EC-SVC: Secure CAN Bus In-Vehicle Communications with Fine-grained Access Control Based on Edge Computing

Donghyun Yu, Ruei-Hau Hsu, *Member, IEEE*, and Jemin Lee, *Member, IEEE*

***Abstract*—In-vehicle communications are not designed for message exchange between the vehicles and outside systems originally. Thus, the security design of message protection is insufficient. Moreover, the internal devices do not have enough resources to process the additional security operations. Nonetheless, due to the characteristic of the in-vehicle network in which messages are broadcast, secure message transmission to specific receivers must be ensured. With consideration of the facts aforementioned, this work addresses resource problems by offloading secure operations to high-performance devices, and uses attribute-based access control to ensure the confidentiality of messages from attackers and unauthorized users. In addition, we reconfigure existing access control based cryptography to address new vulnerabilities arising from the use of edge computing and attribute-based access control. Thus, this paper proposes an edge computing-based security protocol with fine-grained attribute-based encryption using a hash function, symmetric-based cryptography, and reconfigured cryptographic scheme. In addition, this work formally proves the reconfigured cryptographic scheme and security protocol, and evaluates the feasibility of the proposed security protocol in various aspects using the CANoe software.**

***Index Terms*—in-vehicle security, access control, attribute-based encryption, edge computing**

I. INTRODUCTION

With the noticeable improvements in vehicles, internal devices in a vehicle start to share an amount of essential data of the car with each other. The most significant advantage of data sharing is to improve their data processing performance. For instance, electronic control unit (ECU), which is the most common machine sharing data in a car, can exchange information in order to make an important decision rapidly [1]. Therefore, it is definitely indispensable for ECU to increase the amount of shared data for its decision performance in the car. Unfortunately, however, the increase in data to be shared among the internal devices does not always guarantee good results due to the existence of latent attackers whose objectives are to eavesdrop or manipulate the data for misbehaved operations. The message eavesdropping and modification attacks can be even effective against the current vehicles because the typical in-vehicle communications protocol does not provide data confidentiality and message authentication, which are the most basic requirements for secure communications [2]-[7]. In particular, even the most representative in-vehicle communications protocol, that is, controller area network (CAN) protocol, has already been considered to be unable to satisfy the significant security requirements. This is mainly due to its obsoleteness, while it has been widely utilized in-vehicle communications [2], [8]. Moreover, CAN is widely used in industrial control system (ICS) including electronic equipments for aviation and navigation, medical devices and equipments, industrial automation and mechanical control, as well as vehicles. All of these systems are close to real-time systems, which can be fatal if security issues occur. Hence, additional security mechanism such as the fine-grained access control of message exchange among different entities is urgently desired in CAN. Therefore, to solve the security issues of CAN, many researchers have proposed new security protocols for CAN. The details of previous work are described in the following Sec. I-A.

A. Related Work

Initial works recognize the problem that ECUs have insufficient resources, such as power and computing capability to perform cryptographic protocols for secure in-vehicle communications [9]-[11]. Pierre et al. [10] presented the challenges of achieving high-security requirements with the insufficient power resources in in-vehicle networks. Hisashi et al. [9] proposed an attestation-based security architecture for in-vehicle communications using the trusted platform module (TPM) in all ECUs. They designated resource-rich ECUs as master ECUs, and used the key predistribution system (KPS) for the authentication of the software configuration and the authenticated and encrypted communications. Hendrik et al. [11] provided a security solution the hardware security module (HSM) at ECUs and a key master. In the above mentioned papers, they dealt directly or indirectly the problems of the limited computing power of ECUs by using additional hardwares such as TPM and HSM in ECUs. However, as mentioned earlier, mounting additional hardwares on every ECUs is costly, which is practically impossible.

Some works have also been proposed for in-vehicle communications security without mounting additional hardware at ECUs [12]-[15]. Herrewege et al. [12] proposed the backward-compatible broadcast authentication protocol in CAN. Bogdan et al. [13] provided a security protocol based on symmetric primitives using key splitting and message authentication code (MAC) mixing. They utilized nodes with high-computing power for distributing keys and a mixed MAC to achieve inter-group authentication, not one-to-one authentication. Ryo et al. [14] focused on the problem of detecting spoofing messages in CAN bus, and proposed the lightweight authentication method. Chung-Wei et al. [15] provided a security mechanism to keep CAN bus utilization low. In the above mentioned papers, they

---

Corresponding author is J. Lee.

D. Yu and J. Lee are with Daegu Gyeongbuk Institute of Science and Technology (DGIST), 333, Techno Jungang-daero, Daegu, Republic of Korea 42988 (e-mail: {`xaos4715`, `jmnlee`}`@dgist.ac.kr`).

R. H. Hsu is with National Sun Yat-sen University, 70 Lienhai Rd., Kaohsiung 80424, Taiwan, R.O.C. (e-mail: `rhhsu@mail.cse.nsysu.edu.tw`).

did not ensure the confidentiality of the data, so they are vulnerable to eavesdropping attacks.

Some recent works have achieved data confidentiality in in-vehicle communications [16], [17]. Samuel et al. [16] provided a security protocol for secure real-time data processing in CAN through data encryption and authentication technology. The same authors also proposed a security architecture to meet the CAN with flexible data rate (CAN-FD) standards [17]. This work achieved confidentiality, authentication, and integrity with the limited computing power of ECUs using cryptographic techniques with relatively low processing time such as symmetric key cryptography and hash functions. This work also did not provide a solution to the security issues that could occur if the high-performance node responsible for key management is corrupted by an attacker.

To increase the security level of the in-vehicle communications, a security protocol using public key-based cryptography was provided for CAN-FD and FlexRay in [18]. However, this protocol is extremely hard to be executed while driving due to the limited performance of ECUs. Therefore, it should be executed in authorized garages or manufacturers, which, is not good in terms of usability. In addition, all ECUs sharing the same secret key can have the key exposure problem, caused by ECUs corrupted by attackers. Therefore, depending on the data being shared, the key must be shared only with the ECUs that need the data.

B. Motivation and Contribution

Despite of the intensive efforts for securing vehicle communications, most of the prior works have some issues as follows. *First, the excessive computing power is required.* Complicated operations are introduced to satisfy the aforementioned security requirements [9]-[12], [18]. Generally, ECUs are unable to process the complex operations since they have lower computing capabilities than that of a typical computer. To reduce the security computation burden, it has also been proposed to mount additional components at ECUs such as hardware security modules, but this leads to additional costs. *Second, the proposed protocols are not secure enough.* Some studies tried to construct their protocols by considering the real-time constraint of in-vehicle communications. However, they ended up meeting only low-level basic security requirements using symmetric-based cryptographic schemes [9], [11], [16], [17], [19]. Therefore, for secure in-vehicle communications, it is desirable to develop new techniques that give lower computation burden to ECUs, while providing a higher level of security, which is the main objective of this paper. Therefore, this paper proposes a novel secure in-vehicle communications protocol with fine-grained access control based on edge computing, so-called EC-SVC, by exploiting the concept of edge computing to distribute the computation burden at ECUs [20].

The contribution of this paper is as follows.

- *EC-SVC* is proposed for in-vehicle communications that support data confidentiality, authentication, integrity, fine-grained access control, and policy and credential privacy.
- We exploit the concept of edge computing by introducing the security agent (SA), which handles cryptographic operations on behalf of low computing power ECUs. In addition, we propose the *enhanced attribute-based encryption with hidden policy and credential*, which achieves attribute-based access control with data confidentiality and policy privacy even against the SA.

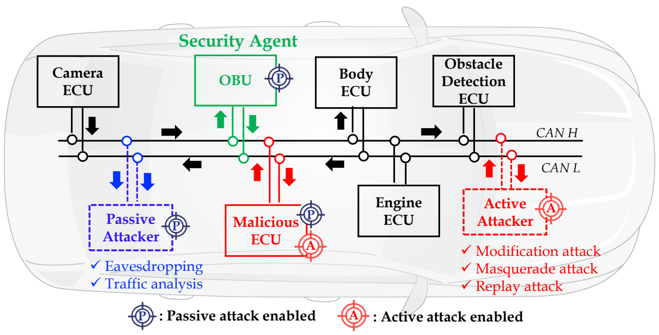


Fig. 1. Overview of the system model.

- We formally prove the security of the enhanced attribute-based encryption and the proposed protocol. Furthermore, by implementing the proposed protocol in CAN, we show the feasibility of the proposed protocol under the real-time requirements in in-vehicle communications.
- By focusing on CAN among the in-vehicle network, we have shown that our work is not limited to the in-vehicle, but is also practical and feasible in a wide range of ICSs.

The remainder of this paper is organized as follows. Sec. II describes the system model, security requirements, and security models. Sec. III introduces the system preliminary including the newly proposed cryptographic scheme. Sec. IV describes the key management model and the proposed security protocols. Sec. V provides security analysis on proposed cryptographic scheme and the proposed protocol. Sec. VI analyzes the performance of the proposed protocol. Finally, Sec. VII presents a conclusion about this work.

II. SYSTEM AND SECURITY MODELS

In this section, we describe the in-vehicle network model including the attack model and the security requirements of the network. We then introduce the security model and some definitions, which are used to analyze the security of the proposed protocol.

A. System Model and Security Requirements

1) In-vehicle Network Model: We consider the in-vehicle network, where there exist two kinds of entities, ECU and on board unit (OBU) as shown in Fig. 1. All entities connect to CAN, which is a multi-master network, and communicate through the CAN bus. Each ECU collects the data from sensors, and shares it with actuators, other ECUs, and OBU. The ECU has low computing power, so long processing time is generally required to execute the advanced cryptography that provides a high-level of security. On the other hand, the OBU, which can communicate with nodes inside and outside of the vehicle, has higher computing power compared to the ECU. We utilize the OBU as an edge computing node, which allows to perform cryptography operations, offloaded by ECUs. We call this type of OBU as the SA [21], and assume the SA can be an honest-but-curious adversary. In this system model, we also consider security issues that may occur during offloading the cryptographic works from the ECU to the SA, and describe the details of the key management model for attribute-based key exchange in Sec. IV-A.

2) Security Requirements: As shown in Fig. 1, there can be two types of attackers in in-vehicle network, the passive attacker and the active attacker. The passive attackers can be connected to the CAN bus and eavesdrop the messages in CAN. The active attacker can cause severe problems to the vehicle by modifying the data sent by the ECU and the OBU or by sending a wrong key. In addition, the active attackers may corrupt the OBU and the ECU to obtain their secrets and use them to participate in communications for obtaining private data or session keys transmitted on the CAN. By considering the aforementioned attackers, we propose the following security requirements for the in-vehicle network and analyze these security requirements in Sec. V-B.

- *Confidentiality* : All messages are broadcasted due to the characteristics of the in-vehicle communications, so an attacker can easily eavesdrop the messages. Hence, the confidentiality should be guaranteed, only the intended recipients can obtain the plaintext of the message.
- *Authentication* : An attacker may retransmit the message, which exchanged between entities in the previous session, to impersonate a legitimate device. Therefore, we need to achieve mutual authentication to ensure that ECUs and SA can authenticate each other for the legitimacy.
- *Fine-grained Access Control* : Most of the transmitted data in a vehicle are not for all ECUs, but for certain ECUs. In addition, when a node (e.g., ECU or OBU) is compromised by an attacker or some malicious nodes are connected in CAN, the transmitted message in CAN can be exposed without access control [22], [23]. Therefore, the fine-grained access control is required in in-vehicle networks.
- *Policy and Credential privacy* : A sender encrypts messages using the given policies to indicate the intended receivers according to their attributes for access control of message exchange. The encryption and decryption procedures should not expose identity information of ECUs to avoid potential analysis or attacks to disrupt normal communications [22].

B. Security Definitions of Enhanced Attribute-based Encryption with Hidden Policy and Credential

To prove the security of the proposed attribute-based encryption scheme in Sec. III, so-called enhanced attribute-based encryption with hidden policy and credential (EABEHP), which achieves partial proxy decryption, hidden credential, and hidden policy against honest-but-curious decryption proxy, this section defines the security properties of EABEHP.

Definition 1 (C-IND-CPA-RUCA): EABEHP is said to be ciphertext indistinguishability against chosen plaintext attack and restricted user coalition attack (C-IND-CPA-RUCA) if any probabilistic polynomial time (PPT) adversary $\mathcal{A}$ has only negligible advantage to distinguish the ciphertext of two given messages in the following security game.

1) **Setup Phase:** The challenger $\mathcal{C}$ sets up the EABEHP scheme and provides the attacker with all the public parameters of the system.
2) **Training Phase 1:** $\mathcal{A}$ can only corrupt either the proxy (edge device) or any user except for the target user. That is, (i) the attacker can corrupt the proxy to learn its secrets and act on its behalf or (ii) $\mathcal{A}$ has the following abilities.
    - $\mathcal{A}$ can register a new user or corrupt an honest user, who do not satisfy a policy P^* on the system, thereby $\mathcal{A}$ can learn their secrets and act on their behalves.
    - $\mathcal{A}$ can make requests of **TransformCipherText**, **Extract**, and **ProxyDecrypt1** to the proxy.
    - $\mathcal{A}$ can make requests of **TimeKeyGen**, **TransformUserKey**, and **Shuffle** to honest users.
3) **Challenge Phase:** $\mathcal{A}$ outputs two messages M_0 and M_1 of equal length, and a policy P^* under the following restriction.
   Restriction 1: None of the corrupted users in **Training Phase 1** satisfy P^*.
   $\mathcal{C}$ then flips the random coin $b \overset{R}{\in} \{0,1\}$ and generates C^* by encrypting the message M_b under the policy P^* by the **Encrypt** algorithm according to b. $\mathcal{C}$ returns C^* to $\mathcal{A}$.
4) **Training Phase 2:** $\mathcal{A}$ can perform the operations defined in **Training Phase 1**, except that none of the corrupted users can satisfy the policy P^*.
5) **Guessing Phase:** $\mathcal{A}$ outputs a guessing $b' \overset{R}{\in} \{0,1\}$. $\mathcal{A}$ wins the game if $b' = b$.

Definition 2 (P-IND-CPA-UCA): EABEHP is said to be policy indistinguishability against chosen plaintext attack and user coalition attack (P-IND-CPA-UCA) if all PPT adversaries only have negligible advantage to distinguish the ciphertext of two given policies in the following security game.

1) **Setup Phase:** Same as that in the Definition 1.
2) **Training Phase 1:** Same as that in the Definition 1.
3) **Challenge Phase:** $\mathcal{A}$ sends a chosen message M^* and two chosen policies, P_0 and P_1, for the encryption of M^* under the following restriction.
   Restriction 2: All the corrupted users satisfy none of the policies, P_0 and P_1, or they all satisfy both policies. $\mathcal{C}$ selects a random bit $b \in \{0,1\}$ and encrypts M^* with the given P_b to generate C^* according to b.
4) **Training Phase 2:** Same as that in the **Training Phase 1**.
5) **Guessing Phase:** Same as that in the Definition 1.

Definition 3 (Credential Privacy): EABEHP is said to support credential privacy if all PPT adversaries only have negligible advantage to distinguish the real credential of the specified user from the credentials of the other users with non-negligible advantages as the following game.

1) **Setup Phase:** Same as that in the Definition 1.
2) **Training Phase 1:** Same as that in the Definition 1.
3) **Challenge Phase:** $\mathcal{A}$ outputs the credentials of two users, a selected policy P^*, and a selected message M.

$\mathcal{C}$ then outputs a ciphertext of M with P^* and SK_{U_b} according to $b \overset{R}{\in} \{0,1\}$, where the associated attributes of SK_{U_1} and SK_{U_2} either both satisfy P^* or do not satisfy P^*.

4) **Training Phase 2:** Same as that in the **Training Phase 1**, except that the corruption of the users possessing SK_{U_1} or SK_{U_2} is not allowed.
5) **Guessing Phase:** Same as that in the Definition 1.

C. Security Definitions of Secure In-Vehicle Communications with Access Control

We capture the capabilities of attackers by following definitions in the system model. We first explain the notation used in the security model. The proposed protocol is called Γ, and we regard the communication between two users A and B in communication sessions at t_1 and t_2 as $\Gamma^{t_1}_{A,B}$ and $\Gamma^{t_2}_{B,A}$, respectively. We describe oracles, propose attackers who can query these oracles, and define the security of the proposed protocols according to security requirements.

The oracles used to capture the attacker's capabilities are as follows.

- **Execute**($\Gamma^{t_1}_{A,B}$, $\Gamma^{t_2}_{B,A}$) : This oracle models all kinds of passive attackers that can eavesdrop all data between $\Gamma^{t_1}_{A,B}$ and $\Gamma^{t_2}_{B,A}$.
- **Send**($\Gamma^{t_1}_{A,B}$, M) : This oracle models an active attacker that sends a message M to $\Gamma^{t_1}_{A,B}$.
- **Expose**($\Gamma^{t_1}_{A,B}$) : This oracle models the exposure of the session key of A, shared with B, at communication session t_1.
- **Corrupt**($\Gamma^{t_1}_{A,B}$) : This oracle models the exposure of the long-term secret key of A, shared with B, at communication session t_1.
- **Test**($\Gamma^{t_1}_{A,B}$): This oracle models the test of session key security. When one queries this oracle and both parties, which are the partners of each other, in the protocol are accepted, it will return a real session key or a random string depending on a random bit. Otherwise, it returns an invalid output.
- **TestPolicy**(P_0, P_1, M) This oracle models to test the privacy of the given policies for encryption. When one queries this oracle with the inputs of two given policies, P_0 and P_1, and a message M, it will output an encryption on M with P_b according to the randomly selected $b \in \{0,1\}$.
- **TestCert**($\Gamma^{t_1}_{A,B}, P^*$): This oracle models to test the privacy of user's credential. When one queries this oracle with the input of $\Gamma^{t_1}_{A,B}$ and the target policy P^*, it will output either the credential of $\Gamma^{t_1}_{A,B}$ or a randomly selected credential with the restriction that the attributes of both credentials satisfy P^* or do not satisfy P^*.

We also define the security properties of the proposed EC-SVC according to the security requirements discussed in Sec. II-A as follows.

Definition 4 (Mutual Authentication): We assume that S simulates $\Gamma^{t_1}_{A,B}$ and $\Gamma^{t_2}_{B,A}$, and interacts with $\mathcal{A}$, who can query polynomial number of **Execute** and **Send** oracles. After $\mathcal{A}$ queries these oracles, it sends a message to be accepted by $\Gamma^{t_1}_{A,B}$ or $\Gamma^{t_2}_{B,A}$, where $\Gamma^{t_1}_{A,B}$ or $\Gamma^{t_2}_{B,A}$ has not accepted each other. $\mathcal{A}$ has the following advantage

$$\mathrm{Adv}^{\mathrm{MuAuth}}_{\mathcal{A}} = \Pr[\mathcal{A} \text{ accepted by } \Gamma^{t_1}_{A,B} \text{ or } \Gamma^{t_2}_{B,A}]. \tag{1}$$

The mutual authentication between A and B is guaranteed if $\mathrm{Adv}^{\mathrm{MuAuth}}_{\mathcal{A}}$ is negligible.

Definition 5 (Attribute-based Key Exchange): There are S simulating $\Gamma^{t_1}_{A,B}$ and $\Gamma^{t_2}_{B,A}$, and $\mathcal{A}$, who can query polynomial number of **Execute** and **Send** oracles in polynomial time. After $\mathcal{A}$ queries these oracles, if $\Gamma^{t_1}_{A,B}$ and $\Gamma^{t_2}_{B,A}$ are accepted by each other with a session key K, $\mathcal{A}$ queries **Test** to obtain a session key K or a random string according to a random bit $b \in \{0,1\}$. $\mathcal{A}$ has the following advantages

$$\mathrm{Adv}^{\mathrm{AKE}}_{\mathcal{A}} = \Pr[\mathrm{Succ}^{\mathrm{AKE}}_{\mathcal{A}}] \ - \ 1/2, \tag{2}$$

where $\mathrm{Succ}^{\mathrm{AKE}}_{\mathcal{A}}$ is the event that $\mathcal{A}$ outputs a guess $b' = b$. If $\mathrm{Adv}^{\mathrm{AKE}}_{\mathcal{A}}$ is negligible, the attribute-based key exchange security is achieved.

Definition 6 (Policy Privacy): S simulates $\Gamma^{t_1}_{A,B}$ and $\Gamma^{t_2}_{B,A}$, and interacts with $\mathcal{A}$, who can query polynomial number of **Execute** and **Send** oracles in polynomial time. After this phase, $\mathcal{A}$ queries **TestPolicy** with message M and two valid policies, P_0 and P_1, as the input to obtain a C_0 or C_1 according to a random bit $b \in \{0,1\}$, where C_0 and C_1 are encryption for M with P_0 and P_1, respectively. $\mathcal{A}$ has the following advantages

$$\mathrm{Adv}^{\mathrm{PP}}_{\mathcal{A}} = \Pr[\mathrm{Succ}^{\mathrm{PP}}_{\mathcal{A}}] \ - \ 1/2, \tag{3}$$

where $\mathrm{Succ}^{\mathrm{PP}}_{\mathcal{A}}$ is the event that $\mathcal{A}$ outputs a guess $b' = b$. If $\mathrm{Adv}^{\mathrm{PP}}_{\mathcal{A}}$ is negligible, the policy privacy is achieved.

Definition 7 (Credential Privacy): S simulates $\Gamma^{t_1}_{A,B}$ and $\Gamma^{t_2}_{B,A}$, and interacts with $\mathcal{A}$, who can query polynomial number of **Execute** and **Send** at polynomial time. After this phase, $\mathcal{A}$ queries **TestCert** with target policy P^* as the input to obtain a credential of $\Gamma^{t_1}_{A,B}$ or a randomly selected credential according to a random bit $b \in \{0,1\}$, where the attributes of both credentials satisfy P^* or do not satisfy P^*. $\mathcal{A}$ has the following advantages

$$\mathrm{Adv}^{\mathrm{CP}}_{\mathcal{A}} = \Pr[\mathrm{Succ}^{\mathrm{CP}}_{\mathcal{A}}] \ - \ 1/2, \tag{4}$$

where $\mathrm{Succ}^{\mathrm{CP}}_{\mathcal{A}}$ is the event that $\mathcal{A}$ outputs a guess $b' = b$. If $\mathrm{Adv}^{\mathrm{CP}}_{\mathcal{A}}$ is negligible, the credential privacy is achieved.

III. SYSTEM PRELIMINARIES

This section introduces the required background for the proposed protocol including the cryptographic algorithms used in the protocol.

A. Pseudorandom Function and Permutations

We describe the hash function, e.g., SHA-256, and the symmetric encryption, e.g., AES128, used in this protocol as pseudorandom function (PRF) and pseudorandom permutation (PRP) [24], respectively. First, the PRF [25], [26] defined over (K, X, Y) is an efficient and deterministic function, which returns a pseudorandom output sequence

$$H : K_H \times X \to Y, \tag{5}$$

where K_H is the key space, $X \subseteq \{0,1\}^{l_1}$ is the input space, $Y \subseteq \{0,1\}^{l_2}$, and $l_1 > l_2$. The PRP [27] defined over (K, X) is an efficient and deterministic function which returns a pseudorandom output sequence

$$E : K_E \times X \rightarrow X', \tag{6}$$

where K_E is the key space, $X \subseteq \{0,1\}^{l}$ is the input space, and $X' \subseteq \{0,1\}^{l}$. There is an efficient inversion algorithm $D(K_E, X')$ for this PRP. We additionally describe the shuffle function, which is a kind of pseudorandom permutation to be used in the EABEHP scheme. The shuffle function has the same input and output space, which returns a pseudorandom output sequence

$$SH : K_{SH} \times X \rightarrow X, \tag{7}$$

where K_{SH} is the key space, and $X \subseteq \{0,1\}^{l_N}$ is the input and output space. There is an efficient inversion algorithm $SH^{-1}(K_{SH}, X)$ for this shuffle function.

B. Enhanced Attribute-based Encryption with Hidden Policy and Credential

1) Intuition: The proposed EABEHP is an enhanced scheme of the privacy-enhanced attribute-based publishing of data (PEAPOD) scheme in [22]. In order to leverage edge computing resources, we use the concept of the security agent (SA), which is the device with more powerful computational resource and performs the cryptographic operations, offloaded by resource-limited devices [20]. Since we consider the SA as an honest-but-curious, the EABEHP needs to satisfy two additional security properties: 1) confidentiality against SAs for proxy decryption, and 2) hidden attributes and policies against SAs.

First, to achieve the confidentiality of encrypted message against the proxy, who performs **ProxyDecrypt1** with the given ciphertext, sC'_r and user secret keys, $(AK_{U_r},\ RK_{U_r})$, sC'_r is shuffled by the shuffle function, **Shuffle**, which takes the time key ω_k of each time slot k. Since ω_k is unknown to the proxy, the proxy cannot recover the original ciphertext C from the shuffled sC'_r and decrypt the ciphertext successfully.

Second, to conceal the policies of encryption from the proxy and outsiders, the policy is used to decide to encrypt a message tuple, which is the divided partial message, i.e., k_i, a random value α_i, or 1, depending on that the i-th attribute is required, unrequired, or irrelevant specified in the policy. Thus, when any attacker wants to distinguish if a ciphertext is encrypted with which one of two given policies, where one of them will be randomly selected as the policy of the encryption, the attacker has to decrypt the encrypted message tuple first. Otherwise, all the encrypted message tuples of the ciphertext are considered as random variables. The attacker will not learn any policy information if the confidentiality of the message tuples based on the ElGamal encryption is guaranteed.

Third, since the proxy for partial decryption process, i.e., **ProxyDecrypt1**, needs to know the attribute indices of each receiver, it may expose user attribute information. Thus, a permuted attribute indices, $\hat{I}_r$, for the receiver r will be given to the proxy for partial decryption. Since **ProxyDecrypt1** takes $\hat{I}_r$ for the partial decryption proceess, the output will remain $p_i \times g^{r_j(s_i - s'_i)}$ for each message tuple, where i is the inverse permuted attribute index and i' is permuted i using a SH_{ω_k}. Thus, the receiver, who knows ω_k, can calculate $\hat{I}_r$ and use the tuples, $AM_{r_j} = \{g^{r_j s_i}\}_{i \in I}$, for the receiver to cancel $g^{r_j(s_i - s'_i)}$ to recover each message tuple p_i during the decryption procedure by **ProxyDecrypt2**.

The proposed scheme consists of ten algorithms.

- **Setup**(1^{λ}): This algorithm chooses the cyclic group $\mathbb{G}$ of prime order p with a generator g. Next, it chooses a large prime number q such that $q|(p-1)$ and random numbers $\{a_i\}_{i \in I}$ for all attributes of the system, where $I = \{1, 2, 3, \ldots, N\}$ is the universal set of attribute indices of the system, and N is the number of system attributes. After that, the algorithm generates the master public key, MPK, of the system as
$$MPK = \{g,\ p,\ q,\ \{PK_{S_i} = g^{a_i}\}_{i \in I}\},$$
where $a_i \in \mathbb{Z}_q^*$. Then, it randomly chooses a master secret key $MSK = K_S$ and generates a transformation secret key $TK = \{TK_{S_i} = s_i\}_{i \in I}$ where $a_i + s_i = K_S$ for all $i \in I$. Finally, it generates a group key K_{group} for all users in the system. The algorithm then outputs MSK, MPK, TK, and K_{group}.
- **KeyGen**(MSK, ID_j, I_j): This algorithm generates user attribute keys, $SK_{U_j} = \{SK_{U_{j,i}} = a_{j,i}\}_{i \in I_j}$, where $I_j \subseteq I$ is the set of attributes indices of user j. Next, it generates a re-encryption key $RK_{U_j} = \sum_{i \in I_j} s_{j,i}$ for all attribute indices of the user j where $a_{j,i} + s_{j,i} = K_S$. This algorithm outputs SK_{U_j} and RK_{U_j} for user j.
- **TimeKeyGen**(K_{group}, t_k): This algorithms takes K_{group} and the time slot t_k as inputs, and generates $\omega_k = H(r_k || K_{\text{group}})$ as the output, where r_k is randomly selected and distinct for different t_k.
- **TransformUserKey**(ω_k, SK_{U_j}): This algorithm takes ω_k and SK_{U_j} as inputs, and outputs $AK_{U_j} = \sum_{i \in I_j} a_{j,i} + \omega_k$ as the transformed user attribute key.
- **Encrypt**(MPK, T, ω_k, M): This algorithm first takes MPK, T, ω_k, and M as inputs, and outputs C as the ciphertext of M. Here, $T = \{t_i\}_{i \in I}$ is a policy set, where $t_i = 1$ if the attribute i is required, $t_i = 0$ if the attribute i is irrelevant, and $t_i = -1$ if the attribute i is unrequired, and $M \in \mathbb{Z}_q$ is the message to be encrypted. Then, it generates the message tuples depending on each $i \in I$ as
$$p_i = \begin{cases} k_i & \text{if } t_i = 1 \\ 1 & \text{if } t_i = 0 \\ \alpha_i & \text{if } t_i = -1 \end{cases},$$
$$\text{such that} \prod_{\substack{t_i \in T \wedge t_i = 1, \\ \forall i \in I}} p_i \equiv M \ (mod\ q),$$
for randomly selected $\alpha_i \in \mathbb{Z}_q$. Next, it randomly selects $r_j \in \mathbb{Z}_q$ and encrypts each message tuple, p_i, with $PK_{S_i} = g^{a_i}$ as
$$C = \{A, \langle B_i \rangle_{i \in I}, D\} = \{g^{r_j}, \langle p_i (g^{a_i})^{r_j} \rangle_{i \in I}, (g^{r_j})^{\omega_k}\},$$
where $\langle \cdot \rangle$ means a sequence.

- **Shuffle**(C, ω_k): This algorithm permutes the order of tuples by a pseudorandom permutation as

$$\begin{aligned} sC &= \{A, \langle \widehat{B}_i \rangle_{i \in I}, D\} \\ &= \{g^{r_j}, \langle \widehat{B}_i = p_{i'}(g^{a_{i'}})^{r_j} \rangle_{i \in I \wedge i' = SH_{\omega_k}(i)}, (g^{r_j})^{\omega_k}\}, \end{aligned}$$

  where $i' = SH_{\omega_k}(i)$, is a shuffle function, which takes ω_k and $i \in I$ as inputs and outputs $i' \in I$.
- **TransformCipherText**(sC, TK): This algorithm use TK to transform sC as

$$\begin{aligned} sC' &= \{A', \langle B'_i \rangle_{i \in I}, D'\} = \{A, \langle \widehat{B}_i A^{s_i} \rangle_{i \in I}, D\} \\ &= \{g^{r_j}, \langle p_{i'} g^{a_{i'} r_j + s_i r_j} \rangle_{i \in I \wedge i' = SH_{\omega_k}(i)}, (g^{r_j})^{\omega_k}\}. \end{aligned}$$

- **Extract**($sC', \hat{I}_r$): This algorithm takes sC' and $\hat{I}_r$ as inputs and outputs the extracted ciphertext sC'_r. Here, $\hat{I}_r$ is a set of elements obtained by inversely permuting each element $i \in I_r$ such as $SH^{-1}_{\omega_k}(i)$. It extracts sC'_r from sC' according to $\hat{I}_r$ as

$$\begin{aligned} sC'_r &= \{A'_r, B'_r, D'_r\} = \{A', \prod_{i \in \hat{I}_r} B'_i, D'\} \\ &= \{A, \prod_{i \in \hat{I}_r} \widehat{B}_i A^{s_i}, D\} \\ &= \{g^{r_j}, \prod_{\substack{i \in \hat{I}_r \wedge \\ i' = SH_{\omega_k}(i)}} p_{i'} g^{a_{i'} r_j + s_i r_j}, (g^{r_j})^{\omega_k}\}. \end{aligned}$$

- **ProxyDecrypt1**($sC'_r, AK_{U_r}, RK_{U_r}, TK$): This algorithm takes sC'_r, AK_{U_r}, and RK_{U_r}, and TK as inputs, and outputs the partial decrypted ciphertext sC''_r and decryption materials AM_{r_j} as

$$\begin{aligned} sC''_r &= D'_r \cdot B'_r / (A'_r)^{AK_{U_r} + RK_{U_r}} \\ &= \Bigg(\prod_{\substack{i \in \hat{I}_r \wedge \\ i' = SH_{\omega_k}(i)}} p_{i'} \Bigg) (g^{r_j})^{\sum_1 s_i - s_{i'}}, \\ AM_{r_j} &= \{(A'_r)^{s_i}\}_{i \in I} = \{g^{r_j s_i}\}_{i \in I}, \end{aligned}$$

  where N_r is the number of attributes of user r, $\sum_1 = \sum_{i \in \hat{I}_r \wedge i' = SH_{\omega_k}(i)}$.
- **ProxyDecrypt2**(sC''_r, AM_{r_j}, I_r): This algorithm takes sC''_r, AM_{r_j}, and I_r as inputs, and outputs M as

$$M = sC''_r \cdot \Bigg(\prod_{i \in \hat{I}_r} g^{r_j s_i} \Bigg) \Bigg/ \Bigg(\prod_{i \in \hat{I}_r} g^{r_j s_i} \Bigg).$$

IV. PROPOSED SECURE IN-VEHICLE COMMUNICATIONS WITH FINE-GRAINED ACCESS CONTROL BASED ON EDGE COMPUTING (EC-SVC)

This section present, the proposed EC-SVC protocol including the key management and the in-vehicle security protocol. In the key management, we present the method to install the symmetric keys as well as the keys in the EABEHP scheme on each device. In addition, we present the in-vehicle security protocol that satisfies the security requirements presented in Sec. II-A. This protocol involves a sender-ECU, receiver-ECUs, and SA, and includes the authentication process between each device, and the EABEHP scheme. We guarantee the security of the protocol by using nonce, signature, symmetric key cryptography, hash function, and EABEHP scheme properly. We first describe the key management and then the in-vehicle security protocol.

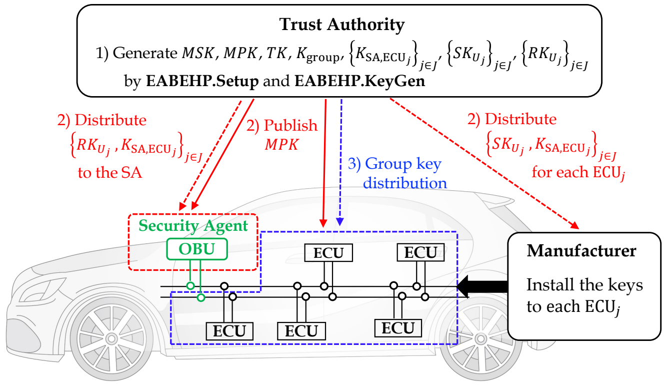

Trust Authority
1) Generate $MSK, MPK, TK, K_{\text{group}}, \{K_{\text{SA,ECU}_j}\}_{j\in J}, \{SK_{U_j}\}_{j\in J}, \{RK_{U_j}\}_{j\in J}$
by EABEHP.Setup and EABEHP.KeyGen
2) Distribute $\{RK_{U_j}, K_{\text{SA,ECU}_j}\}_{j\in J}$ to the SA
2) Publish MPK
3) Group key distribution
2) Distribute $\{SK_{U_j}, K_{\text{SA,ECU}_j}\}_{j\in J}$ for each ECU$_j$
Security Agent
OBU
ECU
ECU
ECU
ECU
ECU
Manufacturer
Install the keys to each ECU$_j$

Fig. 2. Long-term secret key management

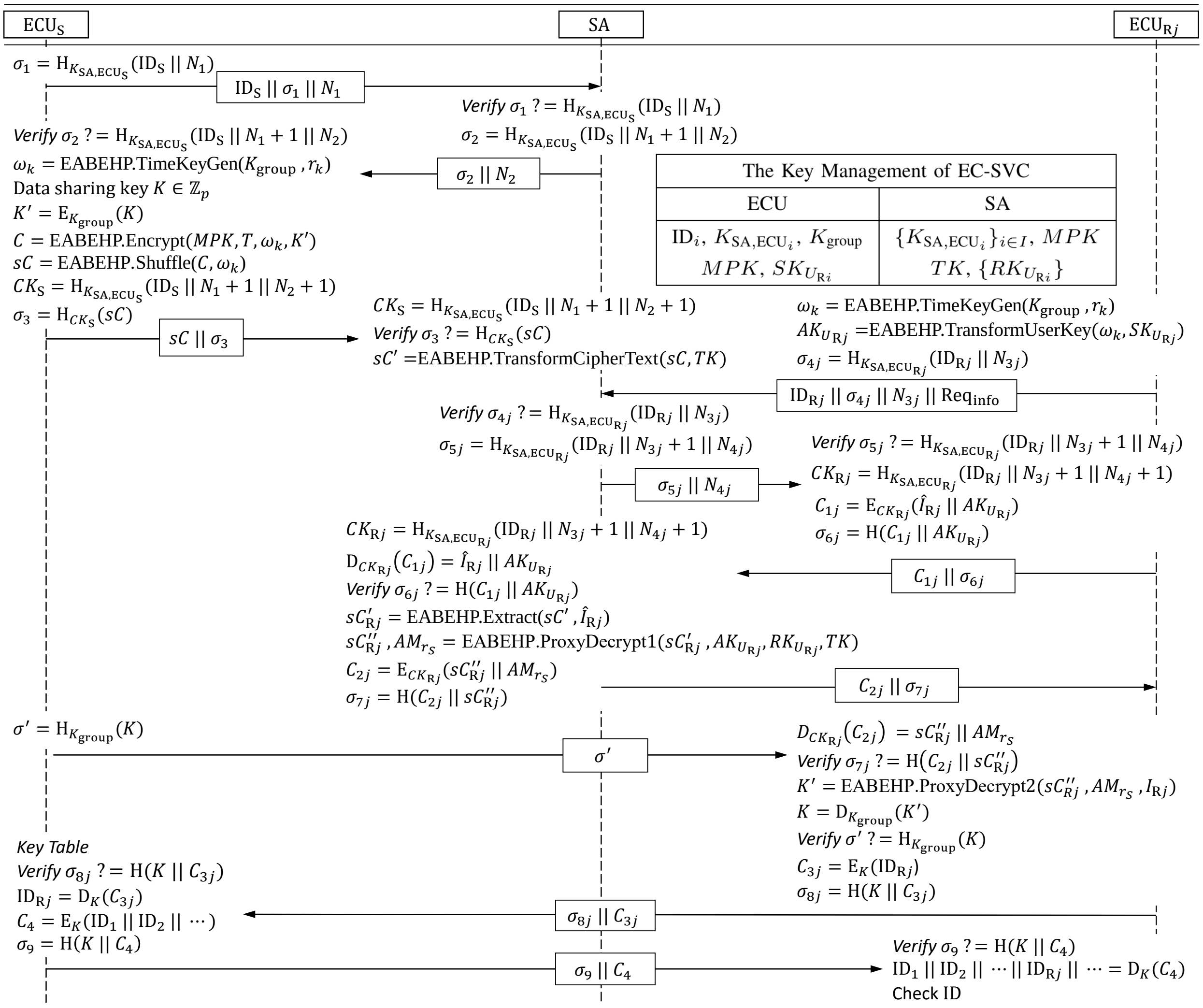
A. Key Management

Figure 2 shows the key management of EC-SVC. The trust authority (TA) issues the required cryptographic keys for all entities in the system by the following procedures: (1) TA generates MSK, MPK, TK, K_{group}, and $\{K_{\text{SA,ECU}_j}\}_{j \in J}$ by **EABEHP.Setup**. It then generates SK_{U_j} and RK_{U_j} for each ECU$_j$ by **EABEHP.KeyGen**. (2) TA publishes MPK and keeps MSK secretly. It then distributes $\{SK_{U_j}, K_{\text{SA,ECU}_j}\}_{j \in J}$ to each user j, and RK_{U_j}, TK and all $\{K_{\text{SA,ECU}_j}\}_{j \in J}$ to the SA. (3) The TA sends the group key K_{group} securely to each ECU by executing the group key distribution mechanism [28]. Assume that the size of the security keys is sufficient to provide the security of the system for the life-time of a vehicle.

B. In-vehicle Security Protocol

This subsection describes the in-vehicle security protocol, which is the edge computing-based in-vehicle authenticated key exchange protocol with attribute-based access control in Fig. 3. The proposed protocol is executed when a vehicle is started and consists of the following twelve steps.

1) The sender-ECU(ECU$_\text{S}$) generates nonce $N_1 \in \{0,1\}^L$ and $\sigma_1 = \text{H}_{K_{\text{SA,ECU}_\text{S}}}(\text{ID}_\text{S} || N_1)$. The ECU$_\text{S}$ then send $\{\text{ID}_\text{S} || \sigma_1 || N_1\}$ to the SA.
2) The SA verifies σ_1 and generates nonce $N_2 \in \{0,1\}^L$. The SA then generates $\sigma_2 = \text{H}_{K_{\text{SA,ECU}_\text{S}}}(\text{ID}_\text{S} || N_1 + 1 || N_2)$ and sends $\{\sigma_2 || N_2\}$ to the ECU$_\text{S}$.
3) The ECU$_\text{S}$ verifies σ_2. If passed, the ECU$_\text{S}$ executes ω_k = **EABEHP.TimeKeyGen**(K_{group}, r_k). The ECU$_\text{S}$ generates data sharing key $K \in \mathbb{Z}_q$ and encrypts K as $K' = \text{E}_{K_{\text{group}}}(K)$. It then computes C = **EABEHP.Encrypt**(MPK, T, ω_k, K'). After that, the ECU$_\text{S}$ computes sC = **EABEHP.Shuffle**(C, ω_k), and generates $CK_\text{S} = \text{H}_{K_{\text{SA,ECU}_\text{S}}}(\text{ID}_\text{S} || N_1 + 1 || N_2 + 1)$ and $\sigma_3 = \text{H}_{CK_\text{S}}(sC)$. Subsequently, the ECU$_\text{S}$ send $\{sC || \sigma_3\}$ to the SA.
4) The SA generates $CK_\text{S} = \text{H}_{K_{\text{SA,ECU}_\text{S}}}(\text{ID}_\text{S} || N_1 + 1 || N_2 + 1)$ and verifies σ_3. It then computes sC' =



$\mathrm{ECU_S}$
SA
$\mathrm{ECU}_{\mathrm{R}j}$
$\sigma_1 = \mathrm{H}_{K_{\mathrm{SA,ECU_S}}}(\mathrm{ID_S} \,||\, N_1)$
$\mathrm{ID_S} \,||\, \sigma_1 \,||\, N_1$
Verify $\sigma_1 \;?= \mathrm{H}_{K_{\mathrm{SA,ECU_S}}}(\mathrm{ID_S} \,||\, N_1)$
$\sigma_2 = \mathrm{H}_{K_{\mathrm{SA,ECU_S}}}(\mathrm{ID_S} \,||\, N_1 + 1 \,||\, N_2)$
Verify $\sigma_2 \;?= \mathrm{H}_{K_{\mathrm{SA,ECU_S}}}(\mathrm{ID_S} \,||\, N_1 + 1 \,||\, N_2)$
$\omega_k = \mathrm{EABEHP.TimeKeyGen}(K_{\mathrm{group}}, r_k)$
$\sigma_2 \,||\, N_2$
Data sharing key $K \in \mathbb{Z}_p$
$K' = \mathrm{E}_{K_{\mathrm{group}}}(K)$
$C = \mathrm{EABEHP.Encrypt}(MPK, T, \omega_k, K')$
$sC = \mathrm{EABEHP.Shuffle}(C, \omega_k)$
$CK_{\mathrm{S}} = \mathrm{H}_{K_{\mathrm{SA,ECU_S}}}(\mathrm{ID_S} \,||\, N_1 + 1 \,||\, N_2 + 1)$
$\sigma_3 = \mathrm{H}_{CK_{\mathrm{S}}}(sC)$
$sC \,||\, \sigma_3$
$CK_{\mathrm{S}} = \mathrm{H}_{K_{\mathrm{SA,ECU_S}}}(\mathrm{ID_S} \,||\, N_1 + 1 \,||\, N_2 + 1)$
Verify $\sigma_3 \;?= \mathrm{H}_{CK_{\mathrm{S}}}(sC)$
$sC' = \mathrm{EABEHP.TransformCipherText}(sC, TK)$
$\omega_k = \mathrm{EABEHP.TimeKeyGen}(K_{\mathrm{group}}, r_k)$
$AK_{U_{\mathrm{R}j}} = \mathrm{EABEHP.TransformUserKey}(\omega_k, SK_{U_{\mathrm{R}j}})$
$\sigma_{4j} = \mathrm{H}_{K_{\mathrm{SA,ECU_{R\mathit{j}}}}}(\mathrm{ID}_{\mathrm{R}j} \,||\, N_{3j})$
$\mathrm{ID}_{\mathrm{R}j} \,||\, \sigma_{4j} \,||\, N_{3j} \,||\, \mathrm{Req_{info}}$
Verify $\sigma_{4j} \;?= \mathrm{H}_{K_{\mathrm{SA,ECU_{R\mathit{j}}}}}(\mathrm{ID}_{\mathrm{R}j} \,||\, N_{3j})$
$\sigma_{5j} = \mathrm{H}_{K_{\mathrm{SA,ECU_{R\mathit{j}}}}}(\mathrm{ID}_{\mathrm{R}j} \,||\, N_{3j} + 1 \,||\, N_{4j})$
Verify $\sigma_{5j} \;?= \mathrm{H}_{K_{\mathrm{SA,ECU_{R\mathit{j}}}}}(\mathrm{ID}_{\mathrm{R}j} \,||\, N_{3j} + 1 \,||\, N_{4j})$
$CK_{\mathrm{R}j} = \mathrm{H}_{K_{\mathrm{SA,ECU_{R\mathit{j}}}}}(\mathrm{ID}_{\mathrm{R}j} \,||\, N_{3j} + 1 \,||\, N_{4j} + 1)$
$\sigma_{5j} \,||\, N_{4j}$
$C_{1j} = \mathrm{E}_{CK_{\mathrm{R}j}}(\hat{I}_{\mathrm{R}j} \,||\, AK_{U_{\mathrm{R}j}})$
$\sigma_{6j} = \mathrm{H}(C_{1j} \,||\, AK_{U_{\mathrm{R}j}})$
$CK_{\mathrm{R}j} = \mathrm{H}_{K_{\mathrm{SA,ECU_{R\mathit{j}}}}}(\mathrm{ID}_{\mathrm{R}j} \,||\, N_{3j} + 1 \,||\, N_{4j} + 1)$
$\mathrm{D}_{CK_{\mathrm{R}j}}(C_{1j}) = \hat{I}_{\mathrm{R}j} \,||\, AK_{U_{\mathrm{R}j}}$
Verify $\sigma_{6j} \;?= \mathrm{H}(C_{1j} \,||\, AK_{U_{\mathrm{R}j}})$
$C_{1j} \,||\, \sigma_{6j}$
$sC'_{\mathrm{R}j} = \mathrm{EABEHP.Extract}(sC', \hat{I}_{\mathrm{R}j})$
$sC''_{\mathrm{R}j}, AM_{r_S} = \mathrm{EABEHP.ProxyDecrypt1}(sC'_{\mathrm{R}j}, AK_{U_{\mathrm{R}j}}, RK_{U_{\mathrm{R}j}}, TK)$
$C_{2j} = \mathrm{E}_{CK_{\mathrm{R}j}}(sC''_{\mathrm{R}j} \,||\, AM_{r_S})$
$\sigma_{7j} = \mathrm{H}(C_{2j} \,||\, sC''_{\mathrm{R}j})$
$C_{2j} \,||\, \sigma_{7j}$
$\sigma' = \mathrm{H}_{K_{\mathrm{group}}}(K)$
σ'
$D_{CK_{\mathrm{R}j}}(C_{2j}) = sC''_{\mathrm{R}j} \,||\, AM_{r_S}$
Verify $\sigma_{7j} \;?= \mathrm{H}(C_{2j} \,||\, sC''_{\mathrm{R}j})$
$K' = \mathrm{EABEHP.ProxyDecrypt2}(sC''_{\mathrm{R}j}, AM_{r_S}, I_{\mathrm{R}j})$
$K = \mathrm{D}_{K_{\mathrm{group}}}(K')$
Verify $\sigma' \;?= \mathrm{H}_{K_{\mathrm{group}}}(K)$
$C_{3j} = \mathrm{E}_K(\mathrm{ID}_{\mathrm{R}j})$
$\sigma_{8j} = \mathrm{H}(K \,||\, C_{3j})$
Key Table
Verify $\sigma_{8j} \;?= \mathrm{H}(K \,||\, C_{3j})$
$\mathrm{ID}_{\mathrm{R}j} = \mathrm{D}_K(C_{3j})$
$C_4 = \mathrm{E}_K(\mathrm{ID}_1 \,||\, \mathrm{ID}_2 \,||\, \cdots)$
$\sigma_9 = \mathrm{H}(K \,||\, C_4)$
$\sigma_{8j} \,||\, C_{3j}$
$\sigma_9 \,||\, C_4$
Verify $\sigma_9 \;?= \mathrm{H}(K \,||\, C_4)$
$\mathrm{ID}_1 \,||\, \mathrm{ID}_2 \,||\, \cdots \,||\, \mathrm{ID}_{\mathrm{R}j} \,||\, \cdots = \mathrm{D}_K(C_4)$
Check ID

The Key Management of EC-SVC

ECU	SA
ID_i, $K_{\mathrm{SA,ECU}_i}$, K_{group} MPK, $SK_{U_{\mathrm{R}i}}$	$\{K_{\mathrm{SA,ECU}_i}\}_{i \in I}$, MPK TK, $\{RK_{U_{\mathrm{R}i}}\}$

Fig. 3. Edge computing-based in-vehicle authenticated key exchange protocol with attribute-based access control

EABEHP.TransformCipherText(sC, TK). After that, the SA waits for the request message from the $\mathrm{ECU}_{\mathrm{R}j}$.

5) When the SA needs to retrieve the data, sent by $\mathrm{ECU}_{\mathrm{R}j}$, it computes ω_k = **EABEHP.TimeKeyGen**(K_{group}, r_k) and $A_{\mathrm{R}j}$ = **EABEHP.TransformUserKey**$(\omega_k, SK_{U_{\mathrm{R}j}})$. The $\mathrm{ECU}_{\mathrm{R}j}$ then generates nonce $N_{3j} \in \{0,1\}^L$ and $\sigma_{4j} = \mathrm{H}_{K_{\mathrm{SA,ECU_{R\mathit{j}}}}}(\mathrm{ID}_{\mathrm{R}j}||N_{3j})$. Subsequently, the $\mathrm{ECU}_{\mathrm{R}j}$ sends $\{ID_{\mathrm{R}j}||\sigma_{4j}||N_{3j}||\mathrm{Req_{info}}\}$ to the SA, where $\mathrm{Req_{info}}$ is the message of requesting a process to obtain a data sharing key K.
6) The SA verifies σ_{4j} first. If passed, the SA generates nonce $N_{4j} \in \{0,1\}^L$ and $\sigma_{5j} = \mathrm{H}_{K_{\mathrm{SA,ECU_{R\mathit{j}}}}}(\mathrm{ID}_{\mathrm{R}j}||N_{3j} + 1||N_{4j})$. The SA then sends $\{\sigma_{5j}||N_{4j}\}$ to the $\mathrm{ECU}_{\mathrm{R}j}$.
7) The $\mathrm{ECU}_{\mathrm{R}j}$ verifies σ_{5j} and generates $CK_{\mathrm{R}j} = \mathrm{H}_{K_{\mathrm{SA,ECU_{R\mathit{j}}}}}(\mathrm{ID}_{\mathrm{R}j}||N_{3j} + 1||N_{4j} + 1)$. The $\mathrm{ECU}_{\mathrm{R}j}$ encrypts $\{\hat{I}_{\mathrm{R}j}||AK_{U_{\mathrm{R}j}}\}$ with the generated $CK_{\mathrm{R}j}$ as $C_{1j} = \mathrm{E}_{CK_{\mathrm{R}j}}(\hat{I}_{\mathrm{R}j}||AK_{U_{\mathrm{R}j}})$ and generates $\sigma_{6j} = \mathrm{H}(C_{1j}||AK_{U_{\mathrm{R}j}})$. Then, the $\mathrm{ECU}_{\mathrm{R}j}$ sends $\{C_{1j}||\sigma_{6j}\}$ to the SA.
8) After the SA receives $\{C_{1j}||\sigma_{6j}\}$, the SA generates $CK_{\mathrm{R}j} = \mathrm{H}_{K_{\mathrm{SA,ECU_{R\mathit{j}}}}}(\mathrm{ID}_{\mathrm{R}j}||N_{3j} + 1||N_{4j} + 1)$ and then decrypts C_{1j} by $CK_{\mathrm{R}j}$ to obtain the $\{\hat{I}_{\mathrm{R}j}||AK_{U_{\mathrm{R}j}}\}$. The SA verifies σ_{6j}. The SA then computes $sC'_{\mathrm{R}j}$ = **EABEHP.Extract**$(sC', \hat{I}_{\mathrm{R}j})$ and $(sC''_{\mathrm{R}j}, AM_{r_\mathrm{S}})$ = **EABEHP.ProxyDecrypt1**$(sC'_{\mathrm{R}j}, AK_{U_{\mathrm{R}j}}, RK_{U_{\mathrm{R}j}}, TK)$, where r_S is a random number generated by $\mathrm{ECU_S}$ during the **EABEHP.Encrypt** process. The SA then encrypts $\{sC''_{\mathrm{R}j}||AM_{r_\mathrm{S}}\}$ with $CK_{\mathrm{R}j}$ to generate C_{2j}, and generates $\sigma_{7j} = \mathrm{H}(C_{2j}||sC''_{\mathrm{R}j})$. It then sends $\{C_{2j}||\sigma_{7j}\}$ to the $\mathrm{ECU}_{\mathrm{R}j}$.
9) The $\mathrm{ECU_S}$ sends a $\sigma' = \mathrm{H}_{K_{\text{group}}}(K)$ that allows the $\mathrm{ECU_R}$ to verify that it has received the correct sharing key K.
10) The $\mathrm{ECU}_{\mathrm{R}j}$ decrypts C_{2j} to obtain $\{sC''_{\mathrm{R}j}||AM_{r_\mathrm{S}}\}$ and verifies $\sigma_{7j} = \mathrm{H}(C_{2j}||sC''_{\mathrm{R}j})$. The $\mathrm{ECU}_{\mathrm{R}j}$ then computes K' = **EABEHP.ProxyDecrypt2**$(sC''_{\mathrm{R}j}, AM_{r_\mathrm{S}}, I_{\mathrm{R}j})$, and decrypts it with K_{group} to obtain K. Afterwards, the $\mathrm{ECU}_{\mathrm{R}j}$ verifies signature σ'. Finally, the $\mathrm{ECU}_{\mathrm{R}j}$ generates $C_{3j} = \mathrm{E}_K(\mathrm{ID}_{\mathrm{R}j})$ and $\sigma_{8j} = \mathrm{H}(K||C_{3j})$ and sends $\{\sigma_{8j}||C_{3j}\}$ to the $\mathrm{ECU_S}$.
11) The $\mathrm{ECU_S}$ stores the ID of the $\mathrm{ECU}_{\mathrm{R}j}$ that exchanged the data sharing key in the key table. The $\mathrm{ECU_S}$ verifies σ_{8j} and decrypts C_{3j} by K to obtain $\mathrm{ID}_{\mathrm{R}j}$. The $\mathrm{ECU_S}$ then generates $C_4 = \mathrm{E}_K(\mathrm{ID}_1||\mathrm{ID}_2||\cdots)$ for the encryp-

tion of all the $\mathrm{ECU}_{\mathrm{R}j}$'s identities and $\sigma_9 = \mathrm{H}(K||C_4)$. Subsequently, the $\mathrm{ECU}_{\mathrm{S}}$ sends $\{\sigma_9||C_4\}$ to the $\mathrm{ECU}_{\mathrm{R}j}$.

12) The $\mathrm{ECU}_{\mathrm{R}j}$ verifies signature σ_9 and decrypts C_4. If the $\mathrm{ECU}_{\mathrm{R}j}$ can find its own identity contained, it can authenticate the $\mathrm{ECU}_{\mathrm{S}}$ successfully, which verified the mutual authentication.

V. SECURITY ANALYSIS

This section proves the security of the cryptographic scheme proposed by Sec. III-B, and proves the security of the proposed protocol based on the security definition and model of Sec. II-C.

A. Security Analysis of Enhanced Attribute-based Encryption with Hidden Policy and Credential

Theorem 1 (Confidentiality of EABEHP): The proposed EABEHP is with ciphertext indistinguishability against chosen plaintext attack and restricted user coalition attack (C-IND-CPA-RUCA) if the decisional Diffie-Hellman (DDH) assumption holds.

Proof Sketch. The proof of C-IND-CPA-RUCA for the proposed EABEHP consists of two parts. One is the confidentiality of the produced ciphertext, i.e., C, in EABEHP. Moreover, sC' and sC'_r are also considered as the ciphertext of EABEHP since they are transformed from C and the only difference is that the positions of tuples of ciphertext are shuffled. Thus, the C-IND-CPA-RUCA security can be proven according to the same security proof for the proposed PEAPOD in [22] since the structure of the ciphertext in EABEHP is the same as that in PEAPOD. The other is the confidentiality of C, sC', sC'_r, and sC''_r against the proxy for **ProxyDecrypt1**. The second part of the proof of C-IND-CPA-RUCA considers that the confidentiality is guaranteed against the proxy even though AK_U and RK_U are known to the proxy. RK_U is the re-encryption key and it preserves the same structure of that in the PEAPOD scheme. Thus, the exposure of RK_U will not affect the confidentiality of the ciphertext in EABEHP. In addition, the exposure of AK_U will not affect the confidentiality as well since additional secret key ω_k, unknown to the proxy, is introduced to protect the secrets, $a_{j,i}$, contained in AK_U. Thus, without known ω_k, the proxy cannot eliminate the factor, g^{ω_k}, of blinding the message in the ciphertext to break the confidentiality of EABEHP.

Theorem 2 (Policy Privacy of EABEHP): Policy privacy holds if no one, including the security agent can learn any knowledge of the given policy T for encryption in the proposed EABEHP scheme.

Proof Sketch. In EABEHP, a message M to be encrypted will first be encoded as p_i for $i \in I$ such that $\Pi_{t_i=1 \wedge i \in I} p_i = M$. Here, $T = \{t_i\}_{i \in I}$ is a policy set, where $t_i = 1, 0, -1$ if the attribute i is required, irrelevant, and unrequired respectively. Here, $p_i = 1$ when $t_i = 0$ and $p_i = \alpha_i$ when $t_i = -1$, for randomly selected $\alpha_i \in \mathbb{Z}_q$. Afterwards, one can then encrypt each p_i as (A, B_i, C) with the public key of its corresponding attribute i. Thus, the only way to learn the given policy I depends on the generated p_i. However, each p_i is encrypted using ElGamal encryption which is the primitive of EABEHP with indistinguishability under chosen plaintext attack (IND-CPA) security. Thus, no one can learn any knowledge from p_i by the ciphertext (A, B_i, C). From the above, no one, including the security agent, can learn the knowledge of policy. Consequently, EABEHP is with policy privacy.

Theorem 3 (Credential Privacy of EABEHP): The proposed EABEHP is with hidden credentials against outsider and decryption proxy if the underlying hash function is a pseudorandom function, and the shuffle function is a pseudorandom permutation.

Proof Sketch. Since the attribute information, i.e., AK_{U_r} or $\hat{I}_r$, of each user will be exposed when during the execution of **ProxyDecrypt1** or **Extract** function, the privacy of user credential is guaranteed if the original attribute information cannot be disclosed. (1) AK_{U_r} is a combination of ω_k and user secret keys for each attribute generated by the transformuserkey algorithm, where $\omega_k = \mathrm{H}(r_k||K_{\text{group}})$, and K_{group} is a pre-distributed key to legitimate users in the system. Therefore, no one has non-negligible probability to distinguish AK_{U_r} by distinguishing a ω_k from random string based on the security of pseudorandom function. (2) $\hat{I}_r$ is the set of inverse permuted attribute indices from the set of original attribute indices by a pseudorandom permutation, shuffle function SH with a given ω_k, which is only known between users. Thus, no one has non-negligible probability to distinguish a permuted index from an original index based on the security of pseudorandom permutation. Thus, the proposed EABEHP is with hidden credentials based on the security of the pseudorandom function and permutations.

B. Security Analysis of EC-SVC Protocol

In this subsection, we present to the security analysis of the protocol proposed in Sec. IV-B. The proposed protocol proves that mutual authentication, attribute-based key exchange, policy privacy, and credential privacy have been achieved as follows.

Theorem 4 (EC-SVC Security): The proposed EC-SVC protocol is said to be the attribute-based authenticated key exchange protocol with hidden policy and credential if H is a pseudorandom function, $\mathrm{E_S}$ is a pseudorandom permutation, and EABEHP is a C-IND-CPA-RUCA-secure and P-IND-CPA-UCA-secure attribute-based encryption scheme.

The advantage $\mathrm{Adv}_{\mathcal{A}}^{\text{EC-SVC}}$ that an attacker $\mathcal{A}$ break the security of EC-SVC protocol are given by

$$\begin{aligned}\mathrm{Adv}_{\mathcal{A}}^{\text{EC-SVC}} \leq & 11\mathrm{Adv}_{\mathrm{H}} + 5\mathrm{Adv}_{\mathrm{E_S}} + 2\mathrm{Adv}_{\text{C-IND-CPA}} \\ & + 2\mathrm{Adv}_{\text{P-IND-CPA}}. \end{aligned} \tag{8}$$

where $\mathrm{Adv}_{\mathrm{H}}$ is an advantage that breaks the security of the pseudorandom function, $\mathrm{Adv}_{\mathrm{E_S}}$ is an advantage that breaks the security of the pseudorandom permutation, $\mathrm{Adv}_{\text{C-IND-CPA}}$ is an advantage that breaks the security of the C-IND-CPA-RUCA security of the EABEHP, and $\mathrm{Adv}_{\text{P-IND-CPA}}$ is an advantage that breaks the security of the P-IND-CPA-UCA security of the EABEHP.

We proceed with the security game to prove the security of the proposed protocol. The security game proceeds each four requirements mentioned above and claims that the advantages of $\mathcal{A}$ for the proposed protocol can be negligible, depending on the advantages of $\mathcal{A}$ in each game, where $\mathcal{A}$ is an attacker that breaks the security of mutual authentication, attribute-based key exchange, policy privacy, and credential privacy. We denote $\mathrm{Adv}^{\text{EC-SVC}}_{\mathcal{A},i}$ as the advantage of $\mathcal{A}$ in game G_i.

Game G_0 : This is a real game, $\mathcal{A}$ has access to EABEHP's master public key MPK, all ECU's identity (ID) $\{ID_i\}_{i=0,1,...}$. In addition, $\mathcal{A}$ has the ability to query all oracles specified in Sec. II-C and knows all the structure of the protocol. Since this paper has shown that EABEHP can be proven IND-CPA secure by simulating EABEHP in Sec. V-A, all the parameters related to EABEHP can be successfully simulated. Therefore, we have

$$\mathrm{Adv}^{\text{EC-SVC}}_{\mathcal{A}} = \mathrm{Adv}^{\text{EC-SVC}}_{\mathcal{A},0}. \tag{9}$$

Game G_1 (Mutual Authentication). In the game G_1, We describe the events of the game as follows. E_1 is an event in which $\mathcal{A}$ impersonates $\mathrm{ECU_S}$ by sending the correct σ_1 to the SA. The E_2 is an event in which $\mathcal{A}$ impersonates the SA by sending the correct σ_2 to the $\mathrm{ECU_S}$. The E_3 is an event in which $\mathcal{A}$ impersonates the $\mathrm{ECU}_{\mathrm{R}j}$ by sending the correct σ_{4j} to the SA. The E_4 is an event in which $\mathcal{A}$ impersonates the SA by sending the correct σ_{5j} to $\mathrm{ECU}_{\mathrm{R}j}$. The E_5 is an event in which $\mathcal{A}$ impersonates the $\mathrm{ECU}_{\mathrm{R}j}$ by sending the correct $\sigma_{8j}||C_{3j}$ to the $\mathrm{ECU_S}$. The E_6 is an event in which $\mathcal{A}$ impersonates the $\mathrm{ECU_S}$ by sending the correct $\sigma_9||C_4$ to the $\mathrm{ECU}_{\mathrm{R}j}$. We construct a simulator S_1 of the EC-SVC that interacts with $\mathcal{A}$ as the security game defined in Definition 4. In addition, S_1 is provided with the master public key of EABEHP to successfully simulate EC-SVC. If the E_1 happens, S_1 can exploit the ability of $\mathcal{A}$ to break the underlying pseudorandom function security. Hence, we have

$$\begin{aligned}
\mathrm{Adv_H} &\geq \{\Pr[\mathrm{S_H}, \mathrm{E}_1] + \Pr[\mathrm{S_H}, \neg\mathrm{E}_1]\} - \frac{1}{2}\\
&= \{\Pr[\mathrm{S_H}|\mathrm{E}_1] \times \Pr[\mathrm{E}_1] + \Pr[\mathrm{S_H}|\neg\mathrm{E}_1] \times (1 - \Pr[\mathrm{E}_1]\} - \frac{1}{2}\\
&= \{1 \times \mathrm{Adv_{E_1}} + \frac{1}{2} \times (1 - \mathrm{Adv_{E_1}})\} - \frac{1}{2} = \frac{\mathrm{Adv_{E_1}}}{2},
\end{aligned} \tag{10}$$

where $\mathrm{S_H}$ is the event of distinguishing a pseudorandom function from a truly random function successfully, the $\neg\mathrm{E}_1$ is the complementary event of the E_1, $\mathrm{Adv_{E_1}}$ is the advantage of the E_1, which is the probability that an attacker sends a valid σ_1 to impersonate an $\mathrm{ECU_S}$. Therefore, we have $\mathrm{Adv_{E_1}} \leq 2\mathrm{Adv_H}$. For the probabilities of events E_2, E_3, and E_4, we have $\mathrm{Adv_{E_2}} \leq 2\mathrm{Adv_H}$, $\mathrm{Adv_{E_3}} \leq 2\mathrm{Adv_H}$, and $\mathrm{Adv_{E_4}} \leq 2\mathrm{Adv_H}$. The security analysis regarding E_5 can be divided into two cases: (1) When E_5 happened, S_1 can also break the security of underlying pseudorandom function or pseudorandom permutation by exploiting the ability of $\mathcal{A}$. Thus, we have $\mathrm{Adv_{E_5}} \leq 2\mathrm{Adv_{E_S}}$, when S_1 simulates the protocol based on the function, which is either a pseudorandom permutation or a random permutation. In addition, when S_1 simulates the protocol based on the function, which is either a pseudorandom function or a random function, we have $\mathrm{Adv_{E_5}} \leq 2\mathrm{Adv_H}$. From the above, we have

$$\mathrm{Adv_{E_5}} \leq \mathrm{Adv_{E_S}} + \mathrm{Adv_H}. \tag{11}$$

(2) When E_5 happened, S_1 can also break the security of underlying pseudorandom permutation or C-IND-CPA-RUCA by exploiting the ability of $\mathcal{A}$. Thus, we have

$$\mathrm{Adv_{E_5}} \leq \mathrm{Adv_{E_S}} + \mathrm{Adv_{C\text{-}IND\text{-}CPA}}. \tag{12}$$

Through the results of both cases, we have

$$\mathrm{Adv_{E_5}} \leq \mathrm{Adv_{E_S}} + \frac{1}{2}(\mathrm{Adv_H} + \mathrm{Adv_{C\text{-}IND\text{-}CPA}}). \tag{13}$$

In the same way, we have $\mathrm{Adv_{E_6}} \leq \mathrm{Adv_{E_S}} + \frac{1}{2}(\mathrm{Adv_H} + \mathrm{Adv_{C\text{-}IND\text{-}CPA}})$. Finally, we have

$$\mathrm{Adv}^{\text{EC-SVC}}_{\mathcal{A},0} \leq \mathrm{Adv}^{\text{EC-SVC}}_{\mathcal{A},1} + 9\mathrm{Adv_H} + 2\mathrm{Adv_{E_S}} + \mathrm{Adv_{C\text{-}IND\text{-}CPA}} \tag{14}$$

Game G_2 (Attribute-based key exchange). The proposed protocol achieves attribute-based key exchange through the *enhanced attribute-based encryption with hidden policy and credential* newly proposed in Sec. III-B. In game G_2, we construct a simulator S_2 that interacts with $\mathcal{A}$ in the security games defined in Definition 5. S_2 is provided with the master public key of EABEHP to successfully simulate EC-SVC. $\mathcal{A}$ queries the **Test** after interacting with the security game with S_2. S_2 responds to the $\mathcal{A}$ with an attribute-based key K or a random string according to a random bit. If $\mathcal{A}$ can successfully guess the attribute-based key K, S_2 can also break the security of underlying pseudorandom permutation or C-IND-CPA-RUCA by exploiting the ability of $\mathcal{A}$. Therefore, we have

$$\begin{aligned}
\mathrm{Adv_{E_S}} &\geq \{\Pr[\mathrm{S_{E_S}}, \mathrm{E_{AKE}}] + \Pr[\mathrm{S_{E_S}}, \neg\mathrm{E_{AKE}}]\} - \frac{1}{2}\\
&= \{\Pr[\mathrm{S_{E_S}}|\mathrm{E_{AKE}}] \times \Pr[\mathrm{E_{AKE}}]\\
&\quad + \Pr[\mathrm{S_{E_S}}|\neg\mathrm{E_{AKE}}] \times (1 - \Pr[\mathrm{E_{AKE}}]\} - \frac{1}{2}\\
&= \{1 \times \mathrm{Adv_{E_{AKE}}} + \frac{1}{2} \times (1 - \mathrm{Adv_{E_{AKE}}})\} - \frac{1}{2} = \frac{\mathrm{Adv_{E_{AKE}}}}{2},
\end{aligned} \tag{15}$$

where $\mathrm{Adv_{E_{AKE}}}$, which is the advantage of the $\mathrm{E_{AKE}}$, is probability that an attacker distinguishes the attribute based key K from a random string. Therefore, we have $\mathrm{Adv_{E_{AKE}}} \leq 2\mathrm{Adv_{E_S}}$. Similar to the game G_1, we have $\mathrm{Adv_{E_{AKE}}} \leq 2\mathrm{Adv_{C\text{-}IND\text{-}CPA}}$. From the above, we have

$$\mathrm{Adv_{E_{AKE}}} \leq \mathrm{Adv_{E_S}} + \mathrm{Adv_{C\text{-}IND\text{-}CPA}}. \tag{16}$$

Therefore, we have

$$\mathrm{Adv}^{\text{EC-SVC}}_{\mathcal{A},1} \leq \mathrm{Adv}^{\text{EC-SVC}}_{\mathcal{A},2} + \mathrm{Adv_{E_S}} + \mathrm{Adv_{C\text{-}IND\text{-}CPA}}. \tag{17}$$

Game G_3 (Policy Privacy). The policy privacy of the proposed protocol can be analyzed in a similar way to the game G_2. In game G_3, we construct a simulator S_3 that interacts with $\mathcal{A}$ in the security games defined in Definition 6. S_3 is provided with the master public key of EABEHP to successfully simulate EC-SVC. $\mathcal{A}$ queries the **TestPolicy** after interacting with the security game with S_3. S_3 responds to the $\mathcal{A}$ with an P_0 or P_1 according to a random bit. If $\mathcal{A}$ can successfully guess the correct policy, then $\mathcal{A}$ has the

TABLE I
COMPARISON ON SECURITY WITH RELATED WORKS

	[9]	[11]	[12]	[13]	Our work
	[14]	[16]	[17]	[18]	(EC-SVC)
Mounted Additional Device	√	√	√	×	×
	×	×	×	×	
Message Authentication and Integrity	√	√	√	√	√
	√	√	√	√	
Data Confidentiality	√	×	×	×	√
	×	√	√	√	
Resistance to Replay Attacks	√	×	×	√	√
	√	√	√	√	
Attribute-based Access Control	×	×	×	×	√
	×	×	×	×	
Privacy Preserving for Corrupted Devices	-	-	-	-	√
	-	-	-	-	

advantage of breaking P-IND-CPA-UCA security of EABEHP. Therefore, we have

$$\begin{aligned}
&\mathrm{Adv}_{\text{P-IND-CPA}} \\
&\geq \{\Pr[\mathrm{S}_{\text{P-IND-CPA}}, \mathrm{E}_{\mathrm{PP}}] + \Pr[\mathrm{S}_{\text{P-IND-CPA}}, \neg \mathrm{E}_{\mathrm{PP}}]\} - \frac{1}{2} \\
&= \{\Pr[\mathrm{S}_{\text{P-IND-CPA}} | \mathrm{E}_{\mathrm{PP}}] \times \Pr[\mathrm{E}_{\mathrm{PP}}] \\
&\quad + \Pr[\mathrm{S}_{\text{P-IND-CPA}} | \neg \mathrm{E}_{\mathrm{PP}}] \times (1 - \Pr[\mathrm{E}_{\mathrm{PP}}]\} - \frac{1}{2} \\
&= \{1 \times \mathrm{Adv}_{\mathrm{PP}} + \frac{1}{2} \times (1 - \mathrm{Adv}_{\mathrm{PP}})\} - \frac{1}{2} = \frac{\mathrm{Adv}_{\mathrm{PP}}}{2}, \qquad (18)
\end{aligned}$$

where E_{PP} is the event that distinguishing the correct policy from random string with additional advantage, and $\mathrm{Adv}_{\mathrm{PP}}$ is the advantage of breaking policy privacy. Thus, we have

$$\mathrm{Adv}_{\mathcal{A},2}^{\text{EC-SVC}} \leq \mathrm{Adv}_{\mathcal{A},3}^{\text{EC-SVC}} + 2\mathrm{Adv}_{\text{P-IND-CPA}}. \qquad (19)$$

Game G_4 (Credential Privacy). In game G_4, we construct a simulator S_4 that interacts with $\mathcal{A}$ in the security games defined in Definition 7. S_4 successfully simulates EC-SVC with the supplied EABEHP's master public key. After interacting with the security game with S_4, $\mathcal{A}$ queries the **TestCert**. S_4 responds to $\mathcal{A}$ with a target credential or randomly selected credential according to a random bit. E_9 is an event in which credentials are successfully guessed in the **ProxyDecrypt1** algorithm by adversary $\mathcal{A}$. E_{10} is a case in which the credentials are successfully guessed in the **Extract** algorithm by $\mathcal{A}$. If the E_9 happens, $\mathcal{A}$ has the advantage of breaking pseudorandom function security. Therefore, we have

$$\begin{aligned}
&\mathrm{Adv}_{\mathrm{H}} \geq \{\Pr[\mathrm{S}_{\mathrm{H}}, \mathrm{E}_9] + \Pr[\mathrm{S}_{\mathrm{H}}, \neg \mathrm{E}_9]\} - \frac{1}{2} \\
&= \{\Pr[\mathrm{S}_{\mathrm{H}} | \mathrm{E}_9] \times \Pr \mathrm{E}_9] + \Pr[\mathrm{S}_{\mathrm{H}} | \neg \mathrm{E}_9] \times (1 - \Pr[\mathrm{E}_9]\} - \frac{1}{2} \\
&= \{1 \times \mathrm{Adv}_{\mathrm{E}_9} + \frac{1}{2} \times (1 - \mathrm{Adv}_{\mathrm{E}_9})\} - \frac{1}{2} = \frac{\mathrm{Adv}_{\mathrm{E}_9}}{2}, \qquad (20)
\end{aligned}$$

where $\mathrm{Adv}_{\mathrm{E}_9}$, which is the advantage of the E_9, is the probability that an attacker distinguishes the real user private key from a random string. Therefore, we have $\mathrm{Adv}_{\mathrm{E}_9} \leq 2\mathrm{Adv}_{\mathrm{H}}$. In the same way, for E_{10} we have $\mathrm{Adv}_{\mathrm{E}_{10}} \leq 2\mathrm{Adv}_{\mathrm{E}_{\mathrm{S}}}$. Therefore, we have $\mathrm{Adv}_{\mathrm{CP}} \leq 2\mathrm{Adv}_{\mathrm{H}} + 2\mathrm{Adv}_{\mathrm{E}_{\mathrm{S}}}$, where E_{CP} is the event that distinguishing the target credential from randomly selected credential, and $\mathrm{Adv}_{\mathrm{CP}}$ is the advantage of breaking credential privacy.

$$\mathrm{Adv}_{\mathcal{A},3}^{\text{EC-SVC}} \leq \mathrm{Adv}_{\mathcal{A},4}^{\text{EC-SVC}} + 2\mathrm{Adv}_{\mathrm{H}} + 2\mathrm{Adv}_{\mathrm{E}_{\mathrm{S}}}. \qquad (21)$$

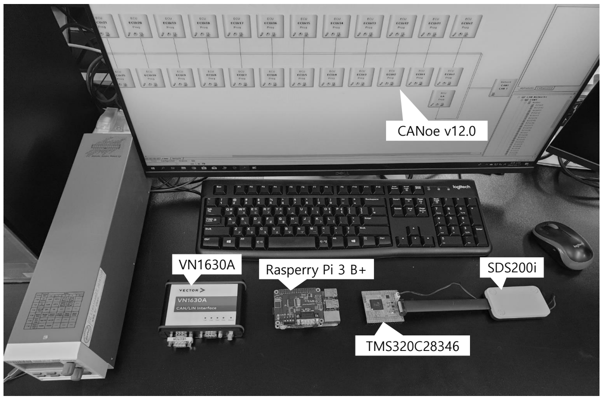


Fig. 4. Performance evaluation environment.

TABLE II
HARDWARE AND SOFTWARE FOR PERFORMANCE EVALUATION

Model	**Note**
Raspberry Pi 3 B+ (Single-board Computer)	Clock speed : 1.4GHz or 600MHz
TI TMS320C28346 (Micro controller unit (MCU))	Clock speed : 300MHz
SDS200i	JTAG Emulator
VN1630A	CAN-FD Network Interface
Java JCA/JCE	Java Cryptography Package
Code Composer Studio V9.3	MCU Compiler
CANoe V12.0	In-vehicle Network simulator

There are no additional advantages beyond those analyzed in the game above. Thus, by equations (14), (17), (19), and (21) we can claim that the advantages of $\mathcal{A}$ to the proposed EC-SVC are as given by

$$\begin{aligned}
\mathrm{Adv}_{\mathcal{A}}^{\text{EC-SVC}} \leq & 11\mathrm{Adv}_{\mathrm{H}} + 5\mathrm{Adv}_{\mathrm{E}_{\mathrm{S}}} + 2\mathrm{Adv}_{\text{C-IND-CPA}} \\
& + 2\mathrm{Adv}_{\text{P-IND-CPA}}. \qquad (22)
\end{aligned}$$

Finally, the overall security comparison between security protocols and related works is shown in Table I. The work satisfies all the security requirements without mounting additional components on ECUs.

VI. PERFORMANCE ANALYSIS

In this section, we evaluate the performance in various aspects for demonstrating that the proposed security protocol is practical in in-vehicle scenarios. This work builds up the testbed based on the hardware and software which are the Raspberry Pi, TMS320C28346, and CANoe by Vector Co [29]. Unless otherwise specified, the simulation environment in Fig. 4 and the specifications of the equipment in Table II are used. The testbed adopts CANoe to implement in-vehicle network based on the flexible data rate (CAN-FD) standard [30]. This section first analyzes the execution time of each cryptographic algorithm for each device. We then evaluate the performance of the proposed security protocols in the simulation environment implemented.

TABLE III
EXECUTION TIME OF CRYPTOGRAPHIC ALGORITHM

	Algorithm execution time (μs)		
Algorithm	**SHA-256**	**AES128(Enc)**	**AES128(Dec)**
ECU	130.8	149.5	198.9
SA (600MHz)	8.4	5.4	6.7
SA (1.4GHz)	3.6	12.7	13.8

TABLE IV
EXECUTION TIME OF EABEHP ALGORITHM

		Algorithm execution time (ms)			
Number of system attributes		4	8	12	16
		20	24	28	32
EABEHP Encrypt+Shuffle	ECU	144.7	241.1	338.8	436.9
		529.5	635.5	714.8	817.9
EABEHP TransformCipherText	SA (600MHz)	7	13	20.9	27.8
		34.4	41.8	47.6	54.8
	SA (1.4GHz)	3	6	9	12
		14.5	17.5	21.2	23.6
Number of receiver attributes		4	8	12	16
		20	24	28	32
EABEHP Extract+ProxyDecrypt1	SA (600MHz)	1.92	2.05	2.25	2.46
		2.65	3	3.24	3.64
	SA (1.4GHz)	0.82	0.89	0.96	1.08
		1.12	1.25	1.44	1.56

A. *Cryptographic Algorithm Evaluation*

This subsection evaluates the execution time of the different cryptographic algorithms (i.e., SHA-256, AES-128, EABEHP). The cryptographic algorithm is implemented and measured in Java Cryptography Architecture (JCA) / Java Cryptography Extension (JCE) and Code Composer Studio. We construct the EABEHP algorithm based on ElGamal encryption. In addition, for better accuracy, we measure the execution time for 10,000 times repetitively and obtain the average execution time. We use the TMS320C28346 as the ECU and the Raspberry Pi as the SA.

In the proposed security protocol, 48byte input is used to the SHA256 algorithm, and 16byte output is obtained by truncated MAC [16], [17], [31]. In addition, 48bytes out of 64bytes in the data payload is used as input data to AES128 algorithm. The length of all messages sent by the ECU and the SA is 64bytes, and the remnant is assumed to be padded. Under this description, the measured execution time of various cryptographic algorithm is shown in Table III. Table III show the cryptographic algorithm execution time of SHA-256, AES128 Encryption, and AES128 Decryption at the ECU and the SA.

Table IV shows the EABEHP Encrypt and Shuffle algorithm execution time at the ECU[1], and the EABEHP TransformCipherText execution time and EABEHP Extract and ProxyDecrypt1 execution time at the SA. The EABEHP Encrypt and Shuffle and EABEHP TransformCipherText algorithms perform exponential operations for each attribute, which significantly increases the algorithm execution time as the number of system attributes increases. On the other hand, the EABEHP Extract and ProxyDecrypt1 algorithm add multiplication operations as the number of system attributes increases, so the execution time of the algorithm slightly increases. From Table III, we can also see that the Encrypt and Shuffle algorithm occupies the majority of the execution time of the proposed EABEHP.

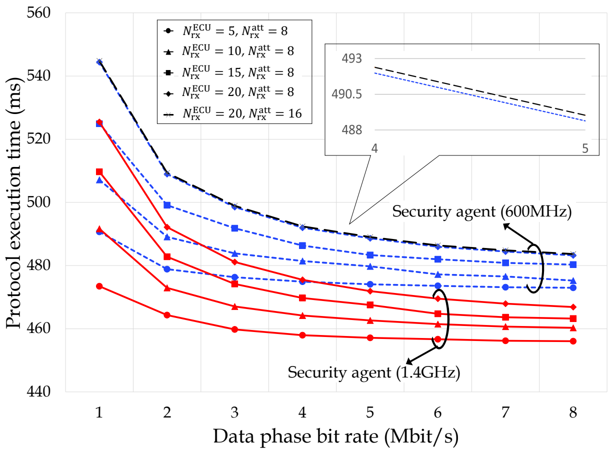


Fig. 5. Execution time of the proposed protocol as a function of the data phase bit rate for different numbers of receiver-ECUs, N_{rx}^{ECU}, and receiver attributes, N_{rx}^{att}. Here, the number of system attributes is 16.

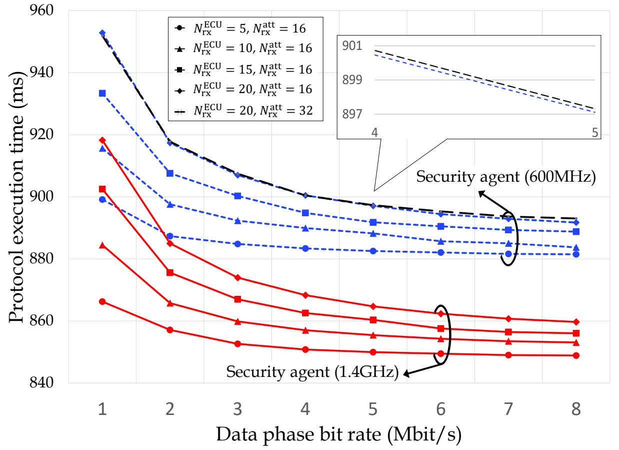


Fig. 6. Execution time of the proposed protocol as a function of the data phase bit rate for different numbers of receiver-ECUs, N_{rx}^{ECU}, and receiver attributes, N_{rx}^{att}. Here, the number of system attributes is 32.

B. *Security Protocol Evaluation*

This subsection measures the execution time of the security protocol, based on the cryptographic algorithm evaluation. Using the CANoe v12.0 by Vector Co, we implement an evaluation environment similar to the real CAN-FD. The execution time of the proposed protocol is measured by considering the communication delay as well as the execution time of the cryptographic algorithms in Tables III and IV at the CANoe virtual ECU node. Note that, this work also achieves several additional features, such as attribute-based access control, and privacy-preserving for corrupted devices in addition to the security features achieved in existing in-vehicle security works. Furthermore, since this work proposes the novel edge computing-based security protocol that achieves a

---

[1] Note that, it was not possible to perform EABEHP Encrypt and Shuffle at the ECU due to the hardware limitation. Hence, we obtain the execution time of them by measuring of Raspberry Pi and then scaling the time by the execution time ratio of other operations (e.g., SHA-256 and AES128 in Table IV). We expect that the ECU, developed in the near future, will be able to support the advanced cryptographic operations.

higher level of security and has reasonable latency in-vehicle systems, performance comparisons with other works are not included in the paper. Instead, we present the performance evaluation in various aspects to show that the proposed security protocol is practical.

As shown in previous page, Figs. 5 and 6 show the execution time of the attribute-based authenticated key exchange protocols for different data phase bit rates. We perform the evaluation with the fixed arbitration phase bit rate of 0.5Mbit/s and adjust the data phase bit rate from 1Mbit/s to 8Mbit/s. The number of system attributes and receiver attributes is set to 16 and 8, respectively, in Fig. 5, and 32 and 16, respectively, in Fig. 6. The measurement results when the SA clock speed is 1.4GHz and 600MHz are represented by the solid and dotted lines, respectively.

From Figs. 5 and 6, we can see that the protocol execution time decreases as the data phase bit rate increases since the communication delay in CAN becomes smaller. By comparing the results with $N_{\mathrm{rx}}^{\mathrm{att}} = 16$ and $N_{\mathrm{rx}}^{\mathrm{att}} = 8$ in Fig. 5 and those with $N_{\mathrm{rx}}^{\mathrm{att}} = 32$ and $N_{\mathrm{rx}}^{\mathrm{att}} = 16$ in Fig. 6, we can see that the number of the receiver attributes has little impact on the protocol execution time, while the number of system attributes affects significantly on the protocol execution time. However, note that even with 32 system attributes, which is quite a large number to classify ECUs since there are many ECUs with overlapping roles, the execution time of the proposed protocol is less than 1 second. This means the proposed protocol can satisfy the practical requirements of in-vehicle networks.

Figure 7 shows the protocol execution time according to the number of system attributes for different numbers of system attributes, $N_{\mathrm{sys}}^{\mathrm{att}}$. The number of receiver attributes is set to be the same for all receivers, where the data phase bit rate is fixed at 4Mbit/s, the number of receiver-ECUs is 10, and the clock speed of the SA is 1.4GHz. We can see that the protocol execution time increases significantly as the number of system attributes increases while as mentioned above the number of receiver attributes has little effect on the protocol execution time. This is because the EABEHP Extract and ProxyDecrypt1, affected by the number of receiver attributes, are performed by a high-performance device, i.e., SA, and with relatively simple operations compared to other EABEHP algorithms. On the other hand, the EABEHP Encrypt algorithm, affected by the number of system attributes, is performed by the low-performance device, i.e., ECU, and with the complex operations.

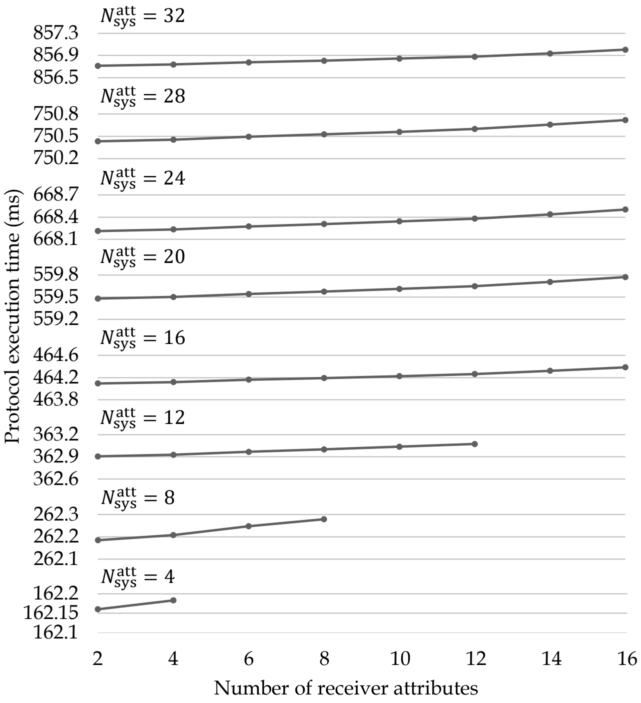


Fig. 7. Execution time of the proposed protocol as a function of the number of receiver attributes, $N_{\mathrm{rx}}^{\mathrm{att}}$ for different numbers of system attribute, $N_{\mathrm{sys}}^{\mathrm{att}}$. Here, the number of receiver-ECUs, $N_{\mathrm{rx}}^{\mathrm{ECU}}$ is 10.

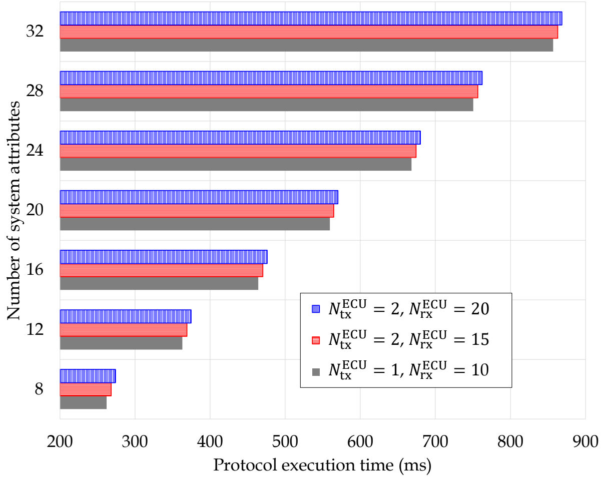


Fig. 8. Execution time of the proposed protocol as a function of numbers of system attribute, $N_{\mathrm{sys}}^{\mathrm{att}}$, for different numbers of sender-ECUs, $N_{\mathrm{tx}}^{\mathrm{ECU}}$. Here, the number of receiver-ECUs, $N_{\mathrm{rx}}^{\mathrm{ECU}}$ allocated to each sender-ECU is 10.

Figure 8 shows the protocol execution time according to the number of system attributes for different numbers of senders and receivers, denoted by $N_{\mathrm{tx}}^{\mathrm{ECU}}$ and $N_{\mathrm{rx}}^{\mathrm{ECU}}$, respectively. Here, we assume that 10 receivers access the message from one sender. The different number of senders and receivers are used : 1 sender and 10 receivers, 2 senders and 15 receivers (5 receivers received messages from both senders), and 2 senders and 20 receivers (all receivers received messages only from one sender). We can see that the number of system attributes have a little impact on the gap in protocol execution time for the above three cases. This is because, when the priority of the message is well-established, the cryptographic algorithm execution time, associated with the number of system attributes, is generally much longer than the communication delay. Here, the increase in communication delay due to the increase in the number of senders who simultaneously transmit the message is negligible. Figure 9 shows the protocol execution time versus the data phase bit rate for the above three cases. We can see that the protocol execution time difference among three cases becomes smaller as the data phase bit rate increases. This is because the number of communications is different in three cases, and the communication delay is inversely proportional to the data phase bit rate. Hence, the larger the data phase bit rate gives the less the protocol execution time difference. Through the results of Figs. 8 and 9, we can see that if the data phase bit rate is high enough, the time taken to execute the proposed attribute-based key exchange process on all ECUs

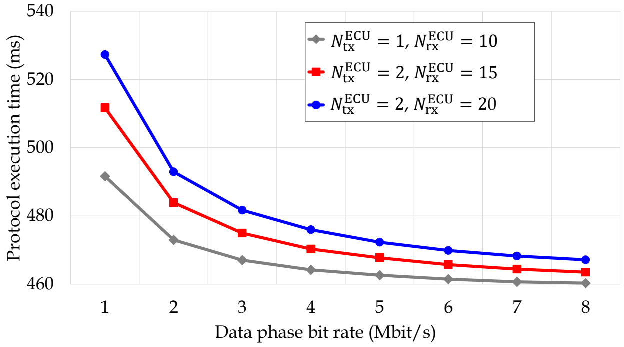


Fig. 9. Execution time of the proposed protocol as a function of the data phase bit rate, for different numbers of sender-ECUs, $N_{\text{tx}}^{\text{ECU}}$. Here, the number of receiver-ECUs, $N_{\text{rx}}^{\text{ECU}}$ allocated to each sender-ECU is 10.

will not be significantly affected by the in-vehicle network size, i.e., the numbers of sender-ECUs and receiver-ECUs. Therefore, the proposed protocol is expected to be executed in a reasonable time even for different in-vehicle network sizes, which shows the feasibility and the practicality of the proposed protocol.

VII. CONCLUSION

This work proposes an edge computing-based in-vehicle security protocol with the attribute-based access control that privacy for policy and credentials. The security of this protocol has been proved through security analysis to be limited to the security of pseudorandom function, pseudorandom permutation, and C-IND-CPA-RUCA and P-IND-CPA-UCA EABEHP. Specifically, the performance analysis of the proposed protocol shows the effect of protocol execution time according to the data phase bit rate, the number of system attributes, the number of receiver attributes, and the number of sender and receiver-ECUs. This shows that a high-security level can be satisfied in an appropriate latency in an in-vehicle communication environment having a resource-poor ECU. Hence, this work has demonstrated to support efficient secure communication with fine-grained access control for in-vehicle networks.

REFERENCES

[1] W. Stephan, R. Solveig, and M. Markus, “Aspects of secure vehicle software flashing,” in *Embedded Security in Cars*. Springer, 2006, pp. 17–26.

[2] M. Wolf, A. Weimerskirch, and C. Paar, “Secure in-vehicle communication,” in *Embedded Security in Cars*. Springer, 2006, pp. 95–109.

[3] F. Kargl, P. Papadimitratos, L. Buttyan, M. Müter, E. Schoch, B. Wiedersheim, T.-V. Thong, G. Calandriello, A. Held, A. Kung *et al.*, “Secure vehicular communication systems: implementation, performance, and research challenges,” *IEEE Commun. Mag.*, vol. 46, no. 11, pp. 110–118, Nov. 2008.

[4] D. K. Nilsson, U. E. Larson, and E. Jonsson, “Efficient in-vehicle delayed data authentication based on compound message authentication codes,” in *Proc. IEEE Veh. Technol. Conf.*, Sep. 2008, pp. 1–5.

[5] T. Hoppe, S. Kiltz, and J. Dittmann, “Security threats to automotive can networks-practical examples and selected short-term countermeasures,” in *Proc. International Conference on Computer Safety, Reliability, and Security*, Sep. 2008, pp. 235–248.

[6] K. Koscher, A. Czeskis, F. Roesner, S. Patel, T. Kohno, S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham *et al.*, “Experimental security analysis of a modern automobile,” in *Proc. IEEE Symposium on Security and Privacy*, May 2010, pp. 447–462.

[7] S. Checkoway, D. McCoy, B. Kantor, D. Anderson, H. Shacham, S. Savage, K. Koscher, A. Czeskis, F. Roesner, T. Kohno *et al.*, “Comprehensive experimental analyses of automotive attack surfaces,” in *Proc. USENIX Security Symposium*, vol. 4, Aug. 2011, pp. 447–462.

[8] M. Farsi, K. Ratcliff, and M. Barbosa, “An overview of controller area network,” *Computing & Control Engineering Journal*, vol. 10, no. 3, pp. 113–120, Jun. 1999.

[9] H. Oguma, A. Yoshioka, M. Nishikawa, R. Shigetomi, A. Otsuka, and H. Imai, “New attestation based security architecture for in-vehicle communication,” in *Proc. IEEE Global Telecommunications Conference*, Nov. 2008, pp. 1–6.

[10] P. Kleberger, T. Olovsson, and E. Jonsson, “Security aspects of the in-vehicle network in the connected car,” in *Proc. IEEE Intelligent Vehicles Symposium (IV)*, Jun. 2011, pp. 528–533.

[11] H. Schweppe, Y. Roudier, B. Weyl, L. Apvrille, and D. Scheuermann, “Car2x communication: securing the last meter-a cost-effective approach for ensuring trust in car2x applications using in-vehicle symmetric cryptography,” in *Proc. IEEE Veh. Technol. Conf.*, Sep. 2011, pp. 1–5.

[12] A. V. Herrewege, D. Singelee, and I. Verbauwhede, “CANAuth-a simple, backward compatible broadcast authentication protocol for CAN bus,” in *Proc. ECRYPT Workshop on Lightweight Cryptography*, vol. 2011, Nov. 2011, p. 20.

[13] B. Groza, S. Murvay, A. V. Herrewege, and I. Verbauwhede, “Libra-can: a lightweight broadcast authentication protocol for controller area networks,” in *Proc. International Conference on Cryptology and Network Security*, Dec. 2012, pp. 185–200.

[14] R. Kurachi, Y. Matsubara, H. Takada, N. Adachi, Y. Miyashita, and S. Horihata, “CaCAN - centralized authentication system in CAN (controller area network),” in *Proc. Int. Conf. on Embedded Security in Cars*, Nov. 2014, pp. 1–10.

[15] C.-W. Lin and A. Sangiovanni-Vincentelli, “Cyber-security for the controller area network (CAN) communication protocol,” in *Proc. International Conference on Cyber Security*, Dec. 2012, pp. 1–7.

[16] S. Woo, H. J. Jo, and D. H. Lee, “A practical wireless attack on the connected car and security protocol for in-vehicle CAN,” *IEEE Trans. Intell. Transp. Syst.*, vol. 16, no. 2, pp. 993–1006, Apr. 2015.

[17] 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.

[18] B. Groza and P.-S. Murvay, “Identity-based key exchange on in-vehicle networks: CAN-FD & FlexRay,” *Sensors*, vol. 19, no. 22, p. 4919, Nov. 2019.

[19] Z. Lu, Q. Wang, X. Chen, G. Qu, Y. Lyu, and Z. Liu, “Leap: a lightweight encryption and authentication protocol for in-vehicle communications,” in *Proc. IEEE Intelligent Transportation Systems Conference*, Oct. 2019, pp. 1158–1164.

[20] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, “Edge computing: Vision and challenges,” *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, Oct. 2016.

[21] R.-H. Hsu, J. Lee, T. Q. Quek, and J.-C. Chen, “Reconfigurable security: Edge-computing-based framework for IoT,” *IEEE Network*, vol. 32, no. 5, pp. 92–99, Sep. 2018.

[22] A. Kapadia, P. P. Tsang, and S. W. Smith, “Attribute-based publishing with hidden credentials and hidden policies,” in *Proc. NDSS*, vol. 7, Feb. 2007, pp. 179–192.

[23] J. Bethencourt, A. Sahai, and B. Waters, “Ciphertext-policy attribute-based encryption,” in *IEEE Symposium on Security and Privacy*, May 2007, pp. 321–334.

[24] M. Luby and C. Rackoff, “How to construct pseudorandom permutations from pseudorandom functions,” *SIAM Journal on Computing*, vol. 17, no. 2, pp. 373–386, 1988.

[25] G. Tsudik, “Message authentication with one-way hash functions,” *ACM SIGCOMM Computer Communication Review*, vol. 22, no. 5, pp. 29–38, Oct. 1992.

[26] M. Bellare, R. Canetti, and H. Krawczyk, “Keying hash functions for message authentication,” in *Proc. Annual international cryptology conference*, Aug. 1996, pp. 1–15.

[27] D. H. Phan and D. Pointcheval, “About the security of ciphers (semantic security and pseudo-random permutations),” in *Proc. International Workshop on Selected Areas in Cryptography*, Aug. 2004, pp. 182–197.

[28] W. Trappe, J. Song, and R. Poovendran, “Key distribution for secure multimedia multicasts via data embedding,” in *Proc. IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings*, May 2001, pp. 1449–1452.

[29] Vector, [Online] Available: www.vector.com.

[30] F. Hartwich *et al.*, “CAN with flexible data-rate,” in *Proc. iCC*, 2012, pp. 1–9.

[31] The EVITA Project, [Online] Available: http://evita-project/org, 2008.