Computers, Materials & Continua

Tech Science Press

Doi:10.32604/cmc.2025.065560

ARTICLE

CD-AKA-IoV: A Provably Secure Cross-Domain Authentication and Key Agreement Protocol for Internet of Vehicle

Tsu-Yang Wu[1,2], Haozhi Wu[2], Maoxin Tang[3], Saru Kumari[4] and Chien-Ming Chen[1,*]

[1]School of Artificial Intelligence/School of Future Technology, Nanjing University of Information Science and Technology, Nanjing, 210044, China
[2]College of Computer Science and Engineering, Shandong University of Science and Technology, Qingdao, 266590, China
[3]School of Computer Science, Nanjing University of Information Science and Technology, Nanjing, 210044, China
[4]Department of Mathematics, Chaudhary Charan Singh University, Meerut, 250004, India

*Corresponding Author: Chien-Ming Chen. Email: chienmingchen@ieee.org

Received: 16 March 2025; Accepted: 19 June 2025; Published: 29 August 2025

ABSTRACT: With the rapid development and widespread adoption of Internet of Things (IoT) technology, the innovative concept of the Internet of Vehicles (IoV) has emerged, ushering in a new era of intelligent transportation. Since vehicles are mobile entities, they move across different domains and need to communicate with the Roadside Unit (RSU) in various regions. However, open environments are highly susceptible to becoming targets for attackers, posing significant risks of malicious attacks. Therefore, it is crucial to design a secure authentication protocol to ensure the security of communication between vehicles and RSUs, particularly in scenarios where vehicles cross domains. In this paper, we propose a provably secure cross-domain authentication and key agreement protocol for IoV. Our protocol comprises two authentication phases: intra-domain authentication and cross-domain authentication. To ensure the security of our protocol, we conducted rigorous analyses based on the ROR (Real-or-Random) model and Scyther. Finally, we show in-depth comparisons of our protocol with existing ones from both security and performance perspectives, fully demonstrating its security and efficiency.

KEYWORDS: Authentication; key agreement; IoV; cross-domain

1 Introduction

With the Internet of Things (IoT) technology and artificial intelligence continuing to evolve and gain popularity, the field of Internet of Vehicles (IoV) [1,2] has gained significant prominence. IoV enables real-time monitoring and early warning of traffic information through data interaction and information sharing between vehicles and infrastructure, reducing the occurrence of traffic accidents. In the IoV environments, on-board units (OBU) installed in vehicles play a crucial role in exchanging messages with roadside units (RSU) deployed on both sides of the road. RSU can transmit the collected messages to passing vehicles to provide real-time traffic information. This information exchange enables vehicles to promptly identify the traffic situations and take necessary measures.

The open nature of the IoV makes messages transmitted over public channels highly susceptible to becoming targets for attackers, posing risks of interception, tampering, and eavesdropping [3–5]. Moreover, attackers can maliciously impersonate legitimate vehicles. This vulnerability jeopardizes communication between vehicles and the RSU, and even potentially compromises users' privacy [6–8]. As a result, ensuring

the communication security of vehicles and RSUs in the IoV environments becomes imperative. To address this concern, researchers have proposed various authentication and key agreement (AKA) protocols [9–11] to enhance the security of communications between vehicles and RSUs.

However, when vehicles move from one domain to another, cross-domain authentication will be a great challenge. Fig. 1 shows the frequently employed cross-domain architecture of the IoV. This architecture comprises three entities: vehicle, RSU, and cloud server (CS). The CS is in charge of registering both vehicles and RSUs, and helps them in the domain to achieve authentication. In recent years, researchers have conducted in-depth research in the IoV, dedicating significant efforts to addre ssing the challenges of cross-domain authentication. Xu et al. [12] devised an authentication protocol aimed at enhancing secure communication among diverse vehicle networks operating within various domains. Yang et al. [13] proposed a multi-domain vehicle authentication architecture based on blockchain technology, which effectively enables cross-domain information sharing among multiple management domains by establishing a distributed trust mechanism. Yu et al. [14] designed an innovative handover authentication protocol that utilizes blockchain to record vehicle information and verify vehicle legitimacy. To address the vulnerability of vehicles to physical attacks, Jiang et al. and Babu et al. [15,16] deployed physically unclonable functions (PUF) [17] in vehicles. Yan et al. [18] devised a certificateless group handover authentication protocol within a 5G vehicular network environment. Chen et al. [19] presented a key transfer authentication protocol that employs confidential computing environments. Their protocol includes a key transmission phase that facilitates the secure exchange of keys between different RSUs within the same fog node (FN) and different RSUs in distinct FNs.

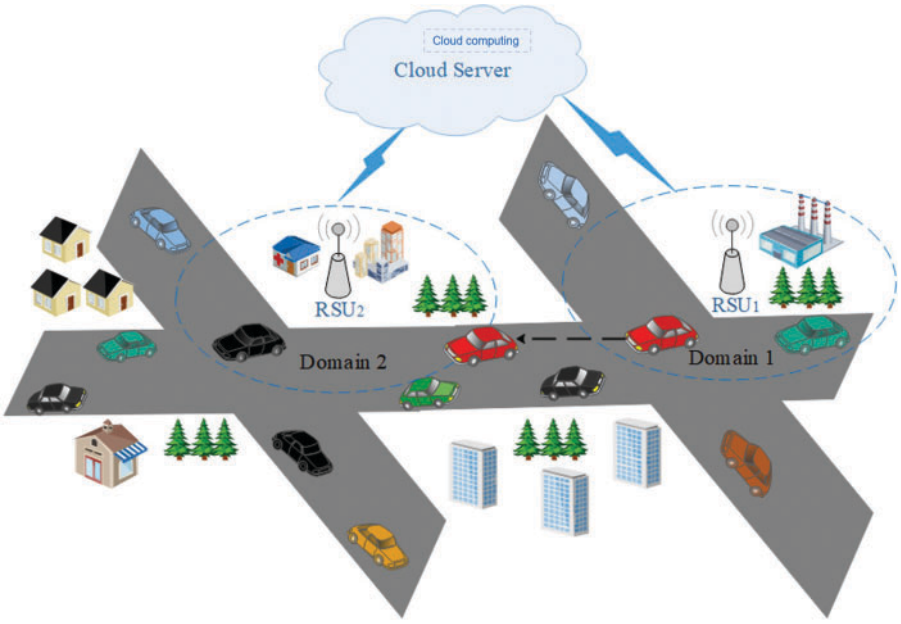

Figure 1: Cross-domain architecture in the IoV

Currently, there is a scarcity of AKA protocols in IoV to ensure secure cross-domain authentication. To enable secure communication when vehicles cross domains, we propose a cross-domain authentication protocol for vehicles in this paper. Building upon the architecture depicted in Fig. 1, we design intra-domain authentication and cross-domain authentication for vehicles. In our design, when a vehicle within a specific domain intends to move to another domain, the cross-domain request needs to be transmitted to the RSU in the destination domain to ensure cross-domain authentication. With the assistance of the CS, the RSU

in the intra-domain can transmit the related cross-domain information about the vehicle to the RSU in the cross-domain. Upon receiving the message, the RSU (cross-domain) searches its database and determines the communication status with the vehicle. If it is the first communication between the vehicle and the RSU (cross-domain), the RSU will request the information of the vehicle from the CS. Otherwise, the vehicle will communicate directly with the RSU (cross-domain). Finally, they can authenticate each other and establish a session key for further encrypting transmitted messages using symmetric encryption. Furthermore, we conducted a systematic analysis and validation of the security of our protocol in the Real-or-Random (ROR) model and Scyther, a security validation tool. The results show that our protocol is secure and resistant to several known attacks. Through a comprehensive comparative analysis of security and performance with other protocols, the results indicate that our protocol has both higher security and performance. The main contributions of this paper are summarized as follows:

1. We propose a provably secure cross-domain authentication protocol for vehicles in IoV environments, which supports secure communication for both intra-domain and cross-domain.
2. We design a handover mechanism that allows the RSU (cross-domain) to determine whether to authenticate vehicles directly or request assistance from the CS.
3. We adopt a two-phase authentication to reduce computation costs by avoiding repeated CS involvement in cross-domain authentication.

The remainder of this paper is structured as follows: Section 2 elaborates on the attacker model and protocol design goals, Section 3 presents our proposed protocol, Section 4 provides the security analysis of the protocol, Section 5 compares the performance of the protocols, and Section 6 provides a summary.

2 System Model, Attacker Model and Design Goals

In this section, we provide a comprehensive description of the system model and the attacker's capabilities. Moreover, we outline the essential security requirements that must be fulfilled by a cross-domain authentication protocol.

2.1 System Model

The proposed protocol consists of three entities: the vehicle V_i, the roadside unit RSU_j, and the cloud server *CS*. Fig. 2 illustrates the overall architecture of the system model.

1. V_i: As a semi-trusted device, V_i is equipped with OBU and Intel software guard extensions (SGX) [20–22]. SGX offers a trusted execution environment, preventing attackers from accessing or altering data stored within SGX.
2. RSU_j: As a semi-trusted device, RSU_j is similarly equipped with Intel SGX. The RSU_j is responsible for collecting real-time traffic information and periodically uploading it to the *CS*. RSU_j is typically deployed on the sides of the road and provides interactions with V_i.
3. *CS*: *CS* is the primary computing and storage resource provider. *CS* serves as the registration center and responds to the registration for V_i and RSU_j. Here, we consider it a trusted entity.

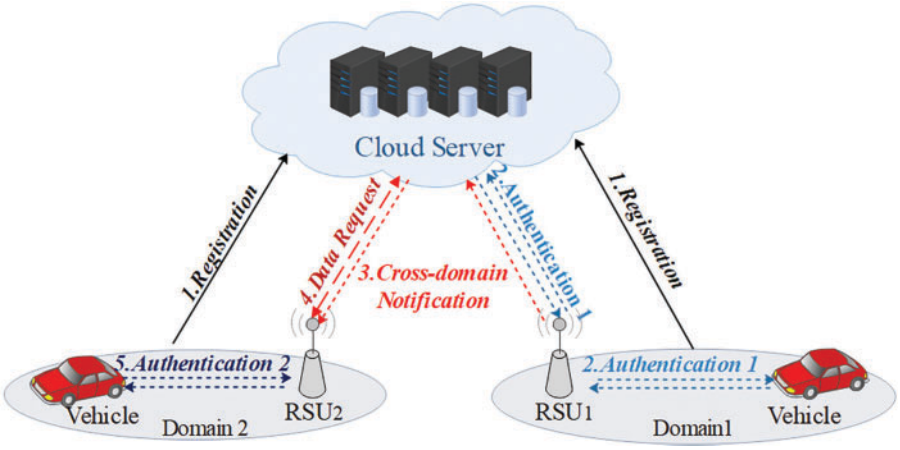

Figure 2: System model

The cross-domain authentication of the vehicle from Domain 1 to Domain 2 is depicted in Fig. 2, and the comprehensive description of each step is provided below.

1. Initialization Phase: In this phase, *CS* pre-allocates unique identities and private keys for each V_i and RSU_j. Then, V_i and RSU_j finish the registration to *CS*.
2. Intra-domain Authentication Phase: In this phase, when V_i needs to communicate with RSU_1 in Domain 1, they will complete mutual authentication and negotiate a session key (SK_1) with the help of *CS*.
3. Cross-domain Notification and Data Request Phase: Upon receiving this request, RSU_1 generates a V_i's cross-domain message called M and sends it to *CS*. Upon receiving M, *CS* combines M with some private values to generate V_i's cross-domain message called M'. Subsequently, *CS* assists in sending M' to RSU_2. Upon receiving M', RSU_2 will utilize it for subsequent authentication of V_i.
4. Cross-domain Authentication Phase: When V_i moves to Domain 2 and wants to communicate with RSU_2, they can authenticate each other and establish SK_2 using M' without the participation of *CS*.

2.2 Attacker Model

In this paper, the capabilities of the attacker ($\mathcal{A}$) are defined based on the widely recognized Dolev-Yao (DY) [23] model and the Canetti-Krawczyk (CK) [24] model. Specifically, the attacker $\mathcal{A}$ possesses the following capabilities:

(1) The attacker $\mathcal{A}$ can intercept, tamper with, and replay information transmitted over public channels.
(2) The attacker $\mathcal{A}$ can extract the data stored in the OBU of the vehicle V_i, but cannot access the data within its SGX.
(3) The attacker $\mathcal{A}$ can obtain data stored in the RSU_j's database, but cannot access the data within its SGX.
(4) The attacker $\mathcal{A}$ is capable of capturing random numbers generated by any entity.
(5) The attacker $\mathcal{A}$ cannot acquire the long-term key of *CS*.
(6) The attacker $\mathcal{A}$ is allowed to obtain any one of (2), (3), or (4) individually, but cannot obtain all of them simultaneously.

2.3 Design Goals

Based on the attacker model described above, our protocol is designed to meet the following security requirements:

(1) Perfect Forward Secrecy (PFS): Even if an attacker $\mathcal{A}$ eventually compromises the long-term secret key of an entity, they should not be able to derive any previously established session keys.
(2) Resistance to Impersonation Attacks: The protocol must prevent an attacker $\mathcal{A}$ from successfully impersonating a legitimate entity (V_i or RSU_j).
(3) Resistance to Man-in-the-Middle (MITM) Attacks: The protocol should ensure that an attacker $\mathcal{A}$ who positions themselves between two communicating parties cannot intercept, alter, or forge messages without being detected.
(4) Resistance to Capture Attacks: Even if $\mathcal{A}$ compromises and gains access to the internal storage of a legitimate entity (V_i or RSU_j), they should be unable to deduce the session key.
(5) Mutual Authentication: This requirement necessitates mutual authentication between entities. In this process, the entities involved in the communications must present corresponding authentication credentials to ensure the legitimacy and trustworthiness.
(6) Anonymity: Entities maintain anonymity throughout the AKA process, ensuring that their true identity and other sensitive information remain undisclosed.
(7) Untraceability: It is impossible to trace or identify the identity or activities of V_i during the AKA process.

3 The Proposed Protocol

This section elaborates on the various phases of the proposed protocol. For ease of understanding, Table 1 lists the symbols used in this paper along with their specific meanings.

Table 1: Notations table

Symbol	Description
ID_i, ID_{RSU_j}	Identities of V_i, RSU_j
TID_i, PID_i	Temporary identity and pseudo identity of V_i
PID_{RSU_j}	Pseudo identity of RSU_j
$E_y(\cdot)/D_y(\cdot)$	Symmetric encryption/decryption algorithm using key y

3.1 Initialization Phase

Before the registration phase, *CS* uses a random number generator to generate its master key K and selects a one-way anti-collision hash function denoted $h(\cdot)$. Note that K is kept secure by *CS*. When V_i and RSU_j are produced by the factory, V_i will be assigned a private key x_i and a long identity ID_i, where x_i is stored in V_i's SGX. Similarly, RSU_j is assigned a long identity ID_{RSU_j}, which is stored in RSU_j's SGX.

3.1.1 V_i Registration Phase

The specific steps for this phase are elaborated in the following.

(1) V_i first selects a random number r_i and calculates $PID_i = h(ID_i \parallel r_i)$. Subsequently, V_i transmits $\{ID_i, PID_i\}$ to *CS* through a secure channel.

(2) After CS receives $\{ID_i, PID_i\}$, it selects r_{cs_i} and calculates $TID_i = h(ID_i \| r_{cs_i} \| K)$, $B_{i1} = h(PID_i \| TID_i \| K)$. Finally, CS stores $\{TID_i, PID_i\}$ in the database and transmits $\{TID_i, B_{i1}\}$ to V_i.

(3) After V_i receives $\{TID_i, B_{i1}\}$, it calculates $B_{i2} = r_i \oplus h(ID_i \| x_i)$, $B_{i3} = B_{i1} \oplus h(r_i \| x_i)$, $B_{i4} = h(ID_i \| B_{i1} \| r_i)$, and $B_{i5} = ID_i \oplus h(TID_i \| x_i)$. Then, V_i stores $\{B_{i2}, B_{i3}, B_{i4}, B_{i5}, TID_i\}$ into OBU.

3.1.2 RSU_j Registration Phase

The specific steps for this phase are detailed below.

(1) RSU_j transmits the identity $\{ID_{RSU_j}\}$ to CS via a secure channel.

(2) After CS receives $\{ID_{RSU_j}\}$, it selects r_{cs_j} and calculates $PID_{RSU_j} = h(ID_{RSU_j} \| r_{cs_j} \| K)$, $K_{RSU_j-CS} = h(ID_{RSU_j} \| PID_{RSU_j} \| K)$, $C_{RSU_j} = ID_{RSU_j} \oplus h(PID_{RSU_j} \| K)$. Finally, CS stores $\{PID_{RSU_j}, C_{RSU_j}\}$ in the database and transmits $\{PID_{RSU_j}, K_{RSU_j-CS}\}$ to RSU_j.

(3) After receiving the message, RSU_j stores $\{PID_{RSU_j}, K_{RSU_j-CS}\}$ in the database.

3.2 Intra-Domain Authentication Phase

In this phase, V_i and RSU_1 achieve authentication and establish the session key SK_1 with the help of CS. The specific process is shown in Fig. 3, and the detailed steps are as follows:

(1) V_i calculates $ID_i = B_{i5} \oplus h(TID_i \| x_i)$, $r_i = B_{i2} \oplus h(ID_i \| x_i)$, $B_{i1} = B_{i3} \oplus h(ID_i \| r_i)$, and checks $B_{i4} \stackrel{?}{=} h(ID_i \| B_{i1} \| r_i)$. If true, V_i login is successful. Then, it selects random number R_i and timestamp T_1, and calculates $R_i' = R_i \oplus h(PID_i \| B_{i1})$, $VE_1 = h(TID_i \| R_i \| T_1)$. Finally, V_i transmits $M_{11} = \{PID_i, R_i', VE_1, T_1\}$ to RSU_1 via a public channel.

(2) After receiving M_{11}, RSU_1 checks T_1. If the time is not exceeded, RSU_1 selects random number R_j, timestamp T_2 and calculates $R_j' = R_j \oplus h(ID_{RSU_1} \| PID_{RSU_1})$, $VE_2 = h(PID_{RSU_1} \| R_j \| K_{RSU_1-CS} \| T_2)$. Finally, RSU_1 sends $M_{12} = \{PID_i, R_i', VE_1, PID_{RSU_1}, R_j', VE_2, T_1, T_2\}$ to CS.

(3) After receiving M_{12}, CS checks T_2. If the time is not exceeded, CS finds TID_i in the validation table according to PID_i, and then calculates $B_{i1} = h(PID_i \| TID_i \| K)$, $R_i = R_i' \oplus h(PID_i \| B_{i1})$, and checks $VE_1 \stackrel{?}{=} h(TID_i \| R_i \| T_1)$. After successfully verifying V_i, CS finds C_{RSU_1} in the verification table according to PID_{RSU_1}, and then calculates $ID_{RSU_1} = C_{RSU_1} \oplus h(PID_{RSU_1} \| K)$, $R_j = R_j' \oplus h(ID_{RSU_1} \| PID_{RSU_1})$, $K_{RSU_1-CS} = h(ID_{RSU_1} \| PID_{RSU_1} \| K)$, and checks $VE_2 \stackrel{?}{=} h(PID_{RSU_1} \| R_j \| K_{RSU_1-CS} \| T_2)$. After successfully verifying RSU_1, CS selects T_3, calculates $G_i = (ID_{RSU_1} \| R_j) \oplus h(TID_i \| R_i)$, $G_j = (TID_i \| R_i) \oplus h(K_{RSU_1-CS} \| R_j)$, $VE_3 = h(K_{RSU_1-CS} \| ID_{RSU_1} \| T_3)$, and sends $M_{13} = \{G_i, G_j, VE_3, T_3\}$ to RSU_1.

(4) After receiving M_{13}, RSU_1 checks T_3 and $VE_3 \stackrel{?}{=} h(K_{RSU_1-CS} \| ID_{RSU_1} \| T_3)$. After successfully verifying CS, RSU_1 can calculate a session key $SK_1 = h(TID_i \| ID_{RSU_1} \| R_i \| R_j)$, where $(TID_i \| R_i) = G_j \oplus h(K_{RSU_1-CS} \| R_j)$. Then, RSU_1 selects T_4, calculates $VE_4 = h(TID_i \| ID_{RSU_1} \| SK_1 \| T_4)$, and sends $M_{14} = \{G_i, VE_4, T_4\}$ to V_i.

(5) After receiving M_{14}, V_i checks T_4. If the time is not exceeded, V_i calculates $SK_1 = h(TID_i \| ID_{RSU_1} \| R_i \| R_j)$, where $(PID_{RSU_1} \| R_j) = G_i \oplus h(TID_i \| R_i)$. Finally, V_i verifies $VE_4 \stackrel{?}{=} h(TID_i \| PID_{RSU_1} \| SK_1 \| T_4)$. If verification is true, V_i sets SK_1 as a session key, which is used to encrypt transmitted messages communicating with RSU_1.

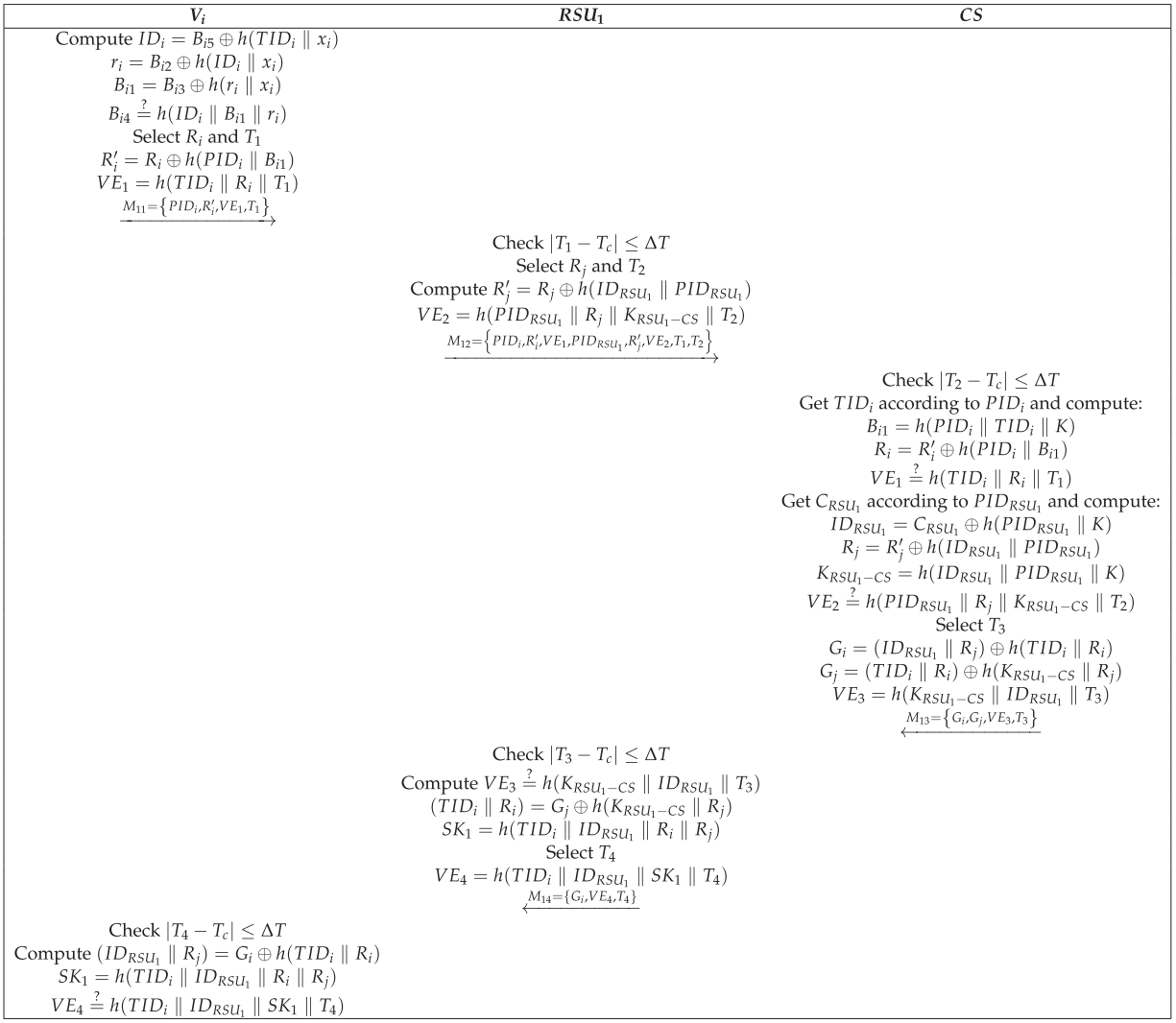

Figure 3: Intra-domain authentication phase

3.3 Cross-Domain Notification and Data Request Phase

In this phase, vehicle V_i wants to move from the jurisdiction domain RSU_1 to that of RSU_2. The specific process is shown in Fig. 4, and the detailed steps are as follows:

(1) RSU_1 generates a notification Not_j and calculates $P_1 = E_{ID_{RSU_1}}(PID_{RSU_2} \| PID_i)$, $P_2 = h(K_{RSU_1-CS} \| ID_{RSU_1} \| Not_j \| T_5)$, where T_5 is the current timestamp. Finally, RSU_1 sends $M_{21} = \{Not_j, PID_{RSU_1}, P_1, P_2, T_5\}$ to CS.

(2) After receiving M_{21}, CS first checks T_5. If the time limit is not exceeded, CS gets C_{RSU_1} according to PID_{RSU_1} and checks $P_2 \stackrel{?}{=} h(K_{RSU_1-CS} \| ID_{RSU_1} \| Not_j \| T_5)$, where $ID_{RSU_1} = C_{RSU_1} \oplus h(PID_{RSU_1} \| K)$. After successfully verifying RSU_1, CS selects T_6 and calculates $P_3 = h(K_{RSU_2-CS} \| ID_{RSU_2} \| Not_j \| T_6)$, where $K_{RSU_2-CS} = h(ID_{RSU_2} \| PID_{RSU_2} \| K)$, $ID_{RSU_2} = C_{RSU_2} \oplus h(PID_{RSU_2} \| K)$, C_{RSU_2} is according to PID_{RSU_2}, and $PID_{RSU_2} \| PID_i = D_{ID_{RSU_1}}(P_1)$. Finally, CS sends $M_{22} = \{Not_j, PID_i, P_3, T_6\}$ to RSU_2.

(3) After receiving M_{22}, RSU_2 first checks T_6 and $P_3 \stackrel{?}{=} h(K_{RSU_2-CS} \| ID_{RSU_2} \| Not_j \| T_6)$. If the verifications are valid, RSU_2 uses PID_i to retrieve H_i from the database to determine whether V_i has communicated with itself before. If not, RSU_2 generates a data request Req_j and $r'_j = r_j \oplus h(PID_{RSU_2} \| ID_{RSU_2})$, $P_4 = h(K_{RSU_2-CS} \| ID_{RSU_2} \| Req_j \| r_j \| T_7)$, where r_j is a random number and T_7 is the current timestamp. Finally, RSU_2 sends $M_{23} = \{Req_j, r'_j, P_4, T_7\}$ to CS.

(4) After receiving M_{23}, CS first checks T_7. If the time limit is not exceeded, CS calculates $r_j = r'_j \oplus h(PID_{RSU_2} \| ID_{RSU_2})$, and checks $P_4 \stackrel{?}{=} h(K_{RSU_2-CS} \| ID_{RSU_2} \| Req_j \| r_j \| T_7)$. After successfully verifying RSU_2, CS gets $\{TID_i\}$ according to $\{PID_i\}$ and calculates $B_{i1} = h(PID_i \| TID_i \| K)$, $H_i = E_{h(ID_{RSU_2} \| K_{RSU_2-CS})}(TID_i \| B_{i1})$. Finally, CS calculates $P_5 = h(TID_i \| PID_{RSU_2} \| T_8)$ and sends $M_{24} = \{H_i, P_5, T_8\}$ to RSU_2, where T_8 is the current timestamp.

(5) After receiving M_{24}, RSU_2 first checks T_8 and $P_5 \stackrel{?}{=} h(TID_i \| PID_{RSU_2} \| T_8)$, where $TID_i \| B_{i1} = D_{h(ID_{RSU_2} \| K_{RSU_2-CS})}(H_i)$. After successfully verifying CS, RSU_2 computes $Z_j = h(ID_{RSU_2} \| PID_{RSU_2}) \oplus (TID_i \| B_{i1})$ and stores $\{PID_i, Z_j\}$ in its database.

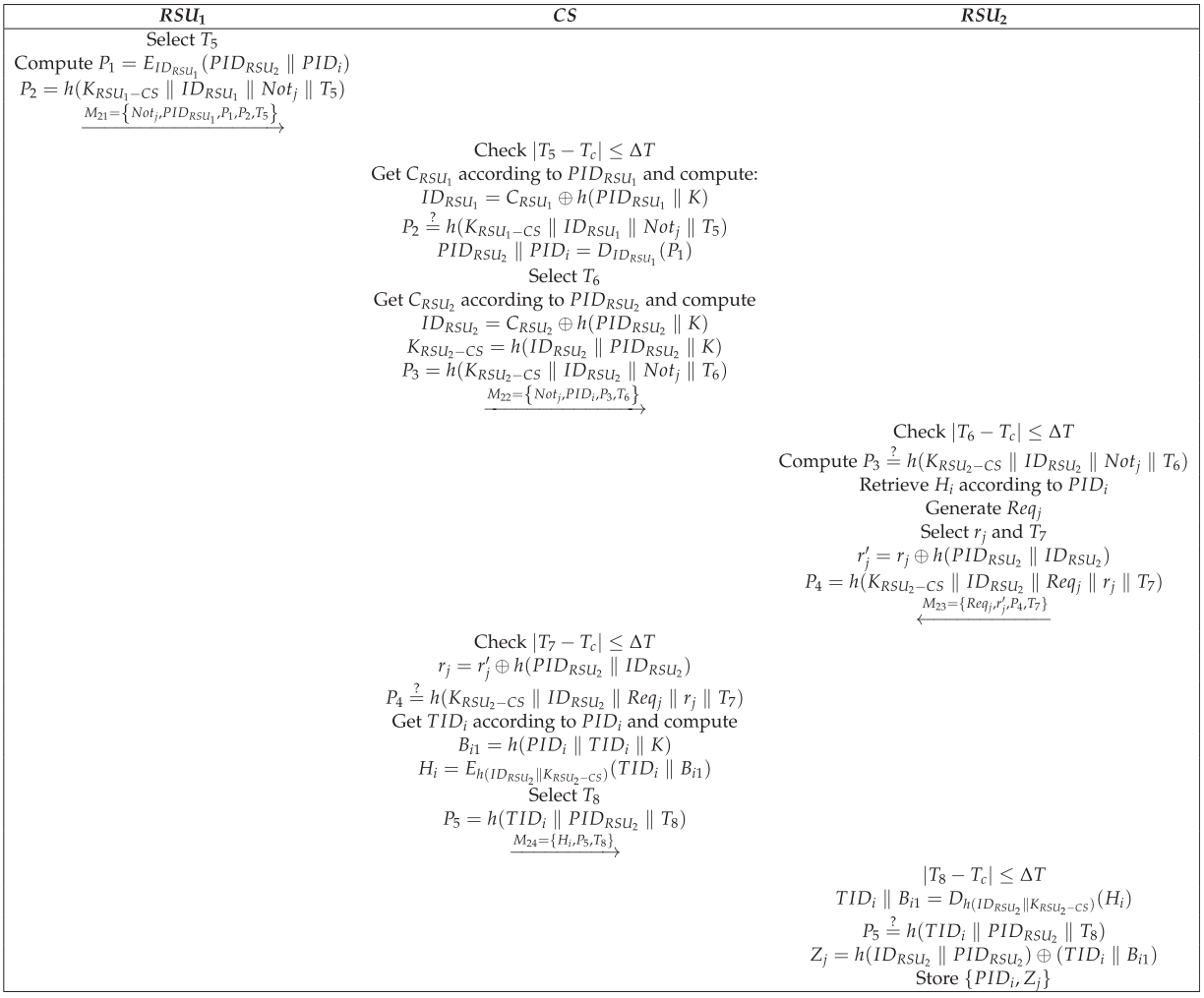

Figure 4: Notification and data transmission phase

3.4 Cross-Domain Authentication Phase

When V_i moves to Domain 2, mutual authentication and session key computation are required with RSU_2.

The specific process is shown in Fig. 5, and the detailed description is as follows:

(1) V_i selects a random number α_i and timestamp T_9, calculates $\alpha_i' = \alpha_i \oplus h(TID_i \| B_{i1})$, $N_1 = h(PID_i \| \alpha_i \| T_9)$. Finally, V_i sends $M_{31} = \{PID_i, \alpha_i', N_1, T_9\}$ to RSU_2.

(2) After receiving M_{31}, RSU_2 first checks T_9. If the time limit is not exceeded, RSU_2 gets $\{Z_j\}$ from its database according to PID_i and checks $N_1 \stackrel{?}{=} h(TID_i \| \alpha_i \| T_9)$, where $TID_i \| B_{i1} = Z_j \oplus h(ID_{RSU_2} \| PID_{RSU_2})$, $\alpha_i = \alpha_i' \oplus h(PID_i \| B_{i1})$. After successfully verifying V_i, RSU_2 selects a random number β_j, timestamp T_{10}, calculates $L_i = (ID_2 \| \beta_j) \oplus h(PID_i \| \alpha_i)$, session key $SK_2 = h(TID_i \| ID_2 \| \alpha_i \| \beta_j)$, $N_2 = h(TID_i \| \beta_j \| SK_2 \| T_{10})$. Finally, RSU_2 sends $M_{32} = \{L_i, T_1, T_{10}\}$ to V_i.

(3) After receiving M_{32}, V_i first checks T_{10}. If the time limit is not exceeded, V_i calculates $ID_{RSU_2} \| \beta_j = L_i \oplus h(TID_i \| \alpha_i)$, $SK_2 = h(TID_i \| ID_2 \| \alpha_i \| \beta_j)$, and checks $N_2 \stackrel{?}{=} h(TID_i \| \beta_j \| SK_2 \| T_{10})$. If verification is true, V_i sets SK_2 as a session key, which is used to encrypt transmitted messages communicating with RSU_2.

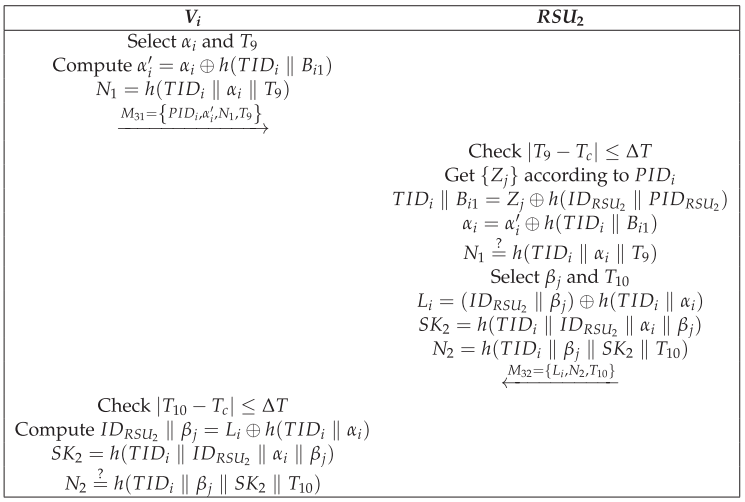

Figure 5: Cross-domain authentication phase

4 Security Analysis

4.1 Formal Security Analysis

In this section, we focus on verifying the session key security under the ROR model, a well-established analytical framework referred to [17,25,26], for proving that an authentication protocol achieves the semantic security of the session key. The protocol involves three primary participants: $\Pi_{V_i}^x$ (the x-th instance of V_i), $\Pi_{RSU_j}^y$ (the y-th instance of RSU_j), and Π_{CS}^z (the z-th instance of CS). The adversary $\mathcal{A}$ is permitted to perform both passive and active attacks through the execution of a set of standard ROR queries.

(1) $Execute(\mathcal{E})$: When $\mathcal{A}$ executes this query, it can intercept messages transmitted between entities $\mathcal{E} = \{\Pi_{V_i}^x, \Pi_{RSU_j}^y, \Pi_{CS}^z\}$.

(2) $Send(\mathcal{E}, M_i)$: This query means that $\mathcal{A}$ sends a message M_i to $\mathcal{E}$ and then receives a reply.

(3) $Hash(string)$: By entering a string, $\mathcal{A}$ can obtain its corresponding hash value.
(4) $Corrupt(\Pi^x_{V_i}, \Pi^y_{RSU_j})$: When this query is executed, $\mathcal{A}$ can extract the values stored in the vehicle's OBU or RSU's database.
(5) $Reveal(\mathcal{E})$: When this query is executed, $\mathcal{A}$ can receive session key SK_1 or SK_2.
(6) $Test(\mathcal{E})$: The game starts with $\mathcal{A}$ to guess the flipping coin $\mathscr{C}$. If $\mathscr{C} = 1$, $\mathcal{E}$ returns session keys SK_1 and SK_2. Otherwise, $\mathcal{E}$ returns a random string.

Based on the ROR model, the security of our protocol can be proved by the following theorem.

Theorem 1: *Based on the ROR model, the advantage of probabilistic polynomial-time ξ, adversary $\mathcal{A}$ in breaking our protocol $\mathcal{P}$ is bounded by:* $Adv^{\mathcal{P}}_{\mathcal{A}}(\xi) \leq \frac{q_h^2}{|Hash|} + 2Adv^{\Omega}_{\mathcal{A}}(k) + \frac{2q_s}{|D|}$. *Here, q_s and q_h denote the hash and send queries respectively,* $Adv^{\Omega}_{\mathcal{A}}(k)$ *denotes the advantage of $\mathcal{A}$ in cracking symmetric encryption algorithm Ω, $|D|$ and $|Hash|$ indicate the space of dictionary and hash function.*

Proof: We define four rounds of games, denoted as GM_0-GM_3, to simulate $\mathcal{A}$ in breaking our protocol. Here, the event in which $\mathcal{A}$ wins in GM_i is denoted by $Succ^{GM_i}_{\mathcal{A}}(\xi)$. In Table 2, we demonstrate that $\mathcal{A}$ simulates detailed queries.

Table 2: Simulation of oracles

Query	Description
$Send(\mathcal{E}, M_i)$	For $Send(\Pi^x_{V_i}, \text{start})$. Π_{V_i} selects R_i, and T_1 to compute $R'_i = R_i \oplus h(PID_i \| B_{i1})$, $VE_i = h(TID_i \| R_i \| T_1)$. Next, the query returns the $M_{11} = \{PID_i, R'_i, VE_1, T_1\}$.
	On $Send(\Pi^y_{RSU_j}, (PID_i, R'_i, VE_1, T_1))$. Assume that $\Pi^y_{RSU_j}$ checks T_1 in a normal state. Then, $\Pi^y_{RSU_j}$ selects R_j, and T_2. Next, $\Pi^y_{RSU_j}$ calculates R'_j, VE_2. The query is answered by $M_{12} = \{PID_i, R'_i, VE_1, PID_{RSU_1}, R'_j, VE_2, T_1, T_2\}$.
	For $Send(\Pi^z_{CS}, (PID_i, R'_i, VE_1, PID_{RSU_1}, R'_j, VE_2, T_1, T_2))$. On receiving the message $\{PID_i, R'_i, VE_1, PID_{RSU_1}, R'_j, VE_2, T_1, T_2\}$, Π_{CS} computes $B_{i1}, R_i, ID_{RSU_1}, R_j, K_{RSU_1-CS}$ and checks the VE_1, VE_2. Then, Π^z_{CS} calculates G_i, G_j, VE_3. Next, Π^z_{CS} returns the output $M_{13} = \{G_i, G_j, VE_3, T_3\}$.
	For $Send(\Pi^y_{RSU_j}, (G_i, G_j, VE_3, T_3))$. $\Pi^y_{RSU_j}$ checks VE_3 in a normal state. If the VE_3 holds, $\Pi^y_{RSU_j}$ calculates TID_i, R_i, SK_1, VE_4 and selects T_4. Then, the query returns the $M_{14} = \{G_i, VE_4, T_4\}$.
	On $Send(\Pi^x_{V_i}, (G_i, VE_4, T_4))$. Upon receiving the message (G_i, VE_4, T_4), $\Pi^x_{V_i}$ computes ID_{RSU_1}, R_j, SK_1 and checks V_4.
$Execute(\mathcal{E})$	On a query $Execute(\mathcal{E})$, we use $Send$ queries to simulate it. $(PID_i, R'_i, VE_1, T_1) \longleftarrow Send(\Pi^x_{V_i}, \text{start})$, $(PID_i, R'_i, VE_1, PID_{RSU_1}, R'_j, VE_2, T_1, T_2) \longleftarrow Send(\Pi^y_{RSU_j}, (PID_i, R'_i, VE_1, T_1))$, $(G_i, G_j, VE_3, T_3) \longleftarrow Send(\Pi^z_{CS}, (PID_i, R'_i, VE_1, PID_{RSU_1}, R'_j, VE_2, T_1, T_2))$, $(G_i, VE_4, T_4) \longleftarrow Send(\Pi^y_{RSU_j}, (G_i, G_j, VE_3, T_3))$. The query returns (PID_i, R'_i, VE_1, T_1), $(PID_i, R'_i, VE_1, PID_{RSU_1}, R'_j, VE_2, T_1, T_2)$, (G_i, G_j, VE_3, T_3) and (G_i, VE_4, T_4).

(Continued)

Table 2 (continued)

Query	Description
$Corrupt(\Pi^x_{V_i}, \Pi^y_{RSU_j})$	If $\Pi^x_{V_i}$ is in an accepted state, this query returns the values stored in the vehicle's OBU: $\{B_{i2}, B_{i3}, B_{i4}, B_{i5}, TID_i\}$. If $\Pi^y_{RSU_j}$ is in an accepted state, it returns the RSU's stored data: $\{PID_{RSU_j}, K_{RSU_j-CS}\}$.

GM_0: It simulates a real attack on the scheme, $\mathcal{A}$ does not execute any query. Therefore, we can get

$$Adv^{\mathcal{P}}_{\mathcal{A}}(\xi) = |2Pr[Succ^{GM_0}_{\mathcal{A}}(\xi)] - 1|. \tag{1}$$

GM_1: During this game, $\mathcal{A}$ endeavors to perform session key disclosure attacks. $\mathcal{A}$ executes $Execute(\mathcal{E})$ query to obtain messages $M_{11} = \{PID_i, R'_i, VE_1, T_1\}$, $M_{12} = \{PID_i, R'_i, VE_1, PID_{RSU_1}, R'_j, VE_2, T_1, T_2\}$, $M_{13} = \{G_i, G_j, VE_3, T_3\}$, $M_{14} = \{G_i, VE_4, T_4\}$, $M_{21} = \{Not_j, PID_{RSU_1}, P_1, P_2, T_5\}$, $M_{22} = \{Not_j, PID_i, P_3, T_6\}$, $M_{23} = \{Req_j, r'_j, P_4, T_7\}$ $M_{31} = \{PID_i, \alpha'_i, N_1, T_9\}$, and $M_{32} = \{L_i, N_2, T_{10}\}$. Then, $\mathcal{A}$ executes $Test(\Pi^x_{V_i}, \Pi^y_{RSU_j})$ to check whether its output is the real session key. According to our protocol, $SK_1 = h(TID_i \parallel ID_{RSU_1} \parallel R_i \parallel R_j)$ and $SK_2 = h(TID_i \parallel ID_{RSU_2} \parallel \alpha_i \parallel \beta_j)$, which has confidential information such as $TID_i, ID_{RSU_1}, ID_{RSU_2}$ and random values $R_i, R_j, \alpha_i, \beta_j$. These secret values cannot be obtained from the messages, so the probability of $\mathcal{A}$ guessing $\mathscr{C}$ has not increased. Therefore, we can get the same result as GM_0.

$$Pr[Succ^{GM_1}_{\mathcal{A}}(\xi)] = Pr[Succ^{GM_0}_{\mathcal{A}}(\xi)]. \tag{2}$$

GM_2: In this game, the adversary $\mathcal{A}$ is additionally granted access to the $Send(\mathcal{E})$ and $Hash(\mathcal{E})$ oracles, thereby enabling active attacks such as message tampering. $\mathcal{A}$ can run multiple hash queries to check for collisions, it also tries to fool participants into accepting forged messages. However, the authentication values $\{VE_1, VE_2, VE_3, VE_4, P_1, P_2, P_3, P_4, P_5, N_1, N_2\}$ are composed of secret credentials and random numbers, and are securely protected by a one-way hash function. At this time, $\mathcal{A}$ attempts to decrypt the ciphertext $H_i = E_{h(ID_{RSU_2} \| K_{RSU_2-CS})}(TID_i \parallel B_{i1})$, which is generated using a secure symmetric encryption algorithm. Since $\mathcal{A}$ does not possess the secret values ID_{RSU_2} and K_{RSU_2-CS}, the encryption key $h(ID_{RSU_2} \parallel K_{RSU_2-CS})$ cannot be computed. Consequently, the plaintext components TID_i, B_{i1} remain inaccessible, and it is computationally infeasible for $\mathcal{A}$ to infer the session keys SK_1 or SK_2. Therefore, according to the birthday paradox and the security of the symmetric encryption algorithm, we can get

$$|Pr[Succ^{GM_2}_{\mathcal{A}}(\xi)] - Pr[Succ^{GM_1}_{\mathcal{A}}(\xi)]| \leq \frac{q_h^2}{2|Hash|} + Adv^{\Omega}_{\mathcal{A}}(k). \tag{3}$$

GM_3: In this game, $\mathcal{A}$ attempts to perform OBU capture attacks and RSU capture attacks. $\mathcal{A}$ issues the $Corrupt(\Pi^x_{V_i})$ query and obtains the stored data from the OBU, which includes $\{B_{i2}, B_{i3}, B_{i4}, B_{i5}, TID_i\}$, where the values are defined as follows $B_{i2} = r_i \oplus h(ID_i \parallel x_i), B_{i3} = B_{i1} \oplus h(r_i \parallel x_i), B_{i4} = h(ID_i \parallel B_{i1} \parallel r_i), B_{i5} = ID_i \oplus h(TID_i \parallel x_i)$, and x_i is a long-term private key securely stored within the V_i's SGX. In addition, $\mathcal{A}$ can also eavesdrop on all messages $\{G_i, VE_4, T4\}$ and $\{L_i, N_2, T_{10}\}$ transmitted on the communication channel. If $\mathcal{A}$ wants to calculate $SK_1 = h(TID_i \parallel ID_{RSU_1} \parallel R_i \parallel R_j)$ or $SK_2 = h(TID_i \parallel ID_{RSU_2} \parallel \alpha_i \parallel \beta_j)$, it must first recover B_{i1}. However, $\mathcal{A}$ must use r_i and x_i to get B_{i1}. Similarly, if $\mathcal{A}$ issues the $Corrupt(\Pi^y_{RSU_j})$ query and obtains the stored data form the RSU, which include $\{PID_{RSU_j}, K_{RSU_j-CS}\}$. However, in order to compute SK_1 or SK_2, $\mathcal{A}$ must also know ID_{RSU_j}, which is securely stored within the

RSU's SGX and is not revealed to the adversary. Therefore, $\mathcal{A}$ may attempt to guess x_i or ID_{RSU_j} from a finite key dictionary of size $|D|$. As a result, the advantage gained in this game compared to GM_2 is bounded by

$$|Pr[Succ_{\mathcal{A}}^{GM_3}(\xi)] - Pr[Succ_{\mathcal{A}}^{GM_2}(\xi)]| \leq \frac{q_s}{|D|}. \tag{4}$$

Finally, $\mathcal{A}$ attempts to distinguish the real session key from a random value by issuing the $Test(\Pi_{V_i}^x, \Pi_{RSU_j}^y)$ query. Since all sensitive information required to compute the session key is secure or computationally infeasible, $\mathcal{A}$ has no advantage over random guessing. Therefore, the adversary's success probability in this game is exactly

$$Pr[Succ_{\mathcal{A}}^{GM_3}(\xi)] = \frac{1}{2}. \tag{5}$$

Given the results from Eqs. (1) through (5), we obtain an upper bound of the advantage

$$\begin{aligned}
\frac{Adv_{\mathcal{A}}^{\mathcal{P}}(\xi)}{2} &= |Pr[Succ_{\mathcal{A}}^{GM_0}(\xi)] - \frac{1}{2}| \\
&= |Pr[Succ_{\mathcal{A}}^{GM_0}(\xi)] - Pr[Succ_{\mathcal{A}}^{GM_3}(\xi)]| \\
&= |Pr[Succ_{\mathcal{A}}^{GM_1}(\xi)] - Pr[Succ_{\mathcal{A}}^{GM_3}(\xi)]| \\
&\leq \sum_{i=1}^{2} |Pr[Succ_{\mathcal{A}}^{GM_{i+1}}(\xi)] - Pr[Succ_{\mathcal{A}}^{GM_i}(\xi)]| \\
&\leq \frac{q_h^2}{2|Hash|} + Adv_{\mathcal{A}}^{\Omega}(k) + \frac{q_s}{|D|}.
\end{aligned} \tag{6}$$

Therefore, the overall advantage of adversary $\mathcal{A}$ in breaking protocol $\mathcal{P}$ is bounded by

$$Adv_{\mathcal{A}}^{\mathcal{P}}(\xi) \leq \frac{q_h^2}{|Hash|} + 2Adv_{\mathcal{A}}^{\Omega}(k) + \frac{2q_s}{|D|}, \tag{7}$$

where $Adv_{\mathcal{A}}^{\Omega}(k)$ is negligible in the security parameter k under a secure symmetric encryption algorithm Ω.

□

4.2 Security Verification Using Scyther

In this section, we adopt Scyther to model and analyze the security of the proposed protocol, including intra-domain and cross-domain authentications. Scyther is a security validation tool based on the D-Y model and performs symbolic execution to automatically detect potential attack paths within defined bounds. Here, we define a range of security claims, including the secrecy of session keys and authentication credentials, identity agreement, and aliveness of participating entities.

In Fig. 6a, it shows the results of the intra-domain authentication in which the entities include vehicle V, intra-domain roadside unit $RSU1$, and cloud server CS. In Fig. 6b, it shows the results of the cross-domain authentication in which the entities include vehicle V and cross-domain roadside unit $RSU2$. The two results show that all the claims are satisfied and no attacks are detected. Meanwhile, all entities achieve mutual authentication and key establishment. This evidence shows that the proposed protocol is capable of resisting common threats, such as man-in-the-middle, replay, and impersonation attacks, in both intra-domain and cross-domain authentications.

Claim				Status	Comments
intra_domain	V	intra_domain,V1	Secret h(cat(TIDi,IDj1,Ri,Rj))	Ok	No attacks within bounds.
		intra_domain,V2	Secret Ri	Ok	No attacks within bounds.
		intra_domain,V3	Secret Bi1	Ok	No attacks within bounds.
		intra_domain,V4	Niagree	Ok	No attacks within bounds.
		intra_domain,V5	Nisynch	Ok	No attacks within bounds.
		intra_domain,V6	Weakagree	Ok	No attacks within bounds.
		intra_domain,V7	Alive	Ok	No attacks within bounds.
	RSU1	intra_domain,R1	Secret h(cat(TIDi,IDj1,Ri,Rj))	Ok	No attacks within bounds.
		intra_domain,R2	Secret Rj	Ok	No attacks within bounds.
		intra_domain,R3	Secret KRj1CS	Ok	No attacks within bounds.
		intra_domain,R4	Niagree	Ok	No attacks within bounds.
		intra_domain,R5	Nisynch	Ok	No attacks within bounds.
		intra_domain,R6	Weakagree	Ok	No attacks within bounds.
		intra_domain,R7	Alive	Ok	No attacks within bounds.
	CS	intra_domain,CS1	Niagree	Ok	No attacks within bounds.
		intra_domain,CS2	Nisynch	Ok	No attacks within bounds.
		intra_domain,CS3	Weakagree	Ok	No attacks within bounds.
		intra_domain,CS4	Alive	Ok	No attacks within bounds.

Done.

(a) Intra-domain authentication.

Claim				Status	Comments
cross_domain	V	cross_domain,V1	Secret h(cat(TIDi,IDj2,ai,bj))	Ok	No attacks within bounds.
		cross_domain,V2	Secret ai	Ok	No attacks within bounds.
		cross_domain,V3	Niagree	Ok	No attacks within bounds.
		cross_domain,V4	Nisynch	Ok	No attacks within bounds.
		cross_domain,V5	Weakagree	Ok	No attacks within bounds.
		cross_domain,V6	Alive	Ok	No attacks within bounds.
	RSU2	cross_domain,R1	Secret h(cat(TIDi,IDj2,ai,bj))	Ok	No attacks within bounds.
		cross_domain,R2	Secret bj	Ok	No attacks within bounds.
		cross_domain,R3	Niagree	Ok	No attacks within bounds.
		cross_domain,R4	Nisynch	Ok	No attacks within bounds.
		cross_domain,R5	Weakagree	Ok	No attacks within bounds.
		cross_domain,R6	Alive	Ok	No attacks within bounds.

Done.

(b) Cross-domain authentication.

Figure 6: Simulation results of Scyther

4.3 Anonymity and Untraceability

In our protocol, the real identity of V_i, ID_i, is ingeniously encapsulated in a pseudo-identity $PID_i = h(ID_i||r_i)$. Similarly, the real identity of RUS_j, ID_{RSU_j}, is ingeniously encapsulated in a pseudo-identity $PID_{RSU_j} = h(ID_{RSU_j}||r_{cs_i}||K)$. Therefore, $\mathcal{A}$ is not able to identify and track them even if PID_i and PID_{RSU_j} are obtained by $\mathcal{A}$.

5 Security and Performance Comparisons

In this section, we conducted a comparative analysis of security and performance between the proposed protocol and authentication protocols in IoV [4,10–13,19].

5.1 Security and Functionality Comparisons

The security and functionality comparison is summarized in Table 3. Note that ✗ indicates that the protocol cannot resist a particular type of attack or does not support the specified feature, while ✓ indicates that the protocol provides protection or support accordingly. It can be observed that Li et al.'s protocol [4] is vulnerable to RSU impersonation attacks and does not provide anonymity and untraceability. Lee et al.'s protocol [10] cannot withstand offline password guessing attacks and lacks mutual authentication. Eftekhar et al.'s protocol [11] violates PFS. On the other hand, Xu et al.'s, Yang et al.'s, Chen et al.'s protocols [12,13,19], and our protocol are resistant to common attacks and both achieve cross-domain authentication. In particular, we design a handover mechanism to reduce computation costs by avoiding the repeated involvement of the cloud server in cross-domain authentication.

Table 3: Comparison of security and functionality

Properties	Xu et al. [12]	Li et al. [4]	Lee et al. [10]	Eftekhari et al. [11]	Yang et al. [13]	Chen et al. [19]	Ours
PFS	✓	✓	✓	✗	✓	✓	✓
RSU impersonation attacks	✓	✗	✓	✓	✓	✓	✓
Mutual authentication	✓	✓	✗	✓	✓	✓	✓
Anonymity	✓	✗	✓	✓	✓	✓	✓
Untraceability	✓	✗	✓	✓	✓	✓	✓
Cross-domain authentication	✓	✗	✗	✗	✓	✓	✓
Without the repeated involvement of CS	✗	✗	✗	✗	✗	✗	✓

5.2 The Comparison of Computational Costs

We are mainly concerned with comparing the computational costs that different protocols used in the authentication phase. Specifically, we conduct experiments to simulate the computational time required by each entity in the protocols in which the XiaoMi 8 smartphone is used to simulate V_i, the Lenovo laptop is used to simulate $RSU(FN)$, and the Lenovo desktop computer is used to simulate $CS(TA)$. The configurations of these devices are detailed in Table 4. Throughout the experiments, each device was executed 80 times, and the average running time was considered the final execution time. Finally, the execution times of each device are presented in Table 5.

Table 4: Equipment configuration

	XiaoMi 8	Lenovo laptop	Lenovo desktop
Operating system	Android system	Windows 10	Windows 10
Running memory	6G	8G	16G
CPU	Qualcomm Snapdragon 845	Intel(R) Core(TM) i7-6700HQ CPU @ 2.60 GHz	Intel(R) Core(TM) i5-9500 CPU @ 3.00 GHz

Table 5: Computational costs for various operations

Operations	Symbolic	XiaoMi 8 (ms)	Lenovo laptop (ms)	Lenovo desktop (ms)
Hash function	T_{hf}	0.0043	0.0031	0.0025
Point addition in ECC	T_{pa}	0.1556	0.0731	0.0503
Symmetric encryption/decryption	T_{se}	0.2269	0.1648	0.1384
Point scalar multiplication in ECC	T_{pm}	20.23	11.4	7.85

Table 6 illustrates the comparison of computational cost results. Yang et al.'s protocol [13] has the highest computational cost across all protocols, due to the use of point scalar multiplication operations in ECC within blockchain-based pseudonym handling and verification processes. Similarly, Eftekhari et al.'s protocol [11] also demonstrates high costs, especially for V_i and RSU_j, due to its use of three times point scalar multiplication operations. In contrast, Xu et al.'s protocol [12] also involves multiple operations in ECC. Its total

cost remains lower than that of Eftekhari et al.'s and Yang et al.'s protocol. For the other protocols [4,10,19], the computational costs are relatively low, primarily based on hash functions and lightweight cryptographic operations. Among them, Chen et al.'s protocol [19] uses symmetric encryption/decryption operation in RSU, resulting in a higher cost than Li et al.'s protocol [4].

Table 6: The comparison of computational cost

Protocols	V_i (ms)	$RSU_j(FN)$ (ms)	$CS(TA)$ (ms)	Total (ms)
Xu et al. [12]	$4T_{hf} + 2T_{se} + 2T_{pm}$ ≈ 40.931	$4T_{hf} + T_{se} + 2T_{pm}$ ≈ 22.9772	$3T_{hf} + T_{se} + T_{pm}$ ≈ 7.9959	71.9041
Li et al. [4]	$6T_{hf} \approx 0.0258$	$4T_{hf} \approx 0.0124$	$12T_{hf} \approx 0.03$	0.0682
Lee et al. [10]	$11T_{hf} \approx 0.0473$	$5T_{hf} \approx 0.0155$	$13T_{hf} \approx 0.0325$	0.0953
Eftekhari et al. [11]	$3T_{pm} + T_{pa} + 11T_{hf}$ ≈ 60.8929	$3T_{pm} + T_{pa} + 12T_{hf}$ ≈ 34.3103	$3T_{pm} + 2T_{pa} + 15T_{hf}$ ≈ 23.6881	118.8913
Yang et al. [13]	$3T_{hf} + 2T_{se} + 4T_{pm}$ ≈ 81.3867	$6T_{hf} + 2T_{se} + 7T_{pm}$ $+2T_{pa} \approx 80.2944$	–	161.6811
Chen et al. [19]	$7T_{hf} \approx 0.0301$	$11T_{hf} + 4T_{se} \approx 0.6933$	$12T_{hf} \approx 0.03$	0.7534
Inter-domain authentication	$9T_{hf} \approx 0.0387$	$6T_{hf} \approx 0.0186$	$10T_{hf} \approx 0.025$	0.0823
Cross-domain authentication	$5T_{hf} \approx 0.0215$	$6T_{hf} \approx 0.0186$	–	0.0401

In the cross-domain authentication phase of our protocol, it achieves the lowest computational cost in V_i and RSU_j with only hash operations. For $CS(TA)$, our protocol requires only a small number of hash operations, resulting in the lowest central computational cost among all protocols. Overall, these results confirm that our protocol achieves lower computational costs, particularly in cross-domain authentication.

5.3 The Comparison of Communicational Costs

We defined the following length of parameters used to evaluate communication costs: identity $|ID|$, ciphertext of symmetric encryption $|E|$, random number $|R|$, timestamp $|T|$, hash function $|H|$, and point $|P|$ as 160, 256, 160, 32, 256, and 320 bits, respectively.

As our protocol consists of two authentication phases, we divided the communication cost into two cases. In case 1 (Inter-domain authentication + notification and data transmission), the transmitted messages comprise $\{M_{11}, M_{12}, M_{13}, M_{14}, M_{21}, M_{22}, M_{23}, M_{24}\}$. The total communication cost is $8|H| + 14|R| + 8|R| + 2|E| = 5056$ bits. In case 2 (cross-domain authentication), the transmitted messages comprise $\{M_{31}, M_{32}\}$. The total communication cost is $3|R| + 2|H| + |T| = 1024$ bits. Using the same methodology, the communication costs of these protocols [4,10–13,19] are calculated as 4000, 2208, 4992, 3616, 5728, and 5216 bits, respectively.

Table 7 illustrates the comparative results. It can be observed that Yang et al.'s protocol [13] incurs the highest communication cost due to its blockchain-based pseudonym exchange among all protocols. In our protocol, Case 1 involves 8 rounds of communications. However, this case occurs only once for the same V_i and RSU. All subsequent cross-domain authentication is executed in Case 2, which achieves only 1024 bits of communicational cost within 2 rounds. It is noteworthy that Case 2 (cross-domain authentication) significantly reduces communication overhead, outperforming all compared protocols.

Table 7: The cost comparison of communication

Protocols	Rounds	Communication cost (bits)
Xu et al. [12]	8	3616
Li et al. [4]	4	4000
Lee et al. [10]	4	2208
Eftekhari et al. [11]	4	4992
Yang et al. [13]	3	5728
Chen et al. [19]	5	5216
Case 1	8	5056
Case 2	2	1024

It is noteworthy that Case 2 significantly reduces communication overhead, outperforming all compared protocols under similar conditions. Through the joint analysis of computational and communicational costs, we claim that our proposed protocol achieves higher performance while maintaining higher security, particularly in cross-domain authentication.

6 Conclusion

In this paper, we have proposed a provably secure cross-domain AKA protocol tailored for the Internet of Vehicles (IoV). This protocol empowers the RSU to evaluate the communication status between itself and the vehicle, thereby enabling rapid cross-domain authentication for the vehicle. Additionally, we conducted a formal security analysis of the proposed protocol using the ROR model and Scyther. Finally, we performed a comparative analysis of the proposed protocol against existing protocols in terms of security and performance, with results demonstrating that our protocol excels in both security and performance.

Acknowledgement: Not applicable.

Funding Statement: This work was supported by the Startup Foundation for Introducing Talent of Nanjing University of Information Science and Technology and Natural Science Foundation of Shandong Province, China (Grant no. ZR202111230202).

Author Contributions: The authors confirm contribution to the paper as follows: Conceptualization, Tsu-Yang Wu and Haozhi Wu; methodology, Tsu-Yang Wu and Haozhi Wu; validation, Maoxin Tang; formal analysis, Haozhi Wu and Maoxin Tang; investigation, Saru Kumari and Chien-Ming Chen; data curation, Saru Kumari and Chien-Ming Chen; writing—original draft preparation, Tsu-Yang Wu, Haozhi Wu, Maoxin Tang, Saru Kumari and Chien-Ming Chen; writing—review and editing, Tsu-Yang Wu, Maoxin Tang, Saru Kumari and Chien-Ming Chen. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: The data are contained within the article.

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest to report regarding the present study.

References

1. Dey KC, Rayamajhi A, Chowdhury M, Bhavsar P, Martin J. Vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communication in a heterogeneous wireless network-Performance evaluation. Transport Res Part C: Emerg Technol. 2016;68(6):168–84. doi:10.1016/j.trc.2016.03.008.

2. Yu S, Lee J, Park K, Das AK, Park Y. IoV-SMAP: secure and efficient message authentication protocol for IoV in smart city environment. IEEE Access. 2020;8:167875–86. doi:10.1109/access.2020.3022778.
3. Mejri MN, Ben-Othman J, Hamdi M. Survey on VANET security challenges and possible cryptographic solutions. Vehic Communicat. 2014;1(2):53–66. doi:10.1016/j.vehcom.2014.05.001.
4. Li X, Liu T, Obaidat MS, Wu F, Vijayakumar P, Kumar N. A lightweight privacy-preserving authentication protocol for VANETs. IEEE Systems J. 2020;14(3):3547–57. doi:10.1109/jsyst.2020.2991168.
5. Gupta D, Rathi R. RDVFF-reliable data dissemination in vehicular ad hoc networks based on validation of far to farthest zone. J Internet Technol. 2024;25(1):87–104.
6. Sutrala AK, Bagga P, Das AK, Kumar N, Rodrigues JJ, Lorenz P. On the design of conditional privacy preserving batch verification-based authentication scheme for internet of vehicles deployment. IEEE Transacti Vehic Technol. 2020;69(5):5535–48. doi:10.1109/tvt.2020.2981934.
7. Lv S, Liu Y. PLVA: privacy-preserving and lightweight V2I authentication protocol. IEEE Transact Intell Transport Syst. 2021;23(7):6633–9. doi:10.1109/tits.2021.3059638.
8. Wang H, Zhang F, Shen Z, Liu P, Liu K. Blockchain-based IVPPA scheme for pseudonym privacy protection in internet of vehicles. J Network Intell. 2024;9(2):1260–77.
9. Liu Y, Wang Y, Chang G. Efficient privacy-preserving dual authentication and key agreement scheme for secure V2V communications in an IoV paradigm. IEEE Transact Intell Transport Syst. 2017;18(10):2740–9. doi:10.1109/tits.2017.2657649.
10. Lee J, Kim G, Das AK, Park Y. Secure and efficient honey list-based authentication protocol for vehicular ad hoc networks. IEEE Transact Netw Sci Eng. 2021;8(3):2412–25. doi:10.1109/tnse.2021.3093435.
11. Eftekhari SA, Nikooghadam M, Rafighi M. Security-enhanced three-party pairwise secret key agreement protocol for fog-based vehicular ad-hoc communications. Veh Commun. 2021;28(1):100306. doi:10.1016/j.vehcom.2020.100306.
12. Xu C, Huang X, Ma M, Bao H. A privacy-preserving and cross-domain group authentication scheme for vehicular in LTE-A networks. J Communicat. 2017;12(11):604–10. doi:10.12720/jcm.12.11.604-610.
13. Yang Y, Wei L, Wu J, Long C, Li B. A blockchain-based multidomain authentication scheme for conditional privacy preserving in vehicular ad-hoc network. IEEE Int Things J. 2021;9(11):8078–90. doi:10.1109/jiot.2021.3107443.
14. Yu F, Ma M, Li X. A blockchain-assisted seamless handover authentication for V2I communication in 5G wireless networks. In: ICC 2021-IEEE International Conference on Communications. Montreal, QC, Canada: IEEE; 2021. p. 1–6.
15. Jiang Q, Zhang X, Zhang N, Tian Y, Ma X, Ma J. Three-factor authentication protocol using physical unclonable function for IoV. Comput Communicat. 2021;173(5):45–55. doi:10.1016/j.comcom.2021.03.022.
16. Babu PR, Reddy AG, Palaniswamy B, Das AK. EV-PUF: lightweight security protocol for dynamic charging system of electric vehicles using physical unclonable functions. IEEE Transact Netw Sci Eng. 2022;9(5):3791–807. doi:10.1109/tnse.2022.3186949.
17. Wu TY, Wu H, Kumari S, Chen CM. An enhanced three-factor based authentication and key agreement protocol using PUF in IoMT. Peer Peer Netw Appl. 2025;18(2):83. doi:10.1007/s12083-024-01839-z.
18. Yan X, Ma M, Su R. A certificateless efficient and secure group handover authentication protocol in 5G enabled vehicular networks. In: ICC 2022-IEEE International Conference on Communications. Seoul, Republic of Korea: IEEE; 2022. p. 1678–84. doi:10.1109/ICC42927.2021.9500334.
19. Chen CM, Li Z, Kumari S, Srivastava G, Lakshmanna K, Gadekallu TR. A provably secure key transfer protocol for the fog-enabled Social Internet of Vehicles based on a confidential computing environment. Veh Commun. 2023;39(1):100567. doi:10.1016/j.vehcom.2022.100567.
20. Costan V, Devadas S. Intel SGX Explained, Cryptology ePrint Archive, Report 2016/086. 2016.
21. Liu X, Guo Z, Ma J, Song Y. A secure authentication scheme for wireless sensor networks based on DAC and Intel SGX. IEEE Int Things J. 2021;9(5):3533–47. doi:10.1109/jiot.2021.3097996.
22. Will NC, Maziero CA. Intel software guard extensions applications: a survey. ACM Comput Surv. 2023;55(14s):322. doi:10.1145/3593021.

23. Dolev D, Yao A. On the security of public key protocols. IEEE Transact Inform Theory. 1983;29(2):198–208. doi:10.1109/tit.1983.1056650.
24. Canetti R, Krawczyk H. Analysis of key-exchange protocols and their use for building secure channels. In: International Conference on the Theory and Applications of Cryptographic Techniques. Berlin/Heidelberg, Germany: Springer; 2001. p. 453–74. doi:10.1007/3-540-44987-6_28.
25. Gao Y, Zhou T, Zheng W, Yang H, Zhang T. High-availability authentication and key agreement for internet of things-based devices in Industry 5.0. IEEE Transact Indust Inform. 2024;20(12):13571–9. doi:10.1109/tii.2024.3414443.
26. Alrashdi I, Tanveer M, Aldossari SA, Alshammeri M, Armghan A. BSCP-SG: blockchain-enabled secure communication protocols for IoT-driven smart grid systems. Int Things. 2025;32(2):101626. doi:10.1016/j.iot.2025.101626.