mathematics

MDPI

Article

ASCON-Based Pairing-Free Certificateless Authentication and Key Agreement Protocol for IoV

Minsu Kim, Taehun Kim, Deokkyu Kwon * and Youngho Park *

School of Electronic and Electrical Engineering, Kyungpook National University, Daegu 41566, Republic of Korea; msk0901@knu.ac.kr (M.K.); kimth028@knu.ac.kr (T.K.)
* Correspondence: kdk145@knu.ac.kr (D.K.); parkyh@knu.ac.kr (Y.P.); Tel.: +82-53-950-7842 (Y.P.)

Abstract

The increased deployment of vehicles with advanced on-board units has facilitated intelligent transportation services using communication with roadside units. However, such communication is conducted over open channels; hence, transmitted messages are vulnerable to security threats, necessitating secure and efficient authentication protocols. Recently, an authentication protocol for the Internet of Vehicles. This paper presents a detailed cryptanalysis of the protocol and demonstrates that the protocol is vulnerable to session key disclosure, man-in-the-middle, and vehicle-capture attacks, and fails to ensure vehicle untraceability. To overcome these security weaknesses, we propose an ASCON-based pairing-free certificateless authentication and key agreement protocol for the IoV. We employ ASCON to offer lightweight confidentiality and integrity protection and adopt certificateless authentication based on elliptic curve cryptography. Moreover, it supports efficient batch authentication of multiple vehicles to reduce authentication delay under high-density traffic conditions. We validate the security of the proposed protocol through informal and formal security analyses and evaluate its practicality through computation and communication cost comparisons, as well as network simulator 3 and simulation of urban mobility experiments. The results demonstrate that the proposed protocol provides secure and low-latency authentication suitable for dynamic IoV environments.

Keywords: ASCON; authentication; certificateless authentication; Internet of Vehicles; pairing-free; security analysis

MSC: 68M12

Check for updates

Academic Editor: Antanas Cenys and Abdullahi Arabo

Received: 28 September 2026
Revised: 24 October 2026
Accepted: 26 October 2026
Published: 3 November 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

1. Introduction

The rapid advancement of on-board units (OBUs) and wireless communication technology has transformed modern vehicles into mobile computing platforms, accelerating the development of the Internet of Vehicles (IoV). Vehicles can communicate with roadside units (RSUs) to exchange real-time traffic and safety information and access a wide range of intelligent transportation services. IoV communications include vehicle-to-roadside unit (V2R) communication and vehicle-to-vehicle (V2V) communication [1]. These services include collision avoidance through timely safety warnings, traffic-flow management based on the current road and congestion conditions, and continuous network connectivity for navigation and data exchange. These capabilities have established the IoV as a foundational technology for intelligent transportation systems [2].

Despite the benefits of these IoV services, communication in the IoV remains exposed to security threats because messages are exchanged over open wireless channels. An

adversary can eavesdrop on transmitted messages to obtain sensitive information, such as vehicle identities, location trajectories, and driving patterns. The adversary can also forge, modify, or replay messages to impersonate vehicles, compromise session keys, or launch man-in-the-middle (MitM) attacks. Such attacks can not only violate user privacy but also disrupt safety-critical vehicular services. For example, falsified traffic or warning messages may mislead vehicles and RSUs, resulting in incorrect driving decisions or even severe accidents. Therefore, a secure authentication and key agreement (AKA) protocol is essential for establishing secure communication in IoV [3].

In addition to satisfying security requirements, pairing-free certificateless authentication protocols have been proposed to support efficient authentication while reducing dependence on a trusted authority (TA) [4,5]. Excessive dependence on the TA can create a single point of failure, as its failure or compromise can disrupt system-wide authentication. For example, if the TA becomes unavailable due to a system failure or attack, vehicles may be unable to complete authentication even when RSUs are available. To address this issue, certificateless public key cryptography (CL-PKC) allows the TA to issue only a partial private key, while each vehicle generates its complete private key [6,7]. Thus, the TA cannot derive the complete private key, mitigating the key escrow problem. Pairing-free schemes also replace costly bilinear pairing operations with elliptic curve cryptography (ECC) operations. Consequently, RSUs can perform batch verification without involving the TA in every authentication session, thereby reducing authentication latency and communication overhead. This is particularly advantageous in highly dynamic IoV environments, where numerous authentication requests must be processed with low latency [8,9].

Recently, Li et al. proposed an authentication protocol for IoV environments [10]. Their protocol was designed to provide identity privacy, revocability, and efficient mutual authentication between vehicles and RSUs. However, it fails to satisfy several important security properties. Specifically, their protocol is vulnerable to session key disclosure, MitM, and vehicle capture attacks and does not provide untraceability. These weaknesses can allow an adversary to recover confidential session data, impersonate vehicles, manipulate vehicular messages, and track vehicles across multiple sessions, thereby compromising both communication security and user privacy.

To address the above security weaknesses, we propose an ASCON-based pairing-free certificateless AKA protocol for IoV environments. The proposed protocol employs ASCON, a lightweight authenticated encryption algorithm, to provide confidentiality and integrity with low computational overhead, thereby supporting real-time vehicular communication. It also integrates ECC-based pairing-free certificateless authentication, reducing reliance on the TA and mitigating the key escrow problem. Moreover, by leveraging ECC operations, the proposed protocol supports batch verification, allowing an RSU to process multiple simultaneous authentication requests in a single verification procedure and thereby reduce latency in IoV environments.

1.1. Research Contributions

The main contributions of this paper are summarized as follows.

- We review Li et al.'s AKA protocol for the IoV [10] and conduct a detailed cryptanalysis. We demonstrate that their protocol is vulnerable to session key disclosure, MitM, and vehicle capture attacks, and fails to ensure vehicle untraceability.
- We propose an ASCON-based pairing-free certificateless AKA protocol for the IoV. The proposed protocol employs ASCON to provide lightweight confidentiality and integrity protection, and integrates ECC-based pairing-free certificateless authentication to reduce dependence on the TA, mitigate the key escrow problem, and support efficient individual and batch authentication for the dynamic vehicle environment.

- We analyze the security of the proposed protocol through both informal and formal methods. The informal security analysis is used to examine whether the proposed protocol satisfies security requirements and resists known attacks in IoV environments. For formal verification, we employ the random oracle model (ROM) [11], Burrows–Abadi–Needham (BAN) logic [12], and the Scyther tool [13,14].
- We evaluate the performance of the proposed protocol through computation and communication cost comparisons with related protocols. In addition, we conduct simulations using network simulator 3 (NS-3) and simulation of urban mobility (SUMO) [15,16] to demonstrate its practicality under realistic vehicular network environments.

1.2. Organization

The remainder of this paper is organized as follows. Section 2 presents related work, and Section 3 introduces the preliminaries. Section 4 reviews Li et al.'s protocol, and Section 5 analyzes its vulnerabilities. Section 6 presents the proposed ASCON-based pairing-free certificateless AKA protocol. Section 7 provides informal and formal analyses, including a ROM-based game proof, BAN logic, and Scyther tool. Section 8 evaluates the performance of the proposed protocol through comparative analysis and NS-3/SUMO-based simulations. Finally, Section 9 concludes this paper.

2. Related Work

In recent years, CL-PKC and aggregate signature techniques have been widely studied for efficient authentication in wireless networks. In 2003, Al-Riyami and Paterson [17] introduced CL-PKC to address the certificate management overhead of traditional public key infrastructure and the key escrow problem of identity-based cryptography. Boneh et al. [18] introduced aggregate signatures, which combine multiple signatures into a compact form for collective verification. Subsequently, Castro and Dahab [19] proposed the first certificateless aggregate signature (CLAS) construction in 2007.

However, many early CLAS protocols rely on bilinear pairing operations, resulting in high computational overhead for resource-constrained OBUs and RSUs. To address this limitation, Cui et al. [20] proposed the first ECC-based pairing-free CLAS protocol for VANETs by eliminating bilinear pairings and map-to-point hash functions. Kamil and Ogundoyin [21], however, showed that Cui et al.'s protocol is vulnerable to forgery attacks and proposed an improved construction. Subsequently, Zhao et al. [22], Thumbur et al. [23], and Yang et al. [24] analyzed pairing-free CLAS protocols under Type 1 and Type 2 adversary models. A Type 1 adversary can replace public keys without knowing the master secret key, whereas a Type 2 adversary knows the master secret key but cannot replace public keys. Their analyses showed that several protocols remain vulnerable to public-key replacement, malicious KGC, coalition, and aggregate-signature forgery attacks. More recently, Ye et al. [25] introduced a Type 3 adversary model to capture aggregate-level attacks in which invalid signatures may pass batch verification through collusion or manipulation. Therefore, an efficient IoV authentication protocol should avoid expensive pairing operations while resisting Type 1, Type 2, and Type 3 adversaries.

Numerous AKA protocols have also been proposed for IoV and VANET environments. In 2023, Badshah et al. [26] proposed AAKE-BIVT, a blockchain-enabled anonymous AKA protocol that establishes session keys among vehicles, cluster heads, RSUs, and cloud servers while supporting decentralized storage and transaction verification. Wang et al. [27] proposed a certificateless anonymous and revocable protocol for V2V communications, separating identity authentication from traffic-message verification to reduce revocation-list checking. Xie et al. [28] introduced a secure and anonymous V2I and V2V protocol

based on ECC, physical unclonable functions, and biometric keys. Their protocol resists RSU capture and OBU intrusion attacks and supports batch verification of multiple V2I requests. In 2024, Wei et al. [29] proposed a threshold-based decentralized AKA protocol using consortium blockchain, which mitigates key exposure through threshold voting and reduces on-chain storage using Cuckoo filters. In 2025, Kong et al. [30] presented an ECC-based handover protocol employing a non-full key escrow mechanism and time-tree technology to reduce cross-domain handover overhead. Yu et al. [31] proposed an ASCON-based AKA protocol for fog-enabled VANETs. Their protocol updates pre-shared keys between sessions, while fog servers support vehicle grouping and intra-group V2V key exchange. Yang et al. [32] proposed pairing-free certificateless authentication scheme with batch verification for VANETs.

Recently, Li et al. [10] proposed an anonymous and revocable authentication protocol with session key agreement for IoV. Although their protocol is computationally efficient because it primarily relies on hash operations, our analysis reveals that it remains vulnerable to session-key disclosure, MitM, vehicle capture, and vehicle tracing attacks. Moreover, its security model does not sufficiently address threats associated with certificateless aggregate authentication. Therefore, we propose an ASCON-based pairing-free certificateless AKA protocol for IoV that overcomes these limitations while supporting secure and efficient individual and batch authentication against Type 1, Type 2, and Type 3 adversaries.

3. Preliminaries

3.1. Adversary Model

We consider a comprehensive adversary model that covers the AKA protocol and the batch verification mechanism based on CL-PKC. For the AKA protocol, we adopt the commonly used Dolev–Yao (DY) model [33] and the Canetti–Krawczyk (CK) model [34]. We also consider Type 1, Type 2, and Type 3 adversaries for certificateless authentication. The adversary $\mathcal{A}$ is assumed to have the following capabilities.

- $\mathcal{A}$ can eavesdrop on, intercept, modify, delete, replay, and inject messages exchanged between legitimate entities.
- $\mathcal{A}$ can extract secret parameters stored in a vehicle's OBU to perform security attacks such as impersonation.
- $\mathcal{A}$ may obtain ephemeral secrets or private parameters of certain entities. Using such exposed information, $\mathcal{A}$ may attempt to compute the session key or compromise the forward secrecy of the protocol.
- $\mathcal{A}$ may act as a Type 1, Type 2 [17], or Type 3 [25] adversary. Accordingly, $\mathcal{A}$ may replace public keys, exploit the TA's master secret key, or forge an aggregate authentication result using multiple signatures and public verification values.

3.2. ASCON

ASCON is a lightweight authenticated encryption with associated data algorithm designed for resource-constrained devices [35,36]. It is based on a permutation-based structure and provides confidentiality and integrity within a single cryptographic operation. ASCON processes a secret key, nonce, associated data, and plaintext through initialization, data processing, and finalization phases. Let K, N, AD, and PT denote a secret key, nonce, associated data, and plaintext, respectively. Its encryption and decryption operations are defined as:

$$(CT, Tag) = E_K(N, AD, PT), \tag{1}$$

$$(PT, \perp) = D_K(N, AD, CT, Tag). \tag{2}$$

The associated data AD is not encrypted but is authenticated, whereas the plaintext PT is both encrypted and authenticated. If the received authentication tag is invalid, the decryption algorithm outputs $\perp$.

In the proposed protocol, V_i and RSU_j derive the same ASCON request key as $K_{ij} = H_5(e_i \cdot Y_j \parallel T_1) = H_5(y_j \cdot E_i \parallel T_1)$ and response key as $W_{ij} = H_5(e_i \cdot F_j \parallel T_2) = H_5(f_j \cdot E_i \parallel T_2)$. The vehicle protects its authentication request by computing $(CT_1, Tag_1) = E_{K_{ij}}(r_1, AID_i \parallel T_1, \sigma_i \parallel n_i \parallel m)$, while the RSU protects its response as $(CT_2, Tag_2) = E_{W_{ij}}(r_1, AID_i \parallel T_2, n_j)$. Each entity verifies the corresponding authentication tag before accepting the decrypted message, thereby ensuring both confidentiality and integrity through lightweight ASCON operations. This enables an efficient authentication process suitable for dynamic IoV environments, where low-latency communication is critical.

3.3. CL-PKC

CL-PKC was introduced to eliminate the certificate management overhead of traditional public key infrastructure while mitigating the key escrow problem of identity-based cryptography [17]. Unlike identity-based cryptography, in which the trusted authority generates the complete private key, CL-PKC divides the private key into a partial private key generated by the TA and a secret value selected independently by the user.

In the proposed protocol, the TA generates a partial private key d_i and the corresponding public value R_i for a registered vehicle V_i. The vehicle independently selects its secret value x_i and computes $X_i = x_i \cdot P$. Thus, the complete private key of V_i consists of x_i and d_i, while its public values consist of X_i and R_i. Since x_i is known only to the vehicle, the TA cannot derive the complete private key. This structure mitigates the key escrow problem and enables an RSU to authenticate vehicles without requiring the TA to participate in every authentication session.

3.4. System Model

The system model of the proposed protocol consists of three main entities, vehicles, RSUs, and a trusted authority (TA), as shown in Figure 1. Communications between vehicles and RSUs are conducted over public wireless channels, whereas registration phases are performed over secure channels.

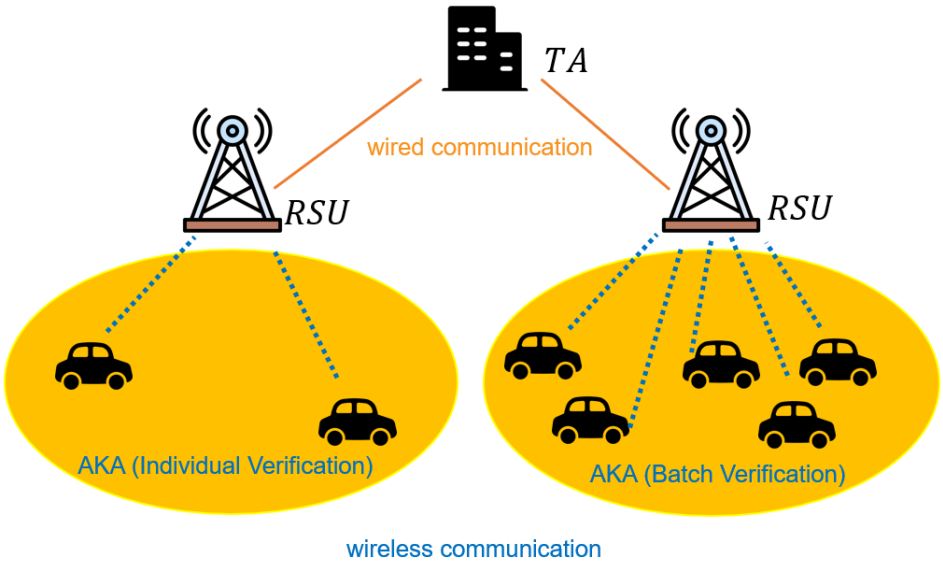


Figure 1. System model of the proposed IoV authentication protocol.

(1) *Vehicle*: A vehicle is equipped with an OBU and is regarded as an untrusted entity before authentication phase. To access network services, the vehicle first performs the proposed AKA protocol with an RSU located within its communication range. After successful mutual authentication and session key establishment, the vehicle can securely communicate with the RSU and use ITS services such as traffic information delivery, safety message exchange, and driving assistance.

(2) *RSU*: RSUs are deployed along the roadside to collect traffic-related information from nearby vehicles and provide network services to them. The RSU monitors the number of authentication requests from vehicles within its coverage area and flexibly selects either individual authentication or batch authentication.

(3) *TA*: The TA is a trusted entity responsible for system initialization, vehicle registration, RSU registration, and KGC operations in the CL-PKC setting. It generates system parameters, issues partial private keys, and manages registration information for vehicles and RSUs. TA is assumed to be fully trusted and equipped with sufficient computational and storage resources.

4. Review of Li et al.'s Protocol

This section reviews Li et al.'s authentication protocol [10] for IoV environments.

4.1. Authentication and Enrollment Phase

Before providing services to vehicles within its coverage area, RSU_j performs the authentication and enrollment phase with the TA. The detailed steps are as follows.

Step 1: RSU_j inputs its identity ID_j, generates a timestamp t_1, and computes $RN_j = h(ID_j) \oplus TID_j$, $B_j = h(TID_j \parallel RN_j)$, $C_j = B_j \oplus TID_j$, and $D_j = h(TID_j \parallel A_j \parallel C_j \parallel t_1)$. Then, RSU_j sends $Msg1 : \langle TID_j, D_j, t_1 \rangle$ to the TA.

Step 2: Upon receiving $Msg1$, the TA checks the validity of t_1. If t_1 is invalid, the process is terminated. Otherwise, the TA checks whether TID_j exists in its memory. If TID_j is valid, the TA retrieves (ID_j, TID_j, RN_j) and computes $S_j = h(K \parallel RN_j)$, $A_j = S_j \oplus TID_j$, $B_j = h(TID_j \parallel RN_j)$, $C_j = B_j \oplus TID_j$, and $D_j^* = h(TID_j \parallel A_j \parallel C_j \parallel t_1)$. The TA then checks whether $D_j^* = D_j$. If the verification fails, the process is terminated.

Step 3: If the verification succeeds, the TA generates a timestamp t_2 and computes $E_j = S_j \oplus RN_j$ and $F_j = h(TID_j \parallel S_j \parallel E_j \parallel t_2)$. Then, the TA sends $Msg2 : \langle E_j, F_j, t_2 \rangle$ to RSU_j.

Step 4: Upon receiving $Msg2$, RSU_j checks the validity of t_2. If t_2 is invalid, the process is terminated. Otherwise, RSU_j computes $S_j = TID_j \oplus A_j$ and $F_j^* = h(TID_j \parallel S_j \parallel E_j \parallel t_2)$. Then, RSU_j checks whether $F_j^* = F_j$. If the verification succeeds, RSU_j is authenticated and broadcasts a "ready" message to all vehicles within its coverage area.

4.2. AKA Phase

In this phase, $Vehicle_i$ and RSU_j perform mutual authentication and establish a session key with the assistance of the TA. The detailed steps are as follows.

Step 1: After receiving the "ready" message, $Vehicle_i$ inputs its identity ID_i, generates a timestamp t_3, and computes $RN_i = h(ID_i) \oplus TID_i$, $B_i = h(TID_i \parallel RN_i)$, and $C_i = h(TID_i \parallel A_i \parallel B_i \parallel t_3)$. Then, $Vehicle_i$ sends $Msg3 : \langle TID_i, C_i, t_3 \rangle$ to RSU_j.

Step 2: Upon receiving $Msg3$, RSU_j checks the validity of t_3. If t_3 is invalid, the process is terminated. Otherwise, RSU_j retrieves TID_j from its memory, generates a timestamp t_4, and forwards $Msg4 : \langle TID_i, TID_j, C_i, t_3, t_4 \rangle$ to the TA.

Step 3: Upon receiving $Msg4$, the TA checks the validities of t_4 and TID_i. If either value is invalid, the process is terminated. Otherwise, the TA retrieves $\langle ID_i, TID_i, RN_i \rangle$ using TID_i and computes $S_i = h(K \parallel RN_i)$, $A_i = S_i \oplus TID_i$, $B_i = h(TID_i \parallel RN_i)$, and

$C_i^* = h(TID_i \| A_i \| B_i \| t_3)$. Then, the TA checks whether $C_i^* = C_i$. If the verification fails, the process is terminated.

Step 4: If the verification succeeds, the TA generates new random numbers RN_i^+ and RN_j^+ for $Vehicle_i$ and RSU_j, respectively. Then, the TA computes $D_i = TID_i \oplus RN_i^+$ and $G_j = TID_j \oplus RN_j^+$, generates a timestamp t_5, and sends $Msg5 : \langle D_i, G_j, t_5 \rangle$ to RSU_j.

Step 5: Upon receiving $Msg5$, RSU_j checks the validity of t_5. If t_5 is invalid, the process is terminated. Otherwise, RSU_j retrieves A_j from its memory and computes $S_j = A_j \oplus TID_j$, $RN_j^+ = G_j \oplus TID_j$, $RN_i^+ = D_i \oplus TID_i$, and $SK_{ij} = h(RN_i^+ \| RN_j^+)$. Then, RSU_j generates a timestamp t_6, computes $H_{ij} = h(D_i \| G_j \| SK_{ij} \| t_6)$, and sends $Msg6 : \langle TID_j, D_i, G_j, A_j, H_{ij}, t_6 \rangle$ to $Vehicle_i$.

Step 6: Upon receiving $Msg6$, $Vehicle_i$ checks the validity of t_6. If t_6 is invalid, the process is terminated. Otherwise, $Vehicle_i$ retrieves A_i and TID_i from its memory and computes $RN_i^+ = D_i \oplus TID_i$, $RN_j^+ = G_j \oplus TID_j$, $SK_{ij} = h(RN_i^+ \| RN_j^+)$, and $H_{ij}^* = h(D_i \| G_j \| SK_{ij} \| t_6)$. Then, $Vehicle_i$ checks whether $H_{ij}^* = H_{ij}$. If the verification fails, the process is terminated.

Step 7: If the verification succeeds, $Vehicle_i$ computes $S_i = TID_i \oplus A_i$, $S_j = TID_j \oplus A_j$, $A_i^+ = S_i \oplus RN_i^+$, and $TID_i^+ = h(ID_i) \oplus RN_i^+$. Then, $Vehicle_i$ replaces (A_i, TID_i) with (A_i^+, TID_i^+) in its memory. Next, $Vehicle_i$ computes $L_{ij} = h(S_j \| RN_i^+ \| RN_j^+)$ and sends $Msg7 : \langle L_{ij}, t_7 \rangle$ to RSU_j.

Step 8: Upon receiving $Msg7$, RSU_j checks the validity of t_7. If t_7 is invalid, the process is terminated. Otherwise, RSU_j computes $S_j = A_j \oplus TID_j$, $RN_i^+ = D_i \oplus TID_i$, $RN_j^+ = G_j \oplus TID_j$, and $L_{ij}^* = h(S_j \| RN_i^+ \| RN_j^+)$. Then, RSU_j checks whether $L_{ij}^* = L_{ij}$. If the verification fails, the process is terminated.

Step 9: If the verification succeeds, RSU_j computes $TID_j^+ = h(ID_j) \oplus RN_j^+$ and $A_j^+ = S_j \oplus TID_j^+$. Then, RSU_j replaces (TID_j, A_j) with (TID_j^+, A_j^+) in its memory, generates a timestamp t_8, and sends $Msg8 : \langle S_j, G_j, t_8 \rangle$ to the TA.

Step 10: Upon receiving $Msg8$, the TA checks the validity of t_8. If t_8 is invalid, the process is terminated. Otherwise, the TA computes $RN_i^+ = D_i \oplus TID_i$, $TID_i^+ = h(ID_i) \oplus RN_i^+$, $RN_j^+ = G_j \oplus TID_j$, and $TID_j^+ = h(ID_j) \oplus RN_j^+$. Finally, the TA replaces (ID_i, TID_i, RN_i) with (ID_i, TID_i^+, RN_i^+) and replaces (ID_j, TID_j, RN_j) with (ID_j, TID_j^+, RN_j^+) in its memory. After this phase, $Vehicle_i$ and RSU_j share the session key SK_{ij}.

5. Cryptanalysis of Li et al.'s Protocol

5.1. Session Key Disclosure Attack

During the AKA phase, $\mathcal{A}$ can eavesdrop on $Msg3 = \langle TID_i, C_i, t_3 \rangle$ and obtain TID_i. In addition, $\mathcal{A}$ can eavesdrop on $Msg6 = \langle TID_j, D_i, G_j, A_j, H_{ij}, t_6 \rangle$ and obtain TID_j, D_i, and G_j. Since D_i and G_j are computed as $D_i = TID_i \oplus RN_i^+$ and $G_j = TID_j \oplus RN_j^+$, respectively, $\mathcal{A}$ can compute $RN_i^+ = TID_i \oplus D_i$ and $RN_j^+ = TID_j \oplus G_j$. Using the recovered values, $\mathcal{A}$ can derive the session key as $SK_{ij} = h(RN_i^+ \| RN_j^+)$. Thus, a passive adversary can compute the same session key as the legitimate vehicle and RSU only by eavesdropping on public messages. Therefore, Li et al.'s protocol fails to guarantee session key secrecy and is vulnerable to session key disclosure attacks.

5.2. MitM Attack

Assume that $\mathcal{A}$ is located between $Vehicle_i$ and RSU_j and controls the public channel. $\mathcal{A}$ intercepts $Msg6$ and obtains TID_j, D_i, G_j, and A_j. Since $\mathcal{A}$ already obtained TID_i from $Msg3$, it can compute $RN_i^+ = TID_i \oplus D_i$ and $RN_j^+ = TID_j \oplus G_j$. Thus, $\mathcal{A}$ derives the legitimate session key $SK_{\mathcal{A}j} = h(RN_i^+ \| RN_j^+) = SK_{ij}$ Next, $\mathcal{A}$ generates new random numbers $RN_i^{\mathcal{A}}$ and $RN_j^{\mathcal{A}}$ and computes $D_i^{\mathcal{A}} = TID_i \oplus RN_i^{\mathcal{A}}$ and $G_j^{\mathcal{A}} = TID_j \oplus RN_j^{\mathcal{A}}$. Then, $\mathcal{A}$ derives a forged session key $SK_{i\mathcal{A}} = h(RN_i^{\mathcal{A}} \| RN_j^{\mathcal{A}})$ and computes

$H_{ij}^{\mathcal{A}} = h(D_i^{\mathcal{A}} \| G_j^{\mathcal{A}} \| SK_{i\mathcal{A}} \| t_6^{\mathcal{A}})$ using a fresh timestamp $t_6^{\mathcal{A}}$. Finally, $\mathcal{A}$ sends the forged message $Msg6^{\mathcal{A}} = \langle TID_j, D_i^{\mathcal{A}}, G_j^{\mathcal{A}}, A_j, H_{ij}^{\mathcal{A}}, t_6^{\mathcal{A}} \rangle$ to $Vehicle_i$. Upon receiving $Msg6^{\mathcal{A}}$, $Vehicle_i$ computes $RN_i^{\mathcal{A}}$, $RN_j^{\mathcal{A}}$, and $SK_{i\mathcal{A}}$, and verifies $H_{ij}^{\mathcal{A}}$. Since the forged value is correctly generated by $\mathcal{A}$, the verification succeeds. As a result, $Vehicle_i$ believes that it has established a session key with RSU_j, while the actual key is shared with $\mathcal{A}$. Furthermore, since A_j and TID_j are included in messages, $\mathcal{A}$ can compute $S_j = A_j \oplus TID_j$. Therefore, $\mathcal{A}$ can also generate a valid response toward RSU_j after receiving $Msg7$ from $Vehicle_i$. Consequently, $\mathcal{A}$ can establish two different session keys: one with $Vehicle_i$ and the other with RSU_j. Therefore, Li et al.'s protocol is vulnerable to MitM attacks.

5.3. Physical-Capture Attack

Assume that $\mathcal{A}$ captures the OBU of V_i and obtains the stored parameters $\{TID_i^+, A_i^+\}$. In addition, $\mathcal{A}$ eavesdrops on $Msg3 = \{TID_i, C_i, t_3\}$ and $Msg6 = \{TID_j, D_i, G_j, A_j, H_{ij}, t_6\}$ to obtain TID_i and D_i at the most recent session. Since D_i is computed as $D_i = TID_i \oplus RN_i^+$, $\mathcal{A}$ can compute the updated random number as $RN_i^+ = TID_i \oplus D_i$. Using this value, $\mathcal{A}$ can generate a valid authentication request by impersonating $Vehicle_i$.

Specifically, $\mathcal{A}$ generates a fresh timestamp $t_3^{\mathcal{A}}$ and computes $B_i^{\mathcal{A}} = h(TID_i^+ \| RN_i^+)$ and $C_i^{\mathcal{A}} = h(TID_i^+ \| A_i^+ \| B_i^{\mathcal{A}} \| t_3^{\mathcal{A}})$. Then, $\mathcal{A}$ constructs a forged message $Msg3^{\mathcal{A}} = \{TID_i^+, C_i^{\mathcal{A}}, t_3^{\mathcal{A}}\}$ and sends it to RSU_j.

Since the forged value $C_i^{\mathcal{A}}$ has the same form as a legitimate authentication value, RSU_j forwards the request to the TA. The TA then verifies the received message using the stored information corresponding to TID_i^+. Therefore, $\mathcal{A}$ can initiate an authentication session as a legitimate vehicle. This shows that Li et al.'s protocol is vulnerable to physical-capture attacks and cannot prevent vehicle impersonation after OBU compromise.

5.4. Untraceability

During an authentication session, $\mathcal{A}$ can obtain TID_i from $Msg3 = \langle TID_i, C_i, t_3 \rangle$ and D_i from $Msg6 = \langle TID_j, D_i, G_j, A_j, H_{ij}, t_6 \rangle$. Since $D_i = TID_i \oplus RN_i^+$, $\mathcal{A}$ can compute $RN_i^+ = TID_i \oplus D_i$. After successful authentication, $Vehicle_i$ updates its temporary identity as $TID_i^+ = h(ID_i) \oplus RN_i^+$. In the next authentication session, the updated identity TID_i^+ is transmitted over the public channel as part of $Msg3$.

Using the observed TID_i^+ and the previously recovered RN_i^+, $\mathcal{A}$ can compute $h(ID_i) = TID_i^+ \oplus RN_i^+ = TID_i^+ \oplus TID_i \oplus D_i$. Since $h(ID_i)$ is fixed for the same vehicle, $\mathcal{A}$ can use this value as a linking factor across different sessions. In the next session, $\mathcal{A}$ eavesdrops on D_i^+ from $Msg6$ and computes $RN_i^{++} = TID_i^+ \oplus D_i^+$ and $TID_i^{++} = h(ID_i) \oplus RN_i^{++}$. Similarly, by obtaining D_i^{++} in the following session, $\mathcal{A}$ can derive $TID_i^{+++} = h(ID_i) \oplus (TID_i^{++} \oplus D_i^{++})$. By continuously eavesdropping $Msg6$, $\mathcal{A}$ can derive all temporary identities. By repeating this process, $\mathcal{A}$ can determine whether multiple authentication sessions belong to the same vehicle. Therefore, Li et al.'s protocol fails to guarantee vehicle untraceability.

6. Proposed Protocol

We propose an ASCON-based pairing-free certificateless AKA protocol for IoV environments. The proposed protocol is designed to overcome the security weaknesses of Li et al.'s protocol while supporting efficient authentication for dynamically changing vehicle density. The notations used in the proposed protocol are summarized in Table 1. The proposed protocol consists of four phases: initialization, registration, individual AKA, and batch AKA. The RSU performs individual verification when the number of authentication requests remains below a predefined threshold. When a large number of requests exceeds this threshold within a short time interval, the RSU switches to batch verification.

Table 1. Notation used in the proposed protocol.

Notation	Description
V_i	ith vehicle equipped with an OBU
RSU_j	jth RSU
TA	Trusted authority
$\mathbb{G}, q, P$	ECC curve group, order, and generator
ID_i	Real identity of V_i
ID_j	Identity of RSU_j
PW_i	Password of V_i
AID_i	Anonymous identity of V_i
SID_i	Secret identity of V_i
α, β	Master secret values of the TA
T_{pub}, K_{pub}	Public keys of the TA
(x_i, d_i)	Private key of V_i
(X_i, R_i)	Public Key of V_i
u_i, U_i	Secret value and public point of V_i
r_i, R_i	Secret value and public point of TA for V_i
y_j	Secret key of RSU_j
Y_j	Public key of RSU_j
$\mathcal{D}_R$	RSU information set for vehicles
$\mathcal{D}_V$	Vehicle information set for RSU
e_i, E_i	Ephemeral secret and public point of V_i
f_j, F_j	Ephemeral secret and public point of RSU_j
σ_i	Certificateless signature generated by V_i
K_{ij}	ASCON request key
W_{ij}	ASCON response key
SK_{ij}	Session key shared between V_i and RSU_j
n_i, n_j	Random nonces of V_i and RSU_j
r_1	Nonce for ASCON operation
T_i, T_1, T_2	Registration and AKA timestamps
$T_{\text{now}}, \Delta T$	Current time and freshness interval
m	Traffic and service information of V_i
C_1	Masked identity and nonce $(AID_i \parallel r_1)$
E_K, D_K	ASCON encryption and decryption functions
$H_0(\cdot) - H_5(\cdot)$	Cryptographic hash functions for registration phase and signature generation
$H_6(\cdot) - H_7(\cdot)$	Cryptographic hash functions for key agreement
n, s_i	Batch size and random verification coefficient
$\parallel$	Concatenation operation
$\oplus$	Exclusive-OR operation

6.1. Initialization Phase

Before all operational phases, the TA performs the initialization phase to generate public system parameters and master secret values. The details of this phase are described as follows.

Step 1: The TA selects an elliptic curve group $\mathbb{G}$ with prime order q and generator P. Then, the TA selects two master secret values $\alpha, \beta \in \mathbb{Z}_q^*$ and computes the corresponding public keys as $T_{pub} = \alpha \cdot P$ and $K_{pub} = \beta \cdot P$.

Step 2: The TA selects cryptographic hash functions $H_0 : \mathbb{G} \times \mathbb{G} \times \{0,1\}^\omega \to \{0,1\}^\nu$, $H_1 : \{0,1\}^\nu \times \mathbb{G} \times \mathbb{G} \times \mathbb{G} \to \mathbb{Z}_q^*$, $H_2 : \{0,1\}^* \to \{0,1\}^*$, $H_3 : \{0,1\}^* \to \mathbb{Z}_q^*$, $H_4 : \{0,1\}^\nu \times \{0,1\}^* \times \{0,1\}^* \to \mathbb{Z}_q^*$, $H_5 : \{0,1\}^\nu \times \mathbb{Z}_q \times \mathbb{Z}_q \times \{0,1\}^* \to \{0,1\}^\ell$, $H_6 : \mathbb{G} \times \{0,1\}^\omega \to \{0,1\}^\kappa$, and $H_7 : \{0,1\}^\nu \times \{0,1\}^\kappa \times \mathbb{G} \times \{0,1\}^* \times \{0,1\}^* \times \{0,1\}^* \times \{0,1\}^\omega \times \{0,1\}^\omega \to \{0,1\}^\ell$. In addition, the TA initializes the ASCON authenticated encryption and decryption functions E_K and D_K. The TA selects a integer n_f, with $2 \leq n_f < 2^\ell$, as the fuzzy-verifier modulus.

Step 3: Finally, the TA publicizes the system parameters $\{\mathbb{G}, q, P, T_{pub}, K_{pub}, H_0 - H_7, E_K, D_K, n_f\}$ together with the specified encoding and output lengths, and securely keeps the master secret values α and β.

6.2. Registration Phase

6.2.1. Vehicle Registration Phase

Each vehicle V_i performs the registration phase with the TA through a secure channel. Figure 2 shows the proposed vehicle registration phase and details are as follows:

Step 1: Vehicle V_i selects its identity ID_i, password PW_i, and independent secret values $x_i, u_i \in \mathbb{Z}_q^*$. Then, V_i generates a random salt r_0 and a timestamp T_i, and computes $X_i = x_i \cdot P$ and $U_i = u_i \cdot P$. It computes $SID_i = ID_i \oplus H_0(x_i \cdot T_{pub} \| T_{pub} \| T_i)$ and sends $\{SID_i, X_i, U_i, T_i\}$ to the TA through the secure channel.

Step 2: Upon receiving the registration request, the TA checks the freshness of T_i and recovers $ID_i = SID_i \oplus H_0(\alpha \cdot X_i \| T_{pub} \| T_i)$. Next, the TA computes the anonymous identity $AID_i = ID_i \oplus H_0(\beta \cdot U_i \| K_{pub} \| T_i)$. The TA selects $r_i \in \mathbb{Z}_q^*$ and computes $R_i = r_i \cdot P$. It computes $h_{1,i} = H_1(AID_i \| R_i \| X_i \| K_{pub})$ and generates the partial private key $d_i = (r_i + h_{1,i} \cdot \beta) \bmod q$. Then, the TA stores $\{X_i, U_i, R_i, T_i\}$ corresponding to AID_i, derives $\mathcal{D}_R = \sum_{j=1}^{n} (ID_j, Y_j)$ for the vehicle's region, and sends $\{AID_i, R_i, d_i, \mathcal{D}_R\}$ to V_i through the secure channel.

Step 3: After receiving the registration response, V_i computes $h_{1,i}$ and verifies $d_i \cdot P \stackrel{?}{=} R_i + h_{1,i} \cdot K_{pub}$ and $ID_i \stackrel{?}{=} AID_i \oplus H_0(u_i \cdot K_{pub} \| K_{pub} \| T_i)$. If verification fails, V_i terminates the registration phase. Otherwise, V_i computes $A_1 = (d_i + H_4(ID_i \| PW_i \| r_0)) \bmod q$, $B_1 = (x_i + H_4(ID_i \| PW_i \| d_i)) \bmod q$, and $V_f = H_5(ID_i \| d_i \| x_i \| PW_i) \bmod n_f$. Finally, V_i stores $\{AID_i, A_1, B_1, V_f, r_0, X_i, R_i, \mathcal{D}_R\}$ in its OBU.

***i*th Vehicle** V_i	**Trusted Authority** TA
Select ID_i, PW_i, x_i, u_i	
Generate r_0, T_i	
Compute	
$X_i = x_i \cdot P, U_i = u_i \cdot P$	
$SID_i = ID_i \oplus H_0(x_i \cdot T_{pub} \| T_{pub} \| T_i)$	
$\xrightarrow{\{SID_i, X_i, U_i, T_i\}}$	
	Checks the freshness of T_i
	Compute
	$ID_i = SID_i \oplus H_0(\alpha \cdot X_i \| T_{pub} \| T_i)$
	$AID_i = ID_i \oplus H_0(\beta \cdot U_i \| K_{pub} \| T_i)$
	Select r_i
	Compute $R_i = r_i \cdot P$
	$h_{1,i} = H_1(AID_i \| R_i \| X_i \| K_{pub})$
	$d_i = (r_i + h_{1,i} \cdot \beta) \bmod q$
	Store $\{X_i, U_i, R_i, T_i\}$ against AID_i
	$\xleftarrow{\{AID_i, R_i, d_i, \mathcal{D}_R\}}$
Compute	
$h_{1,i} = H_1(AID_i \| R_i \| X_i \| K_{pub})$	
Verify	
$d_i \cdot P \stackrel{?}{=} R_i + h_{1,i} \cdot K_{pub}$	
$ID_i \stackrel{?}{=} AID_i \oplus H_0(u_i \cdot K_{pub} \| K_{pub} \| T_i)$	
Compute	
$A_1 = (d_i + H_4(ID_i \| PW_i \| r_0)) \bmod q$	
$B_1 = (x_i + H_4(ID_i \| PW_i \| d_i)) \bmod q$	
$V_f = H_5(ID_i \| d_i \| x_i \| PW_i) \bmod n_f$	
Store $\{AID_i, A_1, B_1, V_f, r_0, X_i, R_i, \mathcal{D}_R\}$	

Figure 2. Vehicle registration phase of the proposed protocol.

6.2.2. RSU Registration Phase

Each RSU also performs a registration phase with the TA through a secure channel. Figure 3 shows the proposed RSU registration phase and details are as follows:

Step 1: RSU_j selects its identity ID_j and secret value $y_j \in \mathbb{Z}_q^*$. Then, RSU_j computes its public key $Y_j = y_j \cdot P$ and sends the registration request $\{ID_j, Y_j\}$ to the TA through the secure channel.

Step 2: After verifying the registration request, the TA stores $\{ID_j, Y_j\}$ in its database. The TA also sends $\mathcal{D}_V = \sum_{i=1}^{n}\{AID_i, X_i, R_i\}$ to RSU_j through the secure channel.

Step 3: Upon receiving $\mathcal{D}_V$, RSU_j stores it in its local memory.

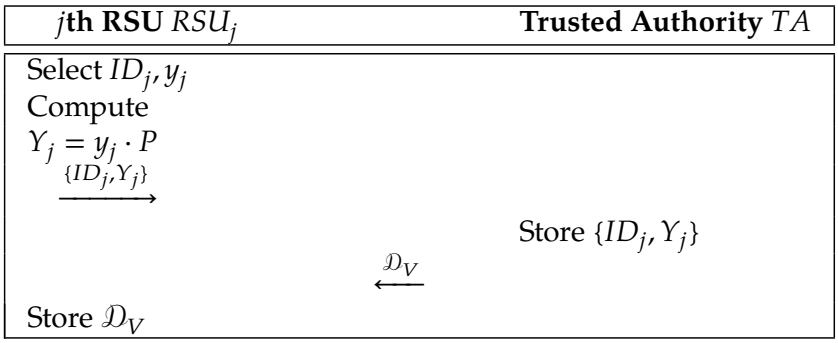


Figure 3. RSU registration phase of the proposed protocol.

6.3. AKA Phase

After completing registration, a vehicle can perform the AKA phase with an RSU. The proposed protocol supports individual verification and batch verification. During time interval, an RSU perform individual authentication phase for one request. If there are two or more requests, the RSU perform batch authentication phase [37]. The vehicle also includes traffic and service context information m in its ASCON-protected authentication request.

6.3.1. Individual AKA Phase

In the individual AKA phase, V_i and RSU_j exchange authentication messages and derive a session key. Figure 4 shows the proposed individual AKA phase and details are as follows:

Step 1: Vehicle V_i inputs ID_i^* and PW_i^* into the OBU. The OBU recovers $d_i^* = (A_1 - H_4(ID_i^* \parallel PW_i^* \parallel r_0)) \bmod q$ and $x_i^* = (B_1 - H_4(ID_i^* \parallel PW_i^* \parallel d_i^*)) \bmod q$. It computes $V_f^* = H_5(ID_i^* \parallel d_i^* \parallel x_i^* \parallel PW_i^*) \bmod n_f$ and checks $V_f^* \stackrel{?}{=} V_f$.

Step 2: V_i independently selects $b_i, e_i \in \mathbb{Z}_q^*$, generates random values r_1, n_i, prepares context information m, and generates T_1. Then, it computes $B_i = b_i P$, $E_i = e_i P$, and sets $\tau_i = enc(AID_i \| X_i \| R_i \| B_i \| K_{pub} \| T_1)$ using bit string encoding. Using τ_i, V_i computes $h_{2,i} = H_2(\tau_i)$, $h_{3,i} = H_3(\tau_i \parallel h_{2,i})$, and $\sigma_i = b_i + h_{2,i}x_i + h_{3,i}d_i \bmod q$. Next, V_i derives the request key $K_{ij} = H_6(e_i Y_j \parallel T_1)$. V_i forms $C_1 = (AID_i \parallel r_1) \oplus K_{ij}$ and $PT_1 = (\sigma_i \parallel n_i \parallel m)$. It computes $E_{K_{ij}}(r_1, AID_i \parallel T_1, PT_1) = (CT_1, Tag_1)$ and sends $\{C_1, B_i, E_i, CT_1, Tag_1, T_1\}$ to RSU_j.

Step 3: Upon receiving the request, RSU_j generates T_2 and checks $|T_2 - T_1| \leq \Delta T$. If the check fails, it aborts the phase. Otherwise, it derives $K_{ij} = H_6(y_j E_i \parallel T_1)$ and recovers $(AID_i \parallel r_1) = C_1 + K_{ij}$. Then RSU_j computes $(PT_1, \perp) = D_{K_{ij}}(r_1, AID_i \parallel T_1, CT_1, Tag_1)$. The RSU_j retrieves (X_i, R_i) corresponding to AID_i from $\mathcal{D}_V$, sets $\tau_i = enc(AID_i \| X_i \| R_i \| B_i \| K_{pub} \| T_1)$, and computes $h_{1,i} = H_1(AID_i \parallel R_i \parallel X_i \parallel K_{pub})$, $h_{2,i} = H_2(\tau_i)$ and $h_{3,i} = H_3(\tau_i \parallel h_{2,i})$. It checks $\sigma_i P \stackrel{?}{=} B_i + h_{2,i}X_i + h_{3,i}(R_i + h_{1,i}K_{pub})$. If verification succeeds, RSU_j independently selects $f_j \in \mathbb{Z}_q^*$ and generates n_j. It

computes $F_j = f_j P$ and $W_{ij} = H_6(f_j E_i \parallel T_2)$, and derives $SK_{ij} = H_7(AID_i \parallel W_{ij} \parallel y_j X_i \parallel n_i \parallel n_j \parallel m \parallel T_1 \parallel T_2)$. Then, the RSU computes $E_{W_{ij}}(r_1, AID_i \parallel T_2, n_j) = (CT_2, Tag_2)$ and sends $\{F_j, CT_2, Tag_2, T_2\}$ to V_i.

Step 4: After receiving the response, V_i checks $|T_{\text{now}} - T_2| \leq \Delta T$, where T_{now} is its current reception time. It computes $W_{ij} = H_5(e_i F_j \parallel T_2)$ and obtains $PT_2 = n_j = D_{W_{ij}}(r_1, AID_i \parallel T_2, CT_2, Tag_2)$. Finally, it derives $SK_{ij} = H_6(AID_i \parallel W_{ij} \parallel x_i Y_j \parallel n_i \parallel n_j \parallel m \parallel T_1 \parallel T_2)$.

Vehicle $Vehicle_i$	**RSU** RSU_j
Enters ID_i^*, PW_i^*	
Computes	
$d_i^* = (A_1 - H_4(ID_i^* \parallel PW_i^* \parallel r_0)) \bmod q$	
$x_i^* = (B_1 - H_4(ID_i^* \parallel PW_i^* \parallel d_i^*)) \bmod q$	Generates T_2
$V_f^* = H_5(ID_i^* \parallel d_i^* \parallel x_i^* \parallel PW_i^*) \bmod n_f$	Checks $\lvert T_2 - T_1 \rvert \leq \Delta T$
Verifies $V_f^* \stackrel{?}{=} V_f$	Computes
Selects $b_i, e_i, r_1, n_i, m, T_1$	$K_{ij} = H_6(y_j \cdot E_i \parallel T_1)$
Computes	$(AID_i \parallel r_1) = C_1 \oplus K_{ij}$
$B_i = b_i \cdot P$	$(PT_1, \perp) = D_{K_{ij}}(r_1, AID_i \parallel T_1, CT_1, Tag_1)$
$E_i = e_i \cdot P$	Sets $\tau_i = enc(AID_i \Vert X_i \Vert R_i \Vert B_i \Vert K_{pub} \Vert T_1)$
Sets $\tau_i = enc(AID_i \Vert X_i \Vert R_i \Vert B_i \Vert K_{pub} \Vert T_1)$	Computes
$h_{2,i} = H_2(\tau_i)$	$h_{1,i} = H_1(AID_i \parallel R_i \parallel X_i \parallel K_{pub})$
$h_{3,i} i = H_3(\tau_i \parallel h_{2,i})$	$h_{2,i} = H_2(\tau_i)$
$\sigma_i = b_i + h_{2,i} x_i + h_{3,i} d_i \pmod q$	$h_{3,i} = H_3(\tau_i \parallel h_{2,i})$
$K_{ij} = H_6(e_i \cdot Y_j \parallel T_1)$	Verifies
$C_1 = (AID_i \parallel r_1) \oplus K_{ij}$	$\sigma_i P \stackrel{?}{=} B_i + h_{2,i} X_i + h_{3,i}(R_i + h_{1,i} K_{pub})$
$PT_1 = (\sigma_i \parallel n_i \parallel m)$	
$E_{K_{ij}}(r_1, AID_i \parallel T_1, PT_1) = (CT_1, Tag_1)$	
$\xrightarrow{\{C_1, B_i, E_i, CT_1, Tag_1, T_1\}}$	
	Generates $f_j, n_j = PT_2$
	Computes
	$F_j = f_j \cdot P$
	$W_{ij} = H_6(f_j \cdot E_i \parallel T_2)$
	$SK_{ij} = H_7(AID_i \parallel W_{ij} \parallel y_j \cdot X_i \parallel n_i \parallel n_j \parallel m \parallel T_1 \parallel T_2)$
Generates T_3	$E_{W_{ij}}(r_1, AID_i \parallel T_2, n_j) = (CT_2, Tag_2)$
	$\xleftarrow{\{F_j, CT_2, Tag_2, T_2\}}$
Checks $\lvert T_3 - T_2 \rvert \leq \Delta T$	
Computes	
$W_{ij} = H_6(e_i \cdot F_j \parallel T_2)$	
$(PT_2, \perp) = D_{W_{ij}}(r_1, AID_i \parallel T_2, CT_2, Tag_2)$	
$SK_{ij} = H_7(AID_i \parallel W_{ij} \parallel x_i \cdot Y_j \parallel n_i \parallel n_j \parallel m \parallel T_1 \parallel T_2)$	

Figure 4. Individual AKA phase of the proposed protocol.

6.3.2. Batch AKA Phase

The batch AKA phase processes multiple requests collected by the same RSU_j within a short interval. Each vehicle follows the individual phase, generates $B_i, E_i, \sigma_i, K_{ij}$, and sends $\{C_1, B_i, E_i, CT_1, Tag_1, T_1\}$. For each request passing the timestamp and ASCON checks, the RSU obtains $(AID_i, \sigma_i, n_i, m)$, retrieves the corresponding parameters, and computes $h_{1,i}, h_{2,i}$, and $h_{3,i}$. After the requests are fixed, it independently selects random coefficients $s_i \in \mathbb{Z}_q^*$ and checks

$$\left(\sum_{i=1}^{n} s_i \sigma_i\right) P \stackrel{?}{=} \sum_{i=1}^{n} s_i B_i + \sum_{i=1}^{n} s_i h_{2,i} X_i + \sum_{i=1}^{n} s_i h_{3,i} R_i + \left(\sum_{i=1}^{n} s_i h_{3,i} h_{1,i}\right) K_{pub}.$$

If the batch check fails, the RSU verifies the signatures individually to identify invalid requests. For each accepted request, it generates a separate ephemeral secret f_j and $F_j = f_j P$, as well as n_j, and computes $W_{ij} = H_6(f_j E_i \parallel T_2)$ and $SK_{ij} = H_7(AID_i \parallel W_{ij} \parallel y_j X_i \parallel n_i \parallel n_j \parallel m \parallel T_1 \parallel T_2)$. It sends $\{F_j, CT_2, Tag_2, T_2\}$ as in the individual phase, and each vehicle derives its own session key.

7. Security Analysis

By analyzing the following security properties, we show that the proposed protocol can resist practical attacks in IoV environments.

7.1. Informal Security Analysis

By analyzing the following security properties, we show that the proposed protocol 453 can resist practical attacks in IoV environments.

7.1.1. Session-Key Disclosure Attacks

In the proposed protocol, the session key is computed as $SK_{ij} = H_7(AID_i \parallel W_{ij} \parallel x_i \cdot Y_j \parallel n_i \parallel n_j \parallel m \parallel T_1 \parallel T_2)$ by V_i and $SK_{ij} = H_7(AID_i \parallel W_{ij} \parallel y_j \cdot X_i \parallel n_i \parallel n_j \parallel m \parallel T_1 \parallel T_2)$ by RSU_j. Although $\mathcal{A}$ may eavesdrop on $\{C_1, B_i, E_i, CT_1, Tag_1, T_1\}$ and $\{F_j, CT_2, Tag_2, T_2\}$, it cannot derive $x_i \cdot Y_j = y_j \cdot X_i$ without knowing x_i or y_j. Moreover, $\mathcal{A}$ needs e_i or f_j to compute W_{ij} Therefore, $\mathcal{A}$ cannot obtain all inputs of SK_{ij}, and the proposed protocol resists session-key disclosure attacks.

7.1.2. Replay and MitM Attacks

The proposed protocol uses timestamps T_1 and T_2 to verify message freshness, preventing expired replayed messages from being accepted. In addition, authentication messages are protected by ASCON. To forge or modify $\{C_1, B_i, E_i, CT_1, Tag_1, T_1\}$, $\mathcal{A}$ must generate a valid ASCON tag and a valid signature $\sigma_i = b_i + h_{2,i}x_i + h_{3,i}d_i \pmod q$, which requires the vehicle's signing key (x_i, d_i). Similarly, forging $\{F_j, CT_2, Tag_2, T_2\}$ requires generating a valid W_{ij}. Thus, $\mathcal{A}$ cannot construct valid forged messages, and the proposed protocol resists replay and MitM attacks.

7.1.3. Physical-Capture Attacks

Assume that $\mathcal{A}$ captures the OBU of V_i and extracts $\{AID_i, A_1, B_1, V_f, r_0, X_i, R_i\}$. Even with these values, $\mathcal{A}$ cannot recover the complete private key (x_i, d_i) without the correct identity ID_i and password PW_i, because x_i and d_i are protected by A_1, B_1, and verified through V_f. Moreover, generating a valid authentication request requires a valid signature σ_i, which cannot be produced without (x_i, d_i). Therefore, the proposed protocol withstands physical-capture attacks.

7.1.4. Untraceability and Anonymity

An external adversary $\mathcal{A}$ can attempt to get identities of vehicles or track static values in multiple sessions. The real identity ID_i is never transmitted over the public channel. Instead, the proposed protocol uses AID_i, which is generated using the TA's master secrets and cannot be reversed by $\mathcal{A}$. Furthermore, AID_i is concealed in $C_1 = (AID_i \parallel r_1) \oplus K_{ij}$ during the AKA phase. Since fresh values b_i, r_1, n_i, and timestamps are used in each session, the transmitted messages are different for every execution. Therefore, $\mathcal{A}$ cannot recover ID_i or link multiple sessions of the same vehicle.

7.1.5. Impersonation Attacks

To impersonate V_i, $\mathcal{A}$ must generate a valid message $\{C_1, B_i, E_i, CT_1, Tag_1, T_1\}$. However, this requires a valid signature σ_i, which can only be generated using the complete signing key (x_i, d_i). To impersonate RSU_j, $\mathcal{A}$ must generate a valid ASCON response $\{F_j, CT_2, Tag_2, T_2\}$ using $W_{ij} = H_6(f_j \cdot E_i \parallel T_2)$. Since $\mathcal{A}$ does not know x_i, d_i, or y_j, the proposed protocol prevents both vehicle and RSU impersonation attacks.

7.1.6. Insider Attacks

A legitimate but malicious insider may possess its own credentials, but these credentials are bound to its own secret values. Since each vehicle has an independent secret value x_i and partial private key d_i, an insider cannot derive another vehicle's signing key from public information such as AID_i, X_i, and R_i. In addition, the session key includes $x_i \cdot Y_j$ and W_{ij} , with fresh nonces protected by ASCON. Therefore, a registered insider cannot compute session keys of other entities or impersonate them.

7.1.7. DoS Attacks

An adversary may flood the RSU with requests $\{C_1, B_i, E_i, CT_1, Tag_1, T_1\}$. In DoS attack, $\mathcal{A}$ sends numerous forged request messages $\{C_1, B_i, E_i, CT_1, Tag_1, T_1\}$ to RSU_j to interrupt the authentication process and vehicular services. In the proposed protocol, RSU_j checks the timestamp $|T_2 - T_1| \leq \Delta T$ and verifies Tag_1 and σ_i. Since $\mathcal{A}$ does not know the secret values required to generate a valid signature, the forged requests are rejected, and RSU_j does not proceed to the subsequent steps. Therefore, the proposed protocol mitigates DoS attacks.

7.1.8. Offline Password-Guessing Attacks

$\mathcal{A}$ can attempt to capture a legitimate vehicle's OBU and extracts the stored parameters $\{AID_i, A_1, B_1, V_f, r_0, X_i, R_i\}$. Then, $\mathcal{A}$ guesses the vehicle's identity ID_i^* and password PW_i^* and computes d_i^*, x_i^*, and V_f^* to obtain PW_i. However, since V_f is computed using fuzzy verifier, $\mathcal{A}$ cannot identify the correct PW_i. Thus, proposed protocol resists offline password-guessing attacks.

7.1.9. Stolen-Verifier Attacks

The RSU stores public verification information such as (AID_i, X_i, R_i) for verification. Even if $\mathcal{A}$ obtains these values, it cannot derive the complete private key (x_i, d_i) or generate a valid signature σ_i. In addition, the session key requires $x_i \cdot Y_j$, W_{ij}, and nonces n_i and n_j. Therefore, stolen verifier information does not allow $\mathcal{A}$ to impersonate a vehicle or compute the session key.

7.1.10. Ephemeral Secret Leakage Attacks

Assume that $\mathcal{A}$ obtains limited ephemeral values such as b_i or r_1. Even in this case, $\mathcal{A}$ cannot compute SK_{ij} because the session key still requires $x_i \cdot Y_j$, W_{ij}, and the nonces n_i and n_j, which are protected by ASCON. Moreover, knowing b_i alone does not reveal the complete signing key (x_i, d_i) from $\sigma_i = b_i + h_{2,i}x_i + h_{3,i}d_i \pmod{q}$. Therefore, the proposed protocol resists ephemeral secret leakage attacks.

7.1.11. Perfect Forward Secrecy

Suppose that an adversary $\mathcal{A}$ obtains the long-term secret keys of V_i and RSU_j. Although $\mathcal{A}$ can compute the $x_iY_j = y_jX_i$, it cannot obtain $e_iF_j = f_jE_i$ from the transmitted values E_i and F_j. Therefore, $\mathcal{A}$ cannot recover W_{ij} and the session key SK_{ij} without ephemeral secrets e_i and f_j. Thus, the proposed protocol supports perfect forward secrecy.

7.2. Formal Security Proof Under the Random Oracle Model

In this section, we prove the security of the proposed pairing-free certificateless authentication mechanism under the ROM [11]. Since the proposed protocol uses certificateless signatures and supports batch verification, we consider Type 1, Type 2, and Type 3 adversaries. A Type 1 adversary can replace the user's public key X_i but cannot obtain the TA's master secret key. A Type 2 adversary knows the TA's master secret key but cannot replace

public keys or obtain the user's secret value. A Type 3 adversary attempts to forge a valid aggregated signature result using multiple invalid signatures.

Theorem 1. *Assume that ECDLP is hard. Then, the proposed protocol is existentially unforgeable against a Type 1 adversary $\mathcal{A}_1$ under adaptive chosen-message attacks.*

Proof. Assume that a probabilistic polynomial-time Type 1 adversary $\mathcal{A}_1$ can forge a valid signature with non-negligible probability. Then, we construct a challenger $\mathcal{C}$ that uses $\mathcal{A}_1$ to solve an ECDLP instance. Given an ECDLP instance (P, aP), the goal of $\mathcal{C}$ is to compute a.

Initialization Phase: $\mathcal{C}$ initializes the system parameters $\{G, q, P, T_{pub}, K_{pub}, H_0 - H_7\}$ where $\mathcal{C}$ sets $K_{pub} = aP$. For the $AID = AID_i^*$, $\mathcal{C}$ selects $x_i, r_i \in \mathbb{Z}_q^*$ and computes $X_i = x_iP$ and $R_i = r_iP$. All hash functions are random oracles, and $\mathcal{C}$ maintains the corresponding hash lists.

Query Phase: $\mathcal{A}_1$ can adaptively issue the following queries.

- *Hash queries:* For an H_1 query, $\mathcal{A}_1$ submits the tuple $(AID_i, R_i, X_i, K_{pub})$ to $\mathcal{C}$. If the corresponding $(AID_i, R_i, X_i, K_{pub}, h_{1,i})$ exists in L_{H_1}, $\mathcal{C}$ returns the stored $h_{1,i}$. Otherwise, it randomly selects $h_{1,i} \in \mathbb{Z}_q^*$, inserts $(AID_i, R_i, X_i, K_{pub}, h_{1,i})$ into L_{H_1}, and returns $h_{1,i}$ to $\mathcal{A}_1$.
  For an H_2 query, $\mathcal{A}_1$ submits τ_i to $\mathcal{C}$. If the corresponding $(\tau_i, h_{2,i})$ exists in L_{H_2}, $\mathcal{C}$ returns the stored $h_{2,i}$. Otherwise, it randomly selects $h_{2,i} \in \mathbb{Z}_q^*$, inserts $(\tau_i, h_{2,i})$ into L_{H_2}, and returns $h_{2,i}$ to $\mathcal{A}_1$.
  For an H_3 query, $\mathcal{A}_1$ submits $\tau_i \parallel h_{2,i}$ to $\mathcal{C}$. If the corresponding $(\tau_i \parallel h_{2,i}, h_{3,i})$ exists in L_{H_3}, $\mathcal{C}$ returns the stored $h_{3,i}$. Otherwise, it randomly selects $h_{3,i} \in \mathbb{Z}_q^*$, inserts $(\tau_i \parallel h_{2,i}, h_{3,i})$ into L_{H_3}, and returns $h_{3,i}$ to $\mathcal{A}_1$.
- *Partial-private-key queries:* $\mathcal{A}_1$ requests the partial private key corresponding to AID_i. If $AID_i = AID_i^*$, $\mathcal{C}$ aborts the simulation. Otherwise, $\mathcal{C}$ returns the corresponding partial private key d_i.
- *Private-key queries:* $\mathcal{A}_1$ submits AID_i to request the private key. If $AID_i = AID_i^*$, $\mathcal{C}$ aborts the simulation. Otherwise, $\mathcal{C}$ checks whether L_{pk} contains the corresponding $(AID_i, X_i, R_i, x_i, d_i)$. If it exists, $\mathcal{C}$ returns $sk_i = (x_i, d_i)$. Otherwise, $\mathcal{C}$ initiates the partial-private-key query procedure to generate and store the corresponding key information in L_{pk}. Finally, $\mathcal{C}$ returns $sk_i = (x_i, d_i)$ to $\mathcal{A}_1$.
- *Public-key queries:* $\mathcal{A}_1$ submits AID_i to $\mathcal{C}$. If the corresponding value exists in L_{pk}, $\mathcal{C}$ returns the stored (X_i, R_i). Otherwise, if $AID_i = AID_i^*$, $\mathcal{C}$ uses $X_i = x_iP$ and $R_i = r_iP$ generated during initialization. For $AID_i \neq AID_i^*$, $\mathcal{C}$ randomly selects $x_i, d_i, h_{1,i} \in \mathbb{Z}_q^*$ and computes $X_i = x_iP$ and $R_i = d_iP - h_{1,i}K_{pub}$. Then, $\mathcal{C}$ sets $H_1(AID_i \parallel R_i \parallel X_i \parallel K_{pub}) = h_{1,i}$, stores the corresponding values in L_{pk} and L_{H_1}, and returns (X_i, R_i) to $\mathcal{A}_1$.
- *Public-key-replacement queries:* As a Type 1 adversary, $\mathcal{A}_1$ may replace X_i with $\widetilde{X}_i$. $\mathcal{C}$ updates $\widetilde{X}_i$ to the key list L_{pk}.
- *Signing queries:* $\mathcal{A}_1$ submits (AID_i, T_1) to $\mathcal{C}$. $\mathcal{C}$ retrieves the corresponding public key from L_{pk} or initiates a public-key query if necessary. For $AID \neq AID_i^*$, $\mathcal{C}$ generates valid signatures normally. For $AID = AID_i^*$, $\mathcal{C}$ selects $z, c, g \in \mathbb{Z}_q^*$ and computes $B_i = zP - c\widetilde{X}_i - g(R_i + h_{1,i}K_{pub})$. Finally, $\mathcal{C}$ programs the unused hash outputs $H_2(\tau_i) = c$ and $H_3(\tau_i \parallel c) = g$, and returns $(B_i, \sigma_i = z)$ to $\mathcal{A}_1$.

Forgery: Finally, $\mathcal{A}_1$ outputs a valid forged signature σ_i^* for the $AID = AID_i^*$. The forgery satisfies

$$\sigma_i^* P = B_i^* + h_{2,i}\widetilde{X}_i + h_{3,i}^*(R_i + h_{1,i}K_{pub}).$$

By the forking lemma, C rewinds A_1 with the same random tape but a different response $h_{3,i}^{**}$ to the corresponding random oracle query. Then, A_1 outputs another valid forgery satisfying

$$\sigma_i^{**}P = B_i^* + h_{2,i}\widetilde{X}_i + h_{3,i}^{**}(R_i + h_{1,i}K_{pub}).$$

Subtracting the two equations gives

$$(\sigma_i^* - \sigma_i^{**})P = (h_{3,i}^* - h_{3,i}^{**})(R_i + h_{1,i}K_{pub}).$$

Then, C calculates $a = [(\sigma_i^* - \sigma_i^{**})(h_{3,i}^* - h_{3,i}^{**})^{-1} - r_i]h_{1,i}^{-1} \pmod q$ and solve the ECDLP instance (P, aP). This contradicts the hardness of ECDLP. Therefore, the proposed protocol is secure against a Type 1 adversary. □

Theorem 2. *Assume that ECDLP is hard. Then, the proposed protocol is existentially unforgeable against a Type 2 adversary A_2 under adaptive chosen-message attacks.*

Proof. Assume that a probabilistic polynomial-time Type 2 adversary A_2 can forge a valid authentication signature σ_i^* with non-negligible probability. Then, we construct a challenger C that uses A_2 to solve an ECDLP instance (P, aP).

Initialization Phase: In the Type 2 adversary model, A_2 knows the TA's master secret key but cannot replace public keys or obtain the target user's secret value. Therefore, C selects β, sets $K_{pub} = \beta P$, and gives β to A_2.

Query Phase: A_2 can adaptively issue the following queries.

- *Hash queries:* For an H_1, H_2 and H_3 queries, similarly in Theorem 1, C checks the corresponding hash list L_{H_1}, L_{H_2}, or L_{H_3}. If the queried input exists, C returns the stored value. Otherwise, it selects a random value from $\mathbb{Z}_q^*$, stores the input–output pair, and returns the selected value.
- *Partial-private-key queries:* A_2 submits AID_i to initiate this query. If $AID_i = AID_i^*$, C aborts the simulation. Otherwise, C first checks whether the corresponding value exists in L_{pk}. If it exists, C directly returns d_i to A_1. If not, C randomly selects $x_i, d_i, h_{1,i} \in \mathbb{Z}_q^*$ and computes $X_i = x_iP$ and $R_i = d_iP - h_{1,i}K_{pub}$. Finally, C returns d_i to A_2 and inserts $(AID_i, X_i, R_i, x_i, d_i)$ and $(AID_i, R_i, X_i, K_{pub}, h_{1,i})$ into L_{pk} and L_{H_1}, respectively.
- *Private-key queries:* A_2 submits AID_i to request the private key. If $AID_i = AID_i^*$, C aborts the simulation. Otherwise, C checks whether L_{pk} contains the corresponding $(AID_i, X_i, R_i, x_i, d_i)$. If it exists, C returns $sk_i = (x_i, d_i)$. Otherwise, C initiates the partial-private-key query procedure to generate and store the corresponding key information in L_{pk}. Finally, C returns $sk_i = (x_i, d_i)$ to A_2.
- *Public-key queries:* A_2 submits AID_i to C. If the corresponding value exists in L_{pk}, C returns the stored (X_i, R_i). Otherwise, if $AID_i = AID_i^*$, C uses $X_i = x_iP$ and $R_i = r_iP$ generated during initialization. For $AID_i \neq AID_i^*$, C randomly selects $x_i, d_i, h_{1,i} \in \mathbb{Z}_q^*$ and computes $X_i = x_iP$ and $R_i = d_iP - h_{1,i}K_{pub}$. Then, C sets $H_1(AID_i \parallel R_i \parallel X_i \parallel K_{pub}) = h_{1,i}$, stores the corresponding values in L_{pk} and L_{H_1}, and returns (X_i, R_i) to A_2.
- *Signing queries:* For $AID_i \neq AID_i^*$, C generates valid signatures using the corresponding private keys. For $AID_i = AID_i^*$, C randomly selects $z, c, g \in \mathbb{Z}_q^*$ and computes $B_i = zP - cX_i - g(R_i + h_{1,i}K_{pub})$. It programs the unused hash outputs as $H_2(\tau_i) = c$ and $H_3(\tau_i \parallel c) = g$, and returns $(B_i, \sigma_i = z)$ to A_2.

Forgery: Finally, A_2 outputs a valid forged signature σ_i^* for the AID_i^*, satisfying

$$\sigma_i^*P = B_i^* + h_{2,i}^*X_i + h_{3,i}^*(R_i + h_{1,i}K_{pub}).$$

Using the forking lemma, C obtains another valid forgery σ_i^{**} with a different random oracle output $h_{2,i}^{**}$:

$$\sigma_i^{**}P = B_i^* + h_{2,i}^{**}X_i + h_{3,i}^{**}(R_i + h_{1,i}K_{pub}).$$

Subtracting the two equations gives

$$(\sigma_i^* - \sigma_i^{**})P = (h_{2,i}^* - h_{2,i}^{**})X_i + (h_{3,i}^* - h_{3,i}^{**})d_iP.$$

Them, $\mathbb{C}$ can calculates $a = [\sigma_i^* - \sigma_i^{**} - (h_{3,i}^* - h_{3,i}^{**})d_i](h_{2,i}^* - h_{2,i}^{**})^{-1} \pmod q$ and solve the ECDLP instance (a, aP). This contradicts the hardness of ECDLP. Therefore, the proposed protocol is secure against a Type 2 adversary. □

Theorem 3. *Assume that RSU independently and uniformly selects $s_i \in \mathbb{Z}_q^*$ during batch verification. Then, the proposed protocol is secure against a Type 3 adversary $\mathbb{A}_3$ under collusion attacks.*

Proof. Assume that a probabilistic polynomial-time Type 3 adversary $\mathbb{A}_3$ can forge a valid aggregate signature with non-negligible probability. Then, we construct a challenger $\mathbb{C}$ that uses $\mathbb{A}_3$ to pass the batch verification.

Initialization Phase: In the Type 3 adversary model, $\mathbb{A}_3$ attempts to make an invalid batch of signatures pass batch verification. Therefore, $\mathbb{C}$ selects $\alpha, \beta \in \mathbb{Z}_q^*$, computes $T_{pub} = \alpha P$ and $K_{pub} = \beta P$, and initializes the key and hash lists to respond to $\mathbb{A}_3$'s queries.

Query Phase: $\mathbb{A}_3$ can adaptively issue the following queries.

- *Partial-private-key queries:* $\mathbb{A}_3$ requests the partial private key corresponding to AID_i. If the corresponding $(AID_i, X_i, R_i, x_i, d_i)$ exists in L_{pk}, $\mathbb{C}$ returns the stored d_i. Otherwise, $\mathbb{C}$ selects $x_i, r_i \in \mathbb{Z}_q^*$, computes $X_i = x_iP$, $R_i = r_iP$, $h_{1,i} = H_1(AID_i \parallel R_i \parallel X_i \parallel K_{pub})$, and $d_i = (r_i + h_{1,i}\beta) \bmod q$. Then, $\mathbb{C}$ stores the corresponding values in L_{pk} and returns d_i to $\mathbb{A}_3$.
- *Private-key queries:* $\mathbb{A}_3$ requests the private key corresponding to AID_i. $\mathbb{C}$ checks whether the corresponding $(AID_i, X_i, R_i, x_i, d_i)$ exists in L_{pk}. If not, it initiates the partial-private-key query procedure to generate the key information. Finally, $\mathbb{C}$ returns the private key $sk_i = (x_i, d_i)$ to $\mathbb{A}_3$.
- *Public-key queries:* $\mathbb{A}_3$ requests the public key corresponding to AID_i. If the corresponding $(AID_i, X_i, R_i, x_i, d_i)$ exists in L_{pk}, $\mathbb{C}$ returns the stored (X_i, R_i). Otherwise, it initiates the partial-private-key query procedure and returns the generated public key (X_i, R_i) to $\mathbb{A}_3$.
- *Batch-verification queries:* $\mathbb{A}_3$ submits a batch of n signatures $\{(AID_i, B_i, \sigma_i, T_{1,i})\}_{i=1}^n$ to $\mathbb{C}$. Then, $\mathbb{C}$ independently selects random coefficients $s_i \in \mathbb{Z}_q^*$ and verifies the signatures using the proposed batch verification equation. Finally, $\mathbb{C}$ returns the verification result to $\mathbb{A}_3$.

Forgery Phase: Finally, $\mathbb{A}_3$ outputs a batch of n signatures $\{(AID_i, B_i, \sigma_i, T_{1,i})\}_{i=1}^n$. $\mathbb{C}$ retrieves the corresponding public keys and hash values from the stored lists. For each signature, let

$$\Delta_i = \sigma_iP - B_i - h_{2,i}X_i - h_{3,i}(R_i + h_{1,i}K_{pub}) = \delta_iP,$$

where $\delta_i \in \mathbb{Z}_q$. $\mathbb{A}_3$ succeeds if the following two conditions hold.

First, the n signatures satisfy the batch verification equation:

$$\left(\sum_{i=1}^n s_i\sigma_i\right)P \stackrel{?}{=} \sum_{i=1}^n s_iB_i + \sum_{i=1}^n s_ih_{2,i}X_i + \sum_{i=1}^n s_ih_{3,i}R_i + \left(\sum_{i=1}^n s_ih_{3,i}h_{1,i}\right)K_{pub}.$$

Therefore, the batch verification condition implies

$$\sum_{i=1}^n s_i\delta_i = 0 \pmod q.$$

Second, at least one individual signature is invalid. Without loss of generality, suppose that the jth signature is invalid, so that $\delta_j \neq 0$. Then, the batch verification condition requires

$$s_j = -\delta_j^{-1} \sum_{\substack{i=1 \\ i \neq j}}^{n} s_i \delta_i \pmod{q}.$$

Since the coefficients $s_i \in \mathbb{Z}_q^*$ are independently selected after receiving the complete batch, $\mathcal{A}_3$ cannot predict them in advance. Furthermore, when at least one individual signature is invalid, the probability that the randomly selected coefficients satisfy the batch verification equation is negligible. Therefore, the proposed batch verification mechanism is secure against a Type 3 adversary. □

From Theorems 1–3, the proposed pairing-free certificateless authentication mechanisms resist Type 1, Type 2, and Type 3 adversaries under the random oracle models.

7.3. Formal Analysis Under BAN Logic

We employ BAN logic to demonstrate that the proposed protocol achieves mutual authentication and secure session key agreement between V_i and RSU_j. The notations used in BAN logic are summarized in Table 2.

Table 2. Notation for BAN logic.

Notation	Description	Notation	Description
N_1, N_2	Principals	S_1, S_2	Statements
$N_1 \mid\equiv S_1$	N_1 believes S_1	$N_1 \mid\sim S_1$	N_1 once said S_1
$N_1 \Rightarrow S_1$	N_1 controls S_1	$N_1 \lhd S_1$	N_1 receives S_1
$\#(S_1)$	S_1 is fresh	$\{S_1\}_K$	S_1 is protected with key K
$N_1 \overset{K}{\leftrightarrow} N_2$	N_1 and N_2 share the secret key K		

7.3.1. Rules

The BAN logic rules used in this analysis are given as follows.

1. Message Meaning Rule (MMR)

$$\frac{N_1 \mid\equiv N_1 \overset{K}{\leftrightarrow} N_2,\ N_1 \lhd \{S_1\}_K}{N_1 \mid\equiv N_2 \mid\sim S_1}$$

2. Nonce Verification Rule (NVR)

$$\frac{N_1 \mid\equiv \#(S_1),\ N_1 \mid\equiv N_2 \mid\sim S_1}{N_1 \mid\equiv N_2 \mid\equiv S_1}$$

3. Jurisdiction Rule (JR)

$$\frac{N_1 \mid\equiv N_2 \Rightarrow S_1,\ N_1 \mid\equiv N_2 \mid\equiv S_1}{N_1 \mid\equiv S_1}$$

4. Belief Rule (BR)

$$\frac{N_1 \mid\equiv (S_1, S_2)}{N_1 \mid\equiv S_1}$$

5. Freshness Rule (FR)

$$\frac{N_1 \mid\equiv \#(S_1)}{N_1 \mid\equiv \#(S_1, S_2)}$$

7.3.2. Goals

The goals of the BAN logic analysis are defined as follows.

Goal 1: $V_i \mid\equiv V_i \overset{SK_{ij}}{\leftrightarrow} RSU_j$

Goal 2: $V_i \mid\equiv RSU_j \mid\equiv V_i \overset{SK_{ij}}{\leftrightarrow} RSU_j$

Goal 3: $RSU_j \mid\equiv V_i \overset{SK_{ij}}{\leftrightarrow} RSU_j$

Goal 4: $RSU_j \mid\equiv V_i \mid\equiv V_i \overset{SK_{ij}}{\leftrightarrow} RSU_j$

7.3.3. Idealized Forms

In the proposed protocol, V_i sends $\{C_1, B_i, E_i, CT_1, Tag_1, T_1\}$ to RSU_j, and RSU_j sends $\{F_j, CT_2, Tag_2, T_2\}$ to V_i. Since CT_1 and CT_2 are protected using ASCON with K_{ij} and W_{ij}, respectively, the messages are idealized as follows.

Message 1: $V_i \rightarrow RSU_j : B_i, E_i, \{AID_i, \sigma_i, n_i, m, T_1\}_{K_{ij}}$
Message 2: $RSU_j \rightarrow V_i : F_j, \{AID_i, n_j, T_2\}_{W_{ij}}$

7.3.4. Assumptions

The following assumptions are used for the BAN logic analysis.

A_1: $RSU_j \mid\equiv \#(T_1)$
A_2: $V_i \mid\equiv \#(T_2)$
A_3: $V_i \mid\equiv V_i \overset{W_{ij}}{\leftrightarrow} RSU_j$
A_4: $RSU_j \mid\equiv V_i \overset{K_{ij}}{\leftrightarrow} RSU_j$
A_5: $V_i \mid\equiv RSU_j \Rightarrow (V_i \overset{SK_{ij}}{\leftrightarrow} RSU_j)$
A_6: $RSU_j \mid\equiv V_i \Rightarrow (V_i \overset{SK_{ij}}{\leftrightarrow} RSU_j)$

7.3.5. BAN Logic Proof

We conduct the BAN logic analysis of the proposed protocol as follows.

Step 1: S_1 can be obtained from Message 1.

$$S_1 : \ RSU_j \lhd B_i, E_i, \{AID_i, \sigma_i, n_i, m, T_1\}_{K_{ij}}$$

Step 2: S_2 can be induced by applying MMR to S_1 and A_4.

$$S_2 : \ RSU_j \mid\equiv V_i \mid\sim (AID_i, \sigma_i, n_i, m, T_1)$$

Step 3: S_3 can be induced by applying FR using A_1.

$$S_3 : \ RSU_j \mid\equiv \#(AID_i, \sigma_i, n_i, m, T_1)$$

Step 4: S_4 can be induced by applying NVR to S_2 and S_3.

$$S_4 : \ RSU_j \mid\equiv V_i \mid\equiv (AID_i, \sigma_i, n_i, m, T_1)$$

Step 5: S_5 can be induced by applying BR using S_4.

$$S_5 : \ RSU_j \mid\equiv V_i \mid\equiv n_i$$

Step 6: S_6 can be obtained from Message 2.

$$S_6 : \ V_i \lhd F_j, \{AID_i, n_j, T_2\}_{W_{ij}}$$

Step 7: S_7 can be induced by applying MMR to S_6 and A_3.

$$S_7 : \ V_i \mid\equiv RSU_j \mid\sim (AID_i, n_j, T_2)$$

Step 8: S_8 can be induced by applying FR using A_2.

$$S_8 : \ V_i \mid\equiv \#(AID_i, n_j, T_2)$$

Step 9: S_9 can be induced by applying NVR to S_7 and S_8.

$$S_9 : \ V_i \mid\equiv RSU_j \mid\equiv (AID_i, n_j, T_2)$$

Step 10: S_{10} can be induced by applying BR using S_9.

$$S_{10}: \ V_i \mid\equiv RSU_j \mid\equiv n_j$$

Step 11: S_{11} and S_{12} can be induced by S_5 and S_{10}. V_i and RSU_j can compute the same session key $SK_{ij} = H_7(AID_i \parallel W_{ij} \parallel x_i \cdot Y_j \parallel n_i \parallel n_j \parallel m \parallel T_1 \parallel T_2)$.

$$S_{11}: \ V_i \mid\equiv RSU_j \mid\equiv (V_i \overset{SK_{ij}}{\leftrightarrow} RSU_j)(\textbf{Goal 2})$$

$$S_{12}: \ RSU_j \mid\equiv V_i \mid\equiv (V_i \overset{SK_{ij}}{\leftrightarrow} RSU_j)(\textbf{Goal 4})$$

Step 12: S_{13} and S_{14} can be induced by applying the JR using S_{11} and A_5, and S_{12} and A_6, respectively.

$$S_{13}: \ V_i \mid\equiv (V_i \overset{SK_{ij}}{\leftrightarrow} RSU_j)(\textbf{Goal 1})$$

$$S_{14}: \ RSU_j \mid\equiv (V_i \overset{SK_{ij}}{\leftrightarrow} RSU_j)(\textbf{Goal 3})$$

7.4. Formal Analysis Under Scyther Tool

To further validate the security of the proposed protocol, we perform a formal verification using the Scyther tool [13,14]. The individual AKA phase between V_i and RSU_j is modeled in Security Protocol Description Language (SPDL), and the verification is conducted using the Scyther GUI. In the SPDL model, ASCON-protected messages are abstracted as encrypted messages under K_{ij} and W_{ij}, and the session key is $SK_{ij} = H_7(AID_i \parallel W_{ij} \parallel x_i \cdot Y_j \parallel n_i \parallel n_j \parallel m \parallel T_1 \parallel T_2)$. We verify several security claims, including secrecy, aliveness, weak agreement, non-injective agreement, and non-injective synchronization. The descriptions of these claims are summarized in Table 3.

As shown in Figure 5, all claims of both V_i and RSU_j are verified as *OK*. In particular, Scyther reports No attacks within bounds for the secrecy claims of n_i and n_j, as well as for the aliveness, weak agreement, non-injective agreement, and non-injective synchronization claims. Therefore, the Scyther verification results confirm that the proposed individual AKA phase achieves secrecy, mutual authentication, and synchronization between V_i and RSU_j.

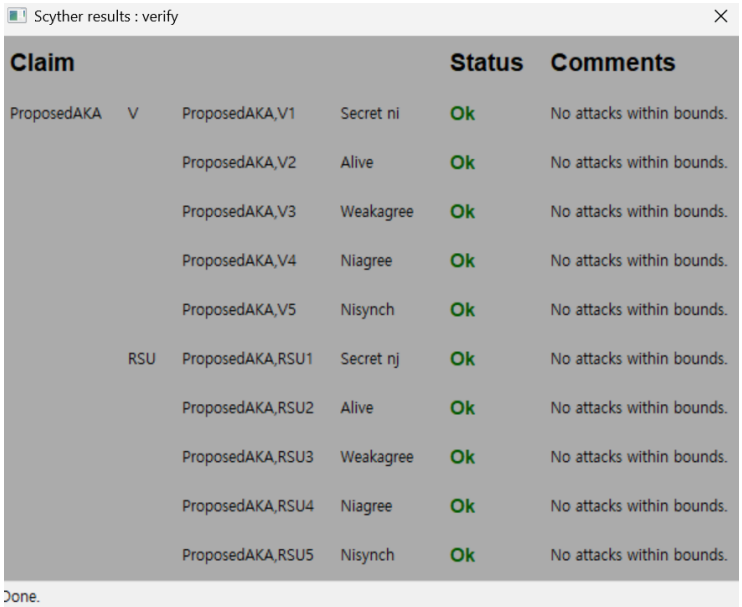


Figure 5. Scyther verification results of the proposed protocol.

Table 3. Descriptions of claim events in Scyther.

Claim Event	Description
Aliveness	Ensures that the communicating party has been active during the protocol execution.
Weak agreement	Guarantees that the communicating party is legitimate.
Non-injective agreement	Ensures that both parties agree on the exchanged data and the identity of the communication partner.
Non-injective synchronization	Guarantees that all protocol messages are exchanged in the correct order while satisfying aliveness and agreement properties.

8. Performance Analysis

To evaluate the practicality of the proposed protocol, we compare it with existing protocols [10,26–30] for IoV environments. We evaluate the compared protocols in terms of security features, computational cost, and communication cost. The simulation-based evaluation using NS-3/SUMO presents the performance of the proposed protocol in a practical environment.

8.1. Comparison of Security Features

We compare the security features of the proposed protocol with those of existing IoV authentication protocols. The considered security features are defined as follows: SF1: resistance to session-key disclosure attacks; SF2: resistance to replay and MitM attacks; SF3: resistance to physical capture attacks; SF4: support for vehicle anonymity; SF5: support for vehicle untraceability; SF6: resistance to impersonation attacks; SF7: resistance to forgery attacks; SF8: resistance to insider attacks; SF9: resistance to stolen-verifier attacks; SF10: resistance to offline password-guessing attacks; SF11: resistance to ephemeral-secret leakage attacks; SF12: support for mutual authentication; and SF13: support for perfect forward secrecy.

As shown in Table 4, the proposed protocol satisfies all the considered security requirements by combining ASCON-based message protection with pairing-free certificateless authentication. Therefore, the proposed protocol provides stronger security guarantees than the compared protocols [10,26–30] and is suitable for secure IoV environments.

Table 4. Security features.

Security Features	[26]	[27]	[28]	[29]	[30]	[10]	Proposed
SF1	∘	∘	∘	∘	∘	×	∘
SF2	∘	∘	∘	∘	∘	×	∘
SF3	∘	∘	∘	∘	×	×	∘
SF4	×	∘	×	∘	∘	∘	∘
SF5	×	×	×	×	∘	×	∘
SF6	×	∘	∘	×	∘	×	∘
SF7	×	∘	∘	∘	∘	×	∘
SF8	∘	∘	∘	∘	∘	∘	∘
SF9	∘	∘	∘	∘	∘	∘	∘
SF10	∘	∘	∘	∘	∘	∘	∘
SF11	∘	×	×	∘	∘	×	∘
SF12	∘	∘	∘	∘	∘	×	∘
SF13	∘	∘	∘	∘	∘	∘	∘

∘: "security feature is satisfied." ×: "security feature is not satisfied.".

8.2. Computation Cost Comparison

We further evaluate the computational efficiency of the proposed protocol by comparing the cryptographic operations required during the AKA phase. The main operations

considered in this analysis include one-way hash computation, elliptic curve scalar multiplication, elliptic curve point addition, and ASCON encryption/decryption. Let T_H, T_{Mul}, T_{ECA}, and T_{ASCON} denote the execution time of a one-way hash operation, an ECC scalar multiplication, an ECC point addition, and an ASCON encryption/decryption operation, respectively. According to the experimental results in [38], shown in Table 5, we use $T_H = 0.0002$ ms, $T_{Mul} = 0.5371$ ms, $T_{ECA} = 0.0038$ ms, and $T_{ASCON} = 0.0002$ ms.

Table 5. Execution time of cryptographic operations.

Operation	T_H	T_{Mul}	T_{ECA}	T_{ASCON}
Time (ms)	0.0002	0.5371	0.0038	0.0002

In the proposed protocol, the vehicle requires $8T_H + 5T_{Mul} + 2T_{ASCON}$, while the RSU requires $6T_H + 8T_{Mul} + 3T_{ECA} + 2T_{ASCON}$. Between the operations, the hash value $h_{1,i}$ and verification term $R_i + h_{1,i}K_{pub}$ can be cached for an unchanged registered public key, while the static ECDH values x_iY_j and y_jX_i can be precomputed. These operations are included in the computational cost without applying any precomputation or caching reduction. Therefore, the vehicle and RSU require 2.6875 ms and 4.3098 ms, respectively, resulting in a total computational cost of 6.9973 ms.

As shown in Table 6, Li et al.'s protocol [10] has the lowest computational cost because it mainly relies on hash operations. However, as demonstrated in Section 5, Li et al.'s protocol [10] suffers from critical security weaknesses, including session-key disclosure, MitM, vehicle capture, and vehicle tracing attacks. Although Wei et al. [29] and Xie et al. [28] also achieve lower computational costs than the proposed protocol, Wei et al.'s protocol fails to ensure vehicle untraceability and resist impersonation attacks, while Xie et al.'s protocol is vulnerable to ephemeral secret leakage attacks. Moreover, the proposed protocol reduces the number of ECC multiplications for verifying n signatures from $4n$ in individual verification to $2n + 2$ in batch verification, providing computational benefits as the number of vehicles increases.

Table 6. Computational cost comparison.

Protocols	Vehicle	RSU	TA	Total Cost	RSU Batch, n Requests
Badshah et al. [26]	$9T_H + 8T_{Mul}$	$18T_H + 7T_{Mul}$	–	8.0619 ms	–
Wang et al. [27]	$6T_H + 8T_{Mul} + 6T_{ECA}$	$7T_H + 10T_{Mul} + 5T_{ECA}$	–	9.7122 ms	–
Xie et al. [28]	$5T_H + 5T_{Mul}$	$4T_H + 5T_{Mul}$	–	5.3728 ms	$4nT_H + (n+2)T_{Mul}$
Wei et al. [29]	$4T_H + 6T_{Mul} + T_{ECA}$	$5T_H + 7T_{Mul} + T_{ECA}$	–	6.9917 ms	–
Kong et al. [30]	$6T_H + 7T_{Mul}$	$7T_H + 9T_{Mul}$	–	8.5962 ms	–
Li et al. [10]	$7T_H$	$4T_H$	$13T_H$	0.0048 ms	–
Proposed	$8T_H + 5T_{Mul}$ $+2T_{ASCON}$	$6T_H + 8T_{Mul} + 3T_{ECA}$ $+2T_{ASCON}$	–	6.9973 ms	$(2n+2)T_{Mul}$

8.3. Communication Cost Comparison

We also compare the communication overhead of the proposed protocol with [10,26–30]. The communication cost is calculated by summing the bit-lengths of all messages transmitted over the public channel during the AKA phase. For a comparison, the bit-lengths of an identity, timestamp, hash output, random number, ECC point, signature, and ASCON authentication tag are set to 128, 32, 256, 128, 320, 128, and 128 bits, respectively. The length of the traffic-related message (m) is excluded from the comparison.

In the proposed protocol, V_i sends $\{C_1, B_i, E_i, CT_1, Tag_1, T_1\}$ to RSU_j, and RSU_j responds with $\{F_j, CT_2, Tag_2, T_2\}$.

- Msg_1: The message $\{C_1, B_i, E_i, CT_1, Tag_1, T_1\}$ requires $(256+320+320+256+128+32) = 1312$ bits.
- Msg_2: The message $\{F_j, CT_2, Tag_2, T_2\}$ requires $(320 + 128 + 128 + 32) = 608$ bits.

In total, the proposed protocol requires 1920 $(1312 + 608 = 1920)$ bits. Table 7 shows the total communication costs of the proposed protocol and related protocols [10,26–30]. The total communication overhead of the proposed protocol is lower than those in [10,27–29], but higher than those in [26,30]. Moreover, Li et al.'s protocol requires communication with the TA to obtain session key components, which introduces additional communication overhead.

Table 7. Communication cost comparison.

Protocols	**[10]**	**[26]**	**[27]**	**[28]**	**[29]**	**[30]**	**Proposed**
Total Cost (bits)	2912	1536	3232	1984	2336	1312	1920
Messages	6	2	3	2	2	2	2

8.4. NS-3/SUMO Simulation

To evaluate the practicality of the proposed protocol under realistic vehicular mobility, we conduct NS-3/SUMO-based simulations [15,16]. SUMO is used to generate vehicle mobility traces, and the generated `mobility.tcl` file is imported into NS-3. The SUMO traces are applied to moving vehicles, whereas RSUs remain stationary using ConstantPositionMobilityModel. Unlike safety-message-oriented simulations, this experiment does not generate BSM traffic. Instead, we only model the two authentication messages of the proposed AKA phase: the vehicle-to-RSU authentication request and the RSU-to-vehicle authentication response. The size of the revised first message $\{C_1, B_i, E_i, CT_1, Tag_1, T_1\}$ is 164 bytes, and the size of the revised second message $\{F_j, CT_2, Tag_2, T_2\}$ is 76 bytes, excluding m as in Section 8.3. The simulation environment generated by SUMO is shown in Figure 6 and the detailed simulation parameters are summarized in Table 8.

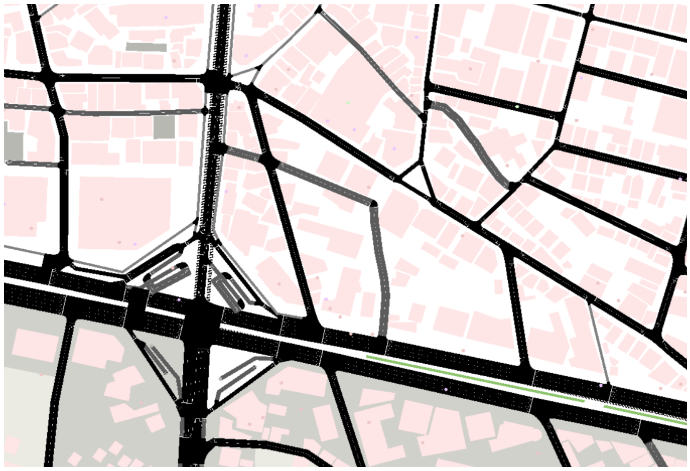


Figure 6. SUMO-based road map used for vehicular mobility generation.

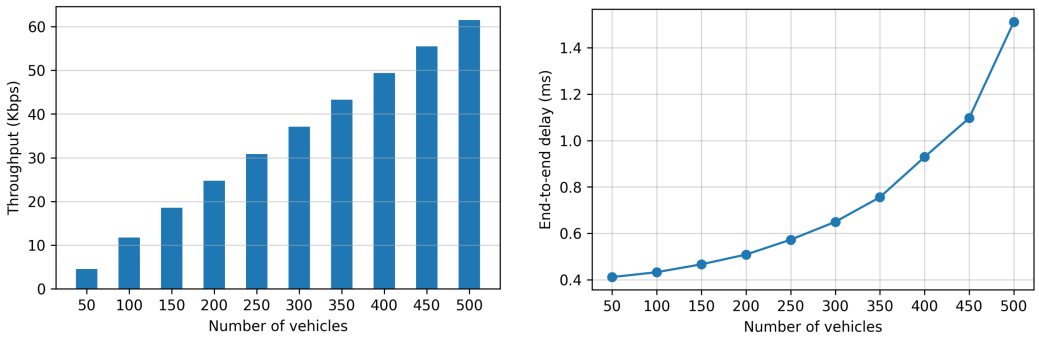
In the simulation, each vehicle periodically initiates the proposed AKA phase with the nearest RSU according to the SUMO-generated mobility trace. When a vehicle sends the first authentication message to the RSU, the RSU responds with the second authentication message after receiving the request. Therefore, the simulation reflects the communication overhead of the proposed AKA phase using only two exchanged messages. The throughput and end-to-end delay are computed as follows: throughput $= \frac{P_{rcv} \times P_{size} \times 8}{T}$; end-to-end delay $= \frac{1}{A_t} \sum_{i=1}^{A_t} (A^i_{rcv} - A^i_{snd})$, where P_{rcv}, P_{size}, T, A_t, A^i_{rcv}, and A^i_{snd} denote the number of received packets, packet size, total simulation time, numbers of received requests and

response packets, reception time of the ith authentication packet, and transmission time of the ith authentication packet, respectively.

Table 8. NS-3/SUMO simulation parameters.

Simulation Parameter	Details
Operating system	Ubuntu 22.04 LTS
NS-3 version	3.34
Mobility generator	SUMO-1.12.0
Mobility trace	`mobility.tcl`
Number of vehicles	50, 100, 150, ..., 500
Number of RSUs	5
Wireless scheme	IEEE 802.11p (WAVE)
Vehicle mobility	SUMO-generated mobility trace
RSU mobility	ConstantPositionMobilityModel
Simulation time	600 seconds
Selected area	500 m × 300 m
Vehicle speed	50 m/s
RSU positions/coverage radius	Uniformly distributed in simulation area/300 m

The simulation results are shown in Figure 7. As the number of vehicles increases, the amount of authentication traffic also increases because more vehicles participate in the AKA phase. Accordingly, the throughput increases with the number of vehicles. For example, the throughput is 4.50, 37.11, and 61.52 kbps for 50, 300, and 500 vehicles. The end-to-end delay also increases as the vehicle density increases, since more authentication messages are transmitted through the wireless channel. The corresponding end-to-end packet delays are 0.41, 0.65, and 1.51 ms, respectively. Nevertheless, the delay remains within a practical range, indicating that the proposed protocol can support efficient authentication under realistic vehicular mobility generated by SUMO.



(**a**) (**b**)

Figure 7. NS-3/SUMO simulation results: (**a**) throughput; (**b**) end-to-end packet delay.

9. Conclusions

This paper analyzed Li et al.'s anonymous and revocable authentication protocol for IoV environments and demonstrated that it suffers from several critical security weaknesses, including session-key disclosure attacks, MitM attacks, vehicle capture attacks, and vehicle untraceability. To address these problems, we proposed an ASCON-based pairing-free

certificateless AKA protocol for IoV. The proposed protocol employs ASCON to provide lightweight confidentiality and integrity protection and adopts ECC-based pairing-free certificateless authentication to mitigate TA overhead and the key escrow problem. In addition, the proposed protocol supports both individual and batch authentication, enabling RSUs to efficiently handle dynamically varying authentication requests from vehicles. The security analysis and performance evaluation demonstrated that the proposed protocol provides stronger security guarantees while maintaining practical computational and communication efficiency compared with existing IoV authentication protocols. Therefore, the proposed protocol offers a secure, efficient, and scalable authentication solution for practical IoV environments. In future work, we plan to extend the proposed protocol to support secure vehicle-to-vehicle (V2V) authentication within IoV environments.

Author Contributions: Conceptualization, M.K.; methodology, M.K. and T.K.; validation, T.K.; formal analysis, M.K. and T.K.; writing—original draft preparation, M.K.; writing—review and editing, T.K., D.K. and Y.P.; supervision, Y.P. and D.K.; project administration, Y.P.; funding acquisition, D.K. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by a National Research Foundation of Korea (NRF) grant funded by the Korea government (Ministry of Science and ICT) (RS-2024-00450915).

Data Availability Statement: The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding authors.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Yang, H.; Duan, D.; Liu, H.; Luo, C.; Wu, Y.; Li, W.; Zomaya, A.Y.; Song, L.; Xu, W. Scenario-adaptive key establishment scheme for LoRa-enabled IoV communications. *IEEE Trans. Mob. Comput.* **2024**, *23*, 12998–13014. [CrossRef]
2. Kwon, D.; Son, S.; Kim, M.; Lee, J.; Das, A.K.; Park, Y. A secure self-certified broadcast authentication protocol for intelligent transportation systems in UAV-assisted mobile edge computing environments. *IEEE Trans. Intell. Transp. Syst.* **2024**, *25*, 19004–19017. [CrossRef]
3. Prajapat, S.; Gautam, D.; Kumar, P.; Jangirala, S.; Das, A.K.; Park, Y.; Lorenz, P. Secure lattice-based aggregate signature scheme for vehicular ad hoc networks. *IEEE Trans. Veh. Technol.* **2024**, *73*, 12370–12384. [CrossRef]
4. Shao, H.; Huang, T.; Gao, G. A lightweight certificateless conditional privacy-preserving scheme for V2V communications. *IEEE Trans. Veh. Technol.* **2026**, *75*, 5846–5861. [CrossRef]
5. Dong, Y.; Yao, Y.; Chang, X.; Mišić, J.; Mišić, V.B. Lightweight certificateless authentication scheme with enhanced privacy for CAVs. *IEEE Internet Things J.* **2025**, *12*, 52644–52657. [CrossRef]
6. Khan, M.A.; Ullah, I.; Nisar, S.; Noor, F.; Qureshi, I.M.; Khanzada, F.U.; Amin, N.U. An efficient and provably secure certificateless key-encapsulated signcryption scheme for flying ad hoc network. *IEEE Access* **2020**, *8*, 36807–36828. [CrossRef]
7. Tedeschi, P.; Sciancalepore, S.; Eliyan, A.; Pietro, R.D. LiKe: Lightweight certificateless key agreement for secure IoT communications. *IEEE Internet Things J.* **2020**, *7*, 621–638. [CrossRef]
8. Kasyoka, P.; Kimwele, M.; Angolo, S.M. Certificateless pairing-free authentication scheme for wireless body area network in healthcare management system. *J. Med. Eng. Technol.* **2020**, *44*, 12–19. [CrossRef]
9. Manasrah, A.; Yaseen, Q.; Al-Aqrabi, H.; Liu, L. Identity-based authentication in VANETs: A review. *IEEE Trans. Intell. Transp. Syst.* **2025**, *26*, 4260–4282. [CrossRef]
10. Li, J.; Chen, J.; Liu, Y.; Liu, S.; Zhang, Y.; Zheng, Y. An anonymous and revocable authentication scheme with session key agreement for V2R communications. *IEEE Trans. Veh. Technol.* **2026**, *early access*. [CrossRef]
11. Canetti, R.; Goldreich, O.; Halevi, S. The random oracle methodology, revisited. *J. ACM* **2004**, *51*, 557–594. [CrossRef]
12. Burrows, M.; Abadi, M.; Needham, R. A logic of authentication. *ACM Trans. Comput. Syst.* **1990**, *8*, 18–36. [CrossRef]
13. Cremers, C.J. The Scyther tool: Verification, falsification, and analysis of security protocols: Tool paper. In Proceedings of the International Conference on Computer Aided Verification (CAV), Princeton, NJ, USA, 7–14 July 2008; pp. 414–418.
14. Cremers, C.J. Scyther Tool—Cas Cremers. Available online: https://people.cispa.io/cas.cremers/scyther (accessed on 21 September 2026).
15. Network Simulator 3. Available online: https://www.nsnam.org (accessed on 21 September 2026).

16. Behrisch, M.; Bieker, L.; Erdmann, J.; Krajzewicz, D. SUMO–Simulation of urban mobility: An overview. In Proceedings of the Third International Conference on Advances in System Simulation (SIMUL 2011), Barcelona, Spain, 23–29 October 2011.
17. Al-Riyami, S.S.; Paterson, K.G. Certificateless public key cryptography. In Proceedings of the International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT), Taipei, Taiwan, 30 November–4 December 2003; pp. 452–473.
18. Boneh, D.; Gentry, C.; Lynn, B.; Shacham, H. Aggregate and verifiably encrypted signatures from bilinear maps. In Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT), Warsaw, Poland, 4–8 May 2003; pp. 416–432.
19. Castro, R.; Dahab, R. Efficient certificateless signatures suitable for aggregation. Cryptology ePrint Archive. 2007. Available online: https://eprint.iacr.org/2007/454 (accessed on 20 October 2026).
20. Cui, J.; Zhang, J.; Zhong, H.; Shi, R.; Xu, Y. An efficient certificateless aggregate signature without pairings for vehicular ad hoc networks. *Inf. Sci.* **2018**, *451*, 1–15. [CrossRef]
21. Kamil, I.A.; Ogundoyin, S.O. An improved certificateless aggregate signature scheme without bilinear pairings for vehicular ad hoc networks. *J. Inf. Secur. Appl.* **2019**, *44*, 184–200. [CrossRef]
22. Zhao, Y.; Hou, Y.; Wang, L.; Kumari, S.; Khan, M.K.; Xiong, H. An efficient certificateless aggregate signature scheme for the Internet of Vehicles. *Trans. Emerg. Telecommun. Technol.* **2020**, *31*, e3708. [CrossRef]
23. Thumbur, G.; Rao, G.S.; Reddy, P.V.; Gayathri, N.B.; Reddy, D.V.R.K.; Padmavathamma, M. Efficient and secure certificateless aggregate signature-based authentication scheme for vehicular ad hoc networks. *J. Inf. Secur. Appl.* **2021**, *58*, 102780.
24. Yang, X.; Chen, A.; Wang, Z.; Du, X.; Wang, C. Cryptanalysis of an efficient and secure certificateless aggregate signature-based authentication scheme for vehicular ad hoc networks. *Secur. Privacy* **2022**, *5*, e194.
25. Ye, F.; Wu, W. A security-enhanced certificateless conditional privacy-preserving authentication scheme with collusion-resistance for IoV. *Comput. Netw.* **2025**, *259*, 111084. [CrossRef]
26. Badshah, A.; Waqas, M.; Muhammad, F.; Abbas, G.; Abbas, Z.H.; Chaudhry, S.A.; Chen, S. AAKE-BIVT: Anonymous authenticated key exchange scheme for blockchain-enabled Internet of Vehicles in smart transportation. *IEEE Trans. Intell. Transp. Syst.* **2023**, *24*, 1739–1755. [CrossRef]
27. Wang, Z.; Zhou, Y.; Qiao, Z.; Yang, B.; Gu, C.; Xu, Y.; Zhang, M. An anonymous and revocable authentication protocol for vehicle-to-vehicle communications. *IEEE Internet Things J.* **2023**, *10*, 5114–5127. [CrossRef]
28. Xie, Q.; Ding, Z.; Zheng, P. Provably secure and anonymous V2I and V2V authentication protocol for VANETs. *IEEE Trans. Intell. Transp. Syst.* **2023**, *24*, 7318–7327. [CrossRef]
29. Wei, L.; Zhang, Y.; Cui, J.; Zhong, H.; Bolodurina, I.; He, D. A threshold-based full-decentralized authentication and key agreement scheme for VANETs powered by consortium blockchain. *IEEE Trans. Mob. Comput.* **2024**, *23*, 12505–12521. [CrossRef]
30. Kong, Y.; Tian, J. An ECC-based anonymous and fast handover authentication protocol for Internet of vehicles. *Appl. Sci.* **2025**, *15*, 5894. [CrossRef]
31. Yu, G.; Li, Q.; Mao, H.; El-Latif, A.A.A.; Rodrigues, J.J.P.C. A multi-scenario authenticated key exchange scheme with forward secrecy for fog-enabled VANETs. *IEEE Trans. Veh. Technol.* **2025**, *74*, 831–846. [CrossRef]
32. Yang, W.; Cao, P.; Zhang, F. A secure pairing-free certificateless online/offline signcryption scheme with batch verification for edge computing-based VANETs. *IEEE Trans. Veh. Technol.* **2025**, *74*, 1570–1583. [CrossRef]
33. Dolev, D.; Yao, A. On the security of public key protocols. *IEEE Trans. Inf. Theory* **1983**, *29*, 198–208. [CrossRef]
34. Canetti, R.; Krawczyk, H. Universally composable notions of key exchange and secure channels. In Proceedings of the International conference on the Theory and Applications of Cryptographic Techniques (EUROCRYPT), Amsterdam, The Netherlands, 28 April–2 May 2002; pp. 337–351.
35. Dobraunig, C.; Eichlseder, M.; Mendel, F.; Schläffer, M. Ascon v1.2: Lightweight authenticated encryption and hashing. *J. Cryptol.* **2021**, *34*, 33. [CrossRef]
36. Jang, H.; Kwon, D.; Park, Y. Ascon-based lightweight and robust authentication scheme for IoT-enabled healthcare system. *IEEE Internet Things J.* **2026**, *13*, 16980–16994. [CrossRef]
37. Chim, T.W.; Yiu, S.M.; Hui, L.C.K.; Li, V.O.K. OPQ: OT-based private querying in VANETs. *IEEE Trans. Intell. Transp. Syst.* **2011**, *12*, 1413–1422. [CrossRef]
38. Tan, H.; Gu, S.; Zhu, H.; Shen, J.; Bhuiyan, Z.A.; Wu, Q.J. An efficient ASCON-based group authentication and key agreement scheme with non-linkability and integrity assurance for IIoTs. *IEEE Trans. Dependable Secure Comput.* **2026**, *23*, 8032–8047. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.