltered passenger vehicle," *Black Hat USA*, vol. 2015, pp. 1–91, 2015.
[8] Q. Ye, "Research and application of CAN and LIN bus in automobile network system," in *Proc. IEEE 3rd Int. Conf. Adv. Comput. Theory Eng.*, 2010, pp. V6-150–V6-154.
[9] Q. Wang and S. Sawhney, "VeCure: A practical security framework to protect the CAN bus of vehicles," in *Proc. IEEE Int. Conf. Internet Things*, 2014, pp. 13–18.
[10] K. Koscher et al., "Experimental security analysis of a modern automobile," in *Proc. IEEE Symp. Secur. Privacy*., 2010, pp. 447–462.
[11] T. Hoppe, S. Kiltz, and J. Dittmann, "Security threats to automotive CAN networks–practical examples and selected short-term countermeasures," in *Proc. Int. Conf. Comput. Saf., Rel., Secur.*. 2008, pp. 235–248.
[12] S. Nie, L. Liu, and Y. Du, "Free-fall: Hacking Tesla from wireless to CAN bus," *Briefing, Black Hat USA*, vol. 25, pp. 1–16, 2017.
[13] 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.
[14] S. Checkoway et al., "Comprehensive experimental analyses of automotive attack surfaces," in *Proc. 20th USENIX Secur. Symp.*, 2011, pp. 447–462.
[15] K.-T. Cho and K. G. Shin, "Error handling of in-vehicle networks makes them vulnerable," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2016, pp. 1044–1055.
[16] M. Pham and K. Xiong, "A survey on security attacks and defense techniques for connected and autonomous vehicles," *Comput. Secur.*, vol. 109, 2021, Art. no. 102269.
[17] H. M. Song, H. R. Kim, and H. K. Kim, "Intrusion detection system based on the analysis of time intervals of CAN messages for in-vehicle network," in *Proc. IEEE Int. Conf. Inf. Netw.*, 2016, pp. 63–68.
[18] K.-T. Cho and K. G. Shin, "Fingerprinting electronic control units for vehicle intrusion detection," in *Proc. 25th USENIX Secur. Symp.*, 2016, pp. 911–927.
[19] X. Ying, S. U. Sagong, A. Clark, L. Bushnell, and R. Poovendran, "Shape of the cloak: Formal analysis of clock skew-based intrusion detection system in controller area networks," *IEEE Trans. Inf. Forensics Secur.*, vol. 14, no. 9, pp. 2300–2314, Sep. 2019.
[20] W. Choi, K. Joo, H. J. Jo, M. C. Park, and D. H. Lee, "Voltageids: Low-level communication characteristics for automotive intrusion detection system," *IEEE Trans. Inf. Forensics Secur.*, vol. 13, no. 8, pp. 2114–2129, Aug. 2018.
[21] B. Palaniswamy, S. Camtepe, E. Foo, and J. Pieprzyk, "An efficient authentication scheme for intra-vehicular controller area network," *IEEE Trans. Inf. Forensics Secur.*, vol. 15, pp. 3107–3122, 2020.
[22] H. J. Jo, J. H. Kim, H.-Y. Choi, W. Choi, D. H. Lee, and I. Lee, "MAuth-CAN: Masquerade-attack-proof authentication for in-vehicle networks," *IEEE Trans. Veh. Technol.*, vol. 69, no. 2, pp. 2204–2218, Feb. 2020.
[23] 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.
[24] A.-I. Radu and F. D. Garcia, "Leia: A lightweight authentication protocol for CAN," in *Proc. 21st Eur. Symp. Res. Comput. Secur.*, 2016, pp. 283–300.
[25] R. Blom, "An optimal class of symmetric key generation systems," in *Proc. Workshop Theory Appl. Cryptographic Techn.*, 1984, pp. 335–338.
[26] M. Rahman, S. Sampalli, and S. Hussain, "A robust pair-wise and group key management protocol for wireless sensor network," in *Proc. IEEE Globecom Workshops*, 2010, pp. 1528–1532.
[27] M. Hell, T. Johansson, and W. Meier, "Grain: A stream cipher for constrained environments," *Int. J. Wireless Mobile Comput.*, vol. 2, no. 1, pp. 86–93, 2007.
[28] M. Hell, T. Johansson, A. Maximov, and W. Meier, "A stream cipher proposal: Grain-128," in *Proc. IEEE Int. Symp. Inf. Theory*, 2006, pp. 1614–1618.
[29] M. Agren, M. Hell, T. Johansson, and W. Meier, "Grain-128a: A new version of Grain-128 with optional authentication," *Int. J. Wireless Mobile Comput.*, vol. 5, no. 1, pp. 48–59, 2011.
[30] E. Aliwa, O. Rana, C. Perera, and P. Burnap, "Cyberattacks and countermeasures for in-vehicle networks," *ACM Comput. Surv.*, vol. 54, no. 1, pp. 1–37, 2021.
[31] T. Ziermann, S. Wildermann, and J. Teich, "CAN: A new backward-compatible controller area network (CAN) protocol with up to 16× higher data rates," in *Proc. IEEE Des., Automat. Test Europe Conf. Exhib.*, 2009, pp. 1088–1093.
[32] S. Woo, H. J. Jo, I. S. Kim, and D. H. Lee, "A practical security architecture for in-vehicle CAN-FD," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 8, pp. 2248–2261, Aug. 2016.
[33] H. Chen and J. Tian, "Research on the controller area network," in *Proc. IEEE Int. Conf. Netw. Digit. Soc.*, 2009, pp. 251–254.
[34] C. Zhang, R. Lu, X. Lin, P.-H. Ho, and X. Shen, "An efficient identity-based batch verification scheme for vehicular sensor networks," in *Proc. IEEE 27th Conf. Comput. Commun.*, 2008, pp. 246–250.
[35] S. Hartzell, C. Stubel, and T. Bonaci, "Security analysis of an automobile controller area network bus," *IEEE Potentials*, vol. 39, no. 3, pp. 19–24, May/Jun. 2020.
[36] R. Di Pietro, L. V. Mancini, and S. Jajodia, "Providing secrecy in key management protocols for large wireless sensors networks," *Ad Hoc Netw.*, vol. 1, no. 4, pp. 455–468, 2003.
[37] M. Bellare and C. Namprempre, "Authenticated encryption: Relations among notions and analysis of the generic composition paradigm," *J. Cryptol.*, vol. 21, no. 4, pp. 469–491, 2008.
[38] J. Wen, M. Zhang, and X. Li, "The study on the application of ban logic in formal analysis of authentication protocols," in *Proc. 7th Int. Conf. Electron. Commerce*, 2005, pp. 744–747.
[39] M. Burrows, M. Abadi, and R. M. Needham, "A logic of authentication," *Proc. Roy. Soc. London. A. Math. Phys. Sci.*, vol. 426, no. 1871, pp. 233–271, 1989.
[40] V. A. Ghafari and H. Hu, "A new chosen IV statistical attack on Grain-128a cipher," in *Proc. IEEE Int. Conf. Cyber-Enabled Distrib. Comput. Knowl. Discov.*, 2017, pp. 58–62.
[41] F. Zhou, S. Li, and X. Hou, "Development method of simulation and test system for vehicle body CAN bus based on canoe," in *Proc. IEEE 7th World Congr. Intell. Control Automat.*, 2008, pp. 7515–7519.
[42] T. Limbasiya, A. Ghosal, and M. Conti, "Autosec: Secure automotive data transmission scheme for in-vehicle networks," in *Proc. 23rd Int. Conf. Distrib. Comput. Netw.*, 2022, pp. 208–216.
[43] S. Fassak, Y. E. H. El Idrissi, N. Zahid, and M. Jedra, "A secure protocol for session keys establishment between ECUs in the CAN bus," in *Proc. IEEE Int. Conf. Wireless Netw. Mobile Commun.*, 2017, pp. 1–6.
[44] S. Jebri, A. Ben Amor, M. Abid, and A. Bouallegue, "Enhanced lightweight algorithm to secure data transmission in IoT systems," *Wireless Pers. Commun.*, vol. 116, no. 3, pp. 2321–2344, 2021.



Jie Cui (Senior Member, IEEE) was born in Henan Province, China, in 1980. He received the Ph.D. degree from the University of Science and Technology of China, Hefei, China, in 2012. He is currently a Professor and Ph.D. Supervisor of the School of Computer Science and Technology, Anhui University, Hefei. He has more than 150 scientific publications in reputable journals, which include 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. His research interests include applied cryptography, IoT security, vehicular ad hoc network, cloud computing security and software-defined networking (SDN).

Yaning Chen is currently a Research Student with the School of Computer Science and Technology, Anhui University, Hefei. His research focuses on the security of vehicular ad hoc network.

Hong Zhong was born in Anhui Province, China, in 1965. She received the Ph.D. degree in computer science from the University of Science and Technology of China, Hefei, China, in 2005. She is currently a Professor and Ph.D. supervisor of the School of Computer Science and Technology, Anhui University. She has more than 200 scientific publications in reputable journals, which include 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. Her research interests include applied cryptography, IoT security, vehicular ad hoc network, cloud computing security and software-defined networking.

Debiao He received the Ph.D. 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. 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. His main research interests include cryptography and information security, in particular, cryptographic protocols. He was the recipient of the 2018 IEEE Systems Journal Best Paper Award and 2019 IET Information Security Best Paper Award. His work has been cited more than 10000 times at Google Scholar. He is on 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*.

Lu Wei is currently currently working toward the Ph.D. degree from the School of Computer Science and Technology, Anhui University, Hefei, China. He has more than 10 scientific publications in reputable journals, which include IEEE Transactions on Information Forensics and Security, IEEE Journal on Selected Areas in Communications, IEEE Transactions on Mobile Computing, and IEEE Transactions on Intelligent Transportation Systems. His research interests include vehicular ad hoc networks and applied cryptography.

Irina Bolodurina received the Ph.D. degree from South Ural State University, Chelyabinsk, Russia. She is currently a Professor and the Head of Department of Applied Mathematics, Orenburg State University, Orenburg, Russia. He has more than 60 scientific publications in academic journals and international conferences which indexing in Scopus and WoS. She has participated in more than 20 scientific projects supported by the RFBR and other Russian scientific programs. Her research interests include theory of optimal control, mathematical modeling, information analysis software, control of social and economic systems, decision support systems, data integration, and processing.

Lu Liu received the Ph.D. degree from the University of Surrey, Guildford, U.K., and the M.Sc. degree in data communication systems from Brunel University, Uxbridge, U.K. He is currently the Professor of informatics and the Head of the School of Informatics, University of Leicester, Leicester, U.K. His research interests include the areas of cloud computing, service computing, computer networks and peer-to-peer networking. He is a Fellow of British Computer Society.