3

Lightweight Authentication and Verification using Tamper-Proof Devices

IoV connects vehicular networks with the broader IoT, creating a dynamic environment where vehicles communicate with each other, infrastructure, pedestrians, and other devices[44]. This interconnected system improves transportation functionality and intelligence but also introduces significant security vulnerabilities and performance challenges[177]. The main challenge lies in balancing security, privacy, and efficiency, each of which presents unique and complex issues requiring efficient and secure solutions. Security in IoV is critical due to the open nature of vehicular networks[14]. Modern vehicles exchange substantial amounts of sensitive data, including real-time GPS locations, driving behaviors, personal identifiers, and vehicle diagnostics. This data encompasses detailed information such as speed, acceleration, driver identification, bio-metric data, and interactions with infotainment systems[16]. The collection and exchange of this data heighten the risk of cyberattacks. Malicious entities can intercept, manipulate, or disrupt communications, potentially causing traffic accidents, unauthorized tracking, or data breaches[15]. Similarly, privacy concerns in IoV are also significant, as vehicles and their users generate extensive data that can reveal personal information. The challenge lies in preserving this privacy while enabling the necessary flow of information for operational efficiency and safety. The privacy preservation methods must be carefully balanced to prevent unauthorized access while allowing legitimate information exchange critical for applications like emergency response and traffic management[51]. Further, efficiency is essential in IoV network due to the real-time nature of vehicular communication and the large data volume. The network must support rapid message dissemination and processing for applications like collision avoidance, traffic flow optimization, and autonomous driving[103]. However, the computational and communication overhead required to maintain high levels of security and privacy can significantly affect the overall performance. Ensuring efficient use of bandwidth, minimizing latency, and optimizing computational resources are essential to maintain seamless and responsive IoV operations[134]. This chapter discusses advanced security solutions designed to address these critical aspects of IoV communication: conditional privacy preservation based authentication, static batch verification, and dynamic batch verification. Each technique offers a unique approach to harmonizing the demands of security, privacy, and efficiency.

The first scheme[42] introduces a balance between privacy and authentication through conditional privacy preservation. It secures users' sensitive information while maintaining the necessary level of authentication for vehicular communication. By employing pseudonymity and MAC functions, the scheme protects user identities from malicious entities without compromising system integrity. In the IoV context, the importance of privacy is paramount, as vehicles constantly exchange sensitive data, including location, speed, and driving behavior[52]. If this information were to be intercepted, it could lead to severe privacy breaches

and potentially harmful consequences. Therefore, the scheme focuses on creating a secure environment where privacy is conditionally preserved based on the context and necessity of the exchanged information. Addressing the trade-off between efficiency and security, the second and third schemes focus on batch verification mechanisms. In the context of IoV, where the volume of data and communication between vehicles and infrastructure is large, ensuring that each message is verified individually can lead to significant delays and inefficiencies. Therefore, the second scheme presents static batch verification[55] , which optimizes the verification process by allowing multiple messages to be verified simultaneously. This method significantly reduces the computational overhead using lightweight cryptographic operations (i.e., one-way hash functions[71], XOR operations, etc.), enhancing the overall efficiency of the network. Static batch verification is particularly useful in scenarios where the communication patterns and message flows are predictable and relatively stable. By processing multiple messages in batches, the system can efficiently handle a higher volume of data without compromising on security. The third technique advances this concept further with dynamic batch verification[12]. This method not only incorporates the principles of static batch verification but also introduces message prioritization based on the criticality of the messages. In an IoV environment, certain messages, such as emergency alerts or critical system updates, require immediate attention and quick verification. Dynamic batch verification addresses this need by dynamically adjusting the verification process to prioritize high-priority messages. This ensures that urgent communications are processed and authenticated promptly, thereby enhancing the system's responsiveness and reliability[32]. By combining batch verification with dynamic prioritization, this technique provides a robust solution to manage the vast and varying data traffic within IoV networks. Collectively, these techniques form a comprehensive framework that addresses the intricate demands of security, privacy, and efficiency in the IoV. The following sections discuss the specifics of each technique, providing a detailed analysis of their implementation and potential impact on the future of ITS.

3.1 CONDITIONAL PRIVACY PRESERVATION AND LIGHTWEIGHT AUTHENTICATION

Traditional authentication methods[178, 47, 134, 48] often rely on fixed cryptographic keys, which are vulnerable to various attacks if compromised. Additionally, maintaining user privacy during the authentication process is crucial to prevent tracking by malicious entities. Therefore, balancing robust authentication and privacy preservation in IoV requires an advanced approach that address both aspects without compromising security and usability. Despite advancements in IoV security protocols, achieving traceability and revocation remains challenging due to the lack of conditional privacy preservation mechanisms. Traceability ensures that malicious activities can be traced back to their source, crucial for accountability and deterrence of illicit behavior[179]. Existing privacy-preserving authentication schemes[48, 113, 102, 116, 118, 119, 120] often sacrifice traceability to protect user anonymity, creating a gap in identifying and addressing malicious activities. Similarly, the absence of conditional privacy preservation hinders the effective revocation of malicious entities, making it difficult to manage authentication credentials dynamically based on context or behavior. This gap poses a substantial risk to IoV security framework. Without the ability to trace malicious activities or revoke compromised credentials effectively, the network remains vulnerable to persistent threats.

Further, the key disclosure problem, where the compromise of cryptographic keys leads to widespread vulnerabilities, is particularly critical aspect in IoV[65]. The static nature of existing privacy-preserving

schemes means that once a key is compromised, it can be used maliciously without the possibility of dynamic update or revocation. Attackers can intercept, manipulate, or forge messages, potentially causing traffic accidents or gaining unauthorized access to vehicle systems. This undermines not only the security of individual vehicles but also the integrity of the entire IoV network[110]. To address these challenges, the

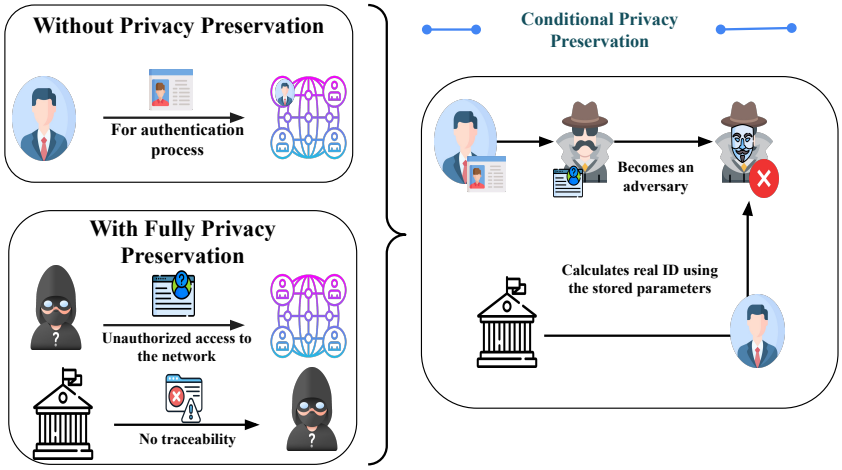


Figure 3.1: The impact of conditional privacy preservation over fully privacy preservation

proposed scheme introduces a Novel MAC-based Authentication Scheme (NoMAS)[42] that incorporates conditional privacy preservation along with HKU and SKU. The mechanism employs a MAC[70] to ensure the integrity and authenticity of messages exchanged within the IoV network. MACs provide a robust method for verifying the authenticity of messages by detecting any modification done during transmission. By integrating HKU and SKU with conditional privacy preservation, the proposed scheme offers a comprehensive solution to the key disclosure problem and enhances the overall security framework of IoV. The concept of conditional privacy preservation is illustrated in Figure 3.1, which examines three privacy protection scenarios in a vehicle network: no privacy preservation, full privacy preservation, and conditional privacy preservation. Without privacy protection, data including location and speed is transmitted directly to the network, exposing sensitive information to potential tracking and unauthorized access. In the case of full privacy preservation, users employ pseudonyms or anonymization techniques to hide their real identity, allowing data transmission without revealing any identifiable information. However, this method poses a risk: if an unauthorized entity gains access to the network, legal authorities cannot trace the malicious user back to their real identity if needed for legal or security purposes[51]. Conversely, conditional privacy preservation involves the use of pseudonyms, but allows the system to reveal the real identity under specific conditions, such as a security breach or legal requirement. This approach seeks to balance privacy and security. The significant contributions of NoMAS in the field of IoV can be compiled as follows:

Lightweight authentication and message verification with conditional privacy preservation: NoMAS introduces a MAC[70] based lightweight yet robust authentication process tailored for IoV environments. By utilizing only five hash functions[71] and an XOR operation for each message, the scheme ensures effective message verification with minimal computational overhead. NoMAS also incorporates conditional privacy preservation using by calculating pseudo-identity to hide the real identity of the user and dynamically adjusting privacy levels based on contextual information.

Secure key update mechanisms (HKU and SKU): A significant contribution of NoMAS is the solution to the key disclosure problem through HKU and SKU. HKU involves the complete replacement of cryptographic keys using the CRT[180], ensuring that any disclosed key is rendered obsolete. SKU, on the other hand, entails periodic modifications to existing keys, balancing security and computational efficiency. This dual approach guarantees continuous key integrity, even if a key is compromised, by dynamically updating or replacing keys.

Optimized communication overhead: NoMAS achieves significant reductions in communication overhead, requiring only 44 bytes for authentication processes, including 20 bytes for two MAC functions and 4 bytes for timestamps. This optimized overhead ensures efficient data transmission, crucial for maintaining the high-speed communication standards required in IoV networks.

Theoretical and formal security proofs: In the security analysis, NoMAS is validated using BAN logic[78], demonstrating that it meets all security demands theoretically. Furthermore, the scheme is subjected to software-based formal security proofs using the ProVerif tool[89] and the concept of Pi calculus. These rigorous validations confirm the robustness of NoMAS against potential security threats.

Performance analysis and simulation: The performance of NoMAS is rigorously analyzed and simulated using the Simulation of Urban Mobility (SUMO) and Network Simulator-2 (NS-2) version 2.35[88]. Key performance metrics, such as computation and communication overhead along with the One-Way Delay (OWD), Packet Delivery Ratio (PDR), Packet Loss Ratio (PLR), and network throughput, are evaluated and discussed. The scheme exhibits a maximum improvement of 99.60% in computation cost and 81.03% in communication overhead, along with significant reductions in energy consumption.

One of the key strengths of the NoMAS scheme is its compatibility with heterogeneous vehicular networks where both legacy and modern systems coexist. Legacy vehicles, often equipped with DSRC (IEEE 802.11p) communication modules and limited computational resources, may not support advanced security capabilities such as public key cryptography, biometric authentication, or dynamic identity management. In contrast, modern vehicles leverage advanced technologies like C-V2X and 5G-V2X, and often include TPDs, edge computing support, and multi-user interaction capabilities. NoMAS is designed to bridge this technological gap by relying exclusively on lightweight symmetric cryptographic operations—specifically hash functions, HMACs, and XOR—which are supported even on low-power OBUs. Furthermore, its stateless message authentication model based on pre-distributed group keys (Key_{gp}) avoids the need for continuous communication with centralized authorities, allowing both legacy and modern nodes to authenticate messages efficiently. The scheme's fixed 44-byte message size also ensures minimal bandwidth usage, supporting uniform communication across devices with varying protocol stacks and capabilities. As a result, NoMAS can operate seamlessly across a diverse vehicular ecosystem, enabling secure and real-time authentication in mixed deployment scenarios such as smart cities, highways, and public transportation systems.

3.1.1 Network and Adversary Model

In NoMAS, the network model is designed using three primary components: OBU in vehicles, RSCN, and a Registration Authority (RA). These components, along with other distinctive elements of the IoV network, ensure secure and efficient communication. Each vehicle is equipped with an OBU that manages the transmission and reception of data, ensuring that messages are authenticated and verified for integrity. Additionally, OBU are equipped with TPD[91] that play a critical role in executing the proposed MAC-based

authentication protocol as part of the NoMAS scheme. TPD protect the integrity and confidentiality of stored cryptographic keys and ensure that even if an attacker gains physical access to the OBU, they cannot extract sensitive information or compromise the security of the system[94]. By incorporating tamper-proof hardware, the OBU enhances the overall security framework of the NoMAS scheme, providing robust protection against physical tampering and unauthorized access. RSCN serve as intermediaries that bridge the communication between vehicles and the broader IoV infrastructure. These nodes are strategically placed along roadways to provide a network of connectivity points[44]. RSCN facilitate the relay of information from vehicles to the RA and other central components like cloud servers. They are also involved in the dissemination of system updates and key management tasks. The interaction between OBU and RSCN is important for maintaining real-time data exchange and network reliability. Further, RA is a critical entity within the NoMAS network model, having substantial storage and computational capabilities. Its responsibilities include user registration, system key updates, and overall network security management. The RA is the only entity with the authority and capability to trace the original identity of a vehicle in cases of malicious activity. For instance, if a user intentionally sends bogus messages, the RA can track the user's original identity and take necessary actions such as blocking the user from the network. Additionally, the RA performs HKU in scenarios involving unauthorized access to system keys by attackers or legitimate access by law enforcement agencies[29]. This capability ensures that the security of the network remains intact even under potential threats or legal inquiry. The RA's role in managing keys and tracing malicious activities is fundamental to the robustness of the NoMAS scheme. The network model depicted in Figure 3.2 illustrates

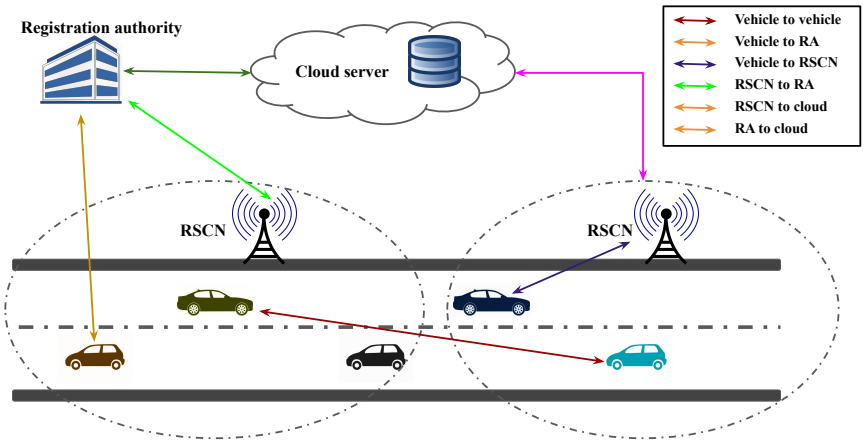


Figure 3.2: Network model for NoMAS

the comprehensive interaction framework for the NoMAS scheme that provides different types of the communications such as: V2V for sharing real-time information such as traffic conditions, accidents, and other critical updates, Vehicle-to-RSCNs (V2R) for updating the vehicle's status, receiving system updates, and participating in broader IoV functions, Vehicle-to-RA for registration and key management tasks, RSCN-to-RA to update system status, report anomalies, and receive instructions, RSCN-to-Cloud to share aggregated traffic information, system updates, and other analytical data necessary for the efficient functioning of IoV services, and RA-to-Cloud for storage and advanced computational tasks. In the context of NoMAS, an unpredictable adversary, capable of intruding into a provably secure communication system. is considered. This adversary model is based on the widely accepted Canetti-Krawczyk (CK)-adversary model[181]. This model assumes an adversary, denoted as $\mathbb{A}$, who possesses extensive capabilities to compromise the security

and integrity of the IoV network. Further, in Table 3.1, potential threats posed by $\mathbb{A}$ are detailed, illustrating the various attack vectors and their implications.

Table 3.1: Adversary model: Detailed representation of attack vectors, descriptions, and implications

Attack vector	Description	Implications
Eavesdropping	Interception of transmitted data without altering the communication flow.	Gathers sensitive information for further attacks.
Compromising RSCNs	Gaining control over RSCNs to manipulate or disrupt communications.	Alters relayed information, injects false data, blocks legitimate messages.
Physical attacks on OBU	Physically stealing OBU to extract stored sensitive information.	Compromises cryptographic keys and other security parameters.
Brute-force and key-guessing	Systematically attempting all possible key combinations to reveal cryptographic keys.	Decrypts messages, fabricates valid messages, impersonates legitimate participants.
Replay attacks	Capturing and re-transmitting valid messages to create network congestion or simulate activities.	Leads to traffic disruptions, resource exhaustion, and misleading network participants.
Message alteration and injection	Modifying intercepted messages or creating fraudulent messages to broadcast.	Misleads vehicles, causes incorrect actions, compromises traffic safety and efficiency.
Repudiation attacks	Denying receipt of messages or participation in communications.	Causes confusion and misjudgments, undermines trust and reliability.
Identity disclosure and spoofing	Disclosing authenticated users' identities and impersonating them.	Gains unauthorized access, issues false commands, diverts data.
Tracking and location privacy	Collecting broadcast messages to trace the vehicle's path over time.	Enables targeted surveillance, unauthorized profiling, impacts user privacy and safety.
Side-channel attacks on TPD	Extracting secret parameters from TPD using physical emanations.	Leads to information leakage and broader security breaches.
System key disclosure	Disclosing the system key through brute-force, key guessing, or exploiting vulnerabilities.	Decrypts, modifies, or fabricates messages, compromising network integrity and security.

By considering these comprehensive attack vectors, NoMAS is designed to address and mitigate the wide range of threats posed by a CK-adversary. The integration of advanced cryptographic techniques (i.e., MAC, one-way hash functions, XOR operations), dynamic key management (i.e., SKU, and SKU), and conditional privacy preservation (i.e., pseudonymity) mechanisms ensures that the IoV network remains secure and resilient against $\mathbb{A}$.

3.1.2 Proposed Scheme: NoMAS

NoMAS operates through several distinct stages: initialization stage, vehicle registration stage, authentication stage, message verification stage, password and key update stage. Each stage plays a critical role in maintaining the integrity and security of the communication within the IoV network. Before discussing the specifics of the proposed scheme, it is essential to establish the foundational assumptions upon which NoMAS is built:

- A secure channel exists between a user and the RA for the initial setup. This channel ensures that sensitive information, such as cryptographic keys and personal details, can be transmitted securely without the risk of interception or tampering.

- A secure channel is also established between OBU and the RA. This channel is crucial for securely transmitting system updates, key management information, and authentication data.

During the initialization stage of the NoMAS scheme, RA generates and calculates all necessary cryptographic parameters using its secure keys, forming the backbone of the scheme with robust cryptographic techniques, including the creation of private and public keys for signing and encryption, as well as system and group keys. In the subsequent vehicle registration stage, users register offline with the RA, providing personal details such as Vehicle Identification (VID), password, license ID, and contact information, ensuring secure transmission of initial setup data. The RA then configures TPD with the user's information, issuing these devices to users to securely store cryptographic keys and personal data. Authentication stage requires users to sign in with their credentials, which are verified against the TPD-stored data using MACs to ensure message integrity and authenticity, granting network access only to legitimate users. In the message verification stage, MACs are used to verify the integrity and authenticity of all messages within the IoV network by hashing message content with a shared secret key, detecting any tampering immediately. The password update stage allows users to independently update their passwords via TPDs, which securely encrypt and store new credentials, enhancing user autonomy and reducing the risk of compromised passwords. Addressing system key disclosure, the key update stage employs HKU and SKU mechanisms. Table 3.2 describes the notations used in the NoMAS scheme and their explanations, providing a clear reference for understanding the various components and operations within the scheme.

Table 3.2: List of notations and their explanations

Notation	**Explanation**
Pri_{sign} and Pri_E	Private keys for signature and encryption respectively.
Pub_{sign}	Public key for signature
a and b	Prime numbers
ck_i	Confidential key
$\alpha_i, \beta_i, \gamma$	Calculating parameters
$InPad$ and $OutPad$	Inner and outer padding for HMAC function
Key_{gp}	Group key
UID_i and VID_i	User identity and vehicle's identity of vehicle V_i
$PASS_i$	Multi-model Biometric password
$SPASS_i$	Calculated secret password
η_i, δ_i	Randomly selected numbers for authentication process
G_i	Encrypted group key
M	Sent message
$PPVID_i$	Primary pseudo identity
$sign_i$	Generated signature for i^{th} message
$MAC_{Key_{gp}}$	Message authentication code using the group key
$\oplus$	XOR operation
$\|\|$	Concatenation operation
$h()$	Hash function

Initialization stage: In this first stage, RA initializes the required parameters for the further stages of the proposed NoMAS. Network initialization of IoV is explained in the Algorithm 1. Firstly, RA generates private key Pri_{sign} for the sign function and Pri_E for the encryption function. Then calculates the public key Pub_{sign} for the sign function. After this, RA chooses confidential key ck_i for each vehicle V_i, where $i = 1,2,3 \ldots N$. For further calculations, RA computes α_i, β_i and γ_i using that ck_i. After all the calculations, RA now generates inner and outer padding of HMAC function $InPad$ and $OutPad$ and a group key Key_{gp}. Using these values, RA calculates λ_1 and λ_2. At last, RA publishes Pub_{sign} in the network, and this resultant information $\langle Key_{gp}, \lambda_1, \lambda_2, \tau_1 \rangle$ are stored in a table for further calculations, where τ_1 is the current timestamp. The whole process of parameter initialization is illustrated in the Figure 3.3a.

Algorithm 1 Network initialization process for secure communication

Input: Number of nodes N
Output: Pub_{sign}, $InPad$, $OutPad$, Key_{gp}, and stored tuple $\langle Key_{gp}, \lambda_1, \lambda_2, \tau_1 \rangle$
1: RA generates Pri_{sign}, Pub_{sign}, and Pri_E.
2: **for** $i = 1$ to N **do**
3: RA chooses ck_i.
4: Compute $\alpha_i = \left\lceil \frac{\prod_{j=1}^{N} ck_j}{ck_i} \right\rceil$.
5: Compute β_i such that $\alpha_i \times \beta_i \equiv 1 \pmod{ck_i}$.
6: Compute γ as $\gamma = \sum_{i=1}^{N} \alpha_i \times \beta_i$.
7: **end for**
8: RA generates $InPad$, $OutPad$, and Key_{gp}.
9: Calculate $\lambda_1 = Key_{gp} \oplus OutPad$.
10: Also calculate $\lambda_2 = Key_{gp} \oplus InPad$.
11: RA publishes Pub_{sign} in the network and stores $\langle Key_{gp}, \lambda_1, \lambda_2, \tau_1 \rangle$.

Vehicle registration stage: After the initialization, all the users need to be registered themselves to the RA using the offline procedure. The process of vehicle registration is discussed in the Algorithm 2. Firstly, as shown in the Figure 3.3b, the user of a particular vehicle V_i submits his own identity UID_i, his vehicle's identity VID_i, and personal information to RA through a secure medium. Then, the user provides his fingerprints as the password $PASS_i$ to the OBU. Related OBU randomly chooses a 160-bit number η_i and then calculates the secret password $SPASS_i$ using the original password and η_i. Calculated $SPASS_i$ and η_i are then sent to the RA by the secure medium. When RA receives $SPASS_i \oplus \eta_i$ and VID_i along with the personal information, it randomly selects 160-bit number δ_i and computes pseudo ID $PVID_i$ and ε_i^1 then these calculated values are used for the encryption process of group key Key_{gp}. RA sends the following information to the OBU by a secure medium:$\langle Pub_{sign}, Pri_E, G_i, PVID_i, VID_i, ID_{RA} \rangle$ and $\langle \varepsilon_i^1, ck_i, InPad, OutPad \rangle$. When related OBU receives the data, it calculates $\xi_i, \varepsilon_i^2, VID_i^{'}, PVID_i^{'}, \varepsilon_i^{1^{'}}, ck_i^{'}, Pri_E^{'}, \varepsilon_i^3$, and $\varepsilon_i^{3'}$. Further, OBU updates the data in its storage table as follows: $\langle VID_i^{'}, PVID_i^{'}, ID_{RA}, G_i \rangle$, $\langle Pri_E^{'}, ck_i^{'}, \varepsilon_i^{1^{'}}, \varepsilon_i^{3'}, \xi_i, \rangle$ and $\langle InPad, OutPad \rangle$.

Authentication stage: When a user tries to log in to TPD by putting his password $PASS_i^*$, the user's authentication process is performed. Figure 3.3c shows the authentication process of a user. For this purpose, TPD calculates $VID_i^*, \eta_i^*, SPASS_i^*, ck_i^*, \varepsilon_i^{1*}, Pri_E^*, PVID_i^*, \varepsilon_i^{3*}, \varepsilon_i^{2*}$, and ε_i^{3**}. To validate the legitimacy of the user, TPD checks $\varepsilon_i^{3*} == \varepsilon_i^{3**}$. If they match, the user successfully logs in to the TPD, then stores the current login timestamp τ_{login} and the $PVID_i^*$ in the table that RA uses to track the user in case of any discrepancy. At last, TPD successfully gets the group key Key_{gp}, λ_1, and λ_2 using the reverse calculations shown in Algorithm 3.

Message verification stage: To achieve better performance and fast processing of message signing and verification, NoMAS uses HMAC function that has much less computation cost . When a user logs into the TPD, the vehicle can communicate with other components of IoV until the user logs out from the TPD. For secure communication, the message has to be signed with a mechanism, and then that signature should be verified at the receiver end. The process of secure communication is described in the Algorithm 4. Whenever OBU wants to communicate with other components, it generates the message M. Now, TPD calculates primary pseudo ID $PPVID_i$ using the current timestamp τ_M and generates a signature $sign_i$ for the message M as:

$$\begin{aligned} sign_i &= h(\lambda_1 || h(\lambda_2 || M || \tau_M || PPVID_i)), \\ &= MAC_{Key_{gp}}(M || \tau_M || PPVID_i). \end{aligned} \tag{3.1}$$

Algorithm 2 Vehicle registration process done by RA including pseudo-identity generation

Input: Number of vehicles N, UID_i, VID_i, $PASS_i$
Output: G_i, Pub_{sign}, Pri_E, and other security parameters updated in the OBU storage

for $i = 1$ to N **do**
  User of V_i submits UID_i, and VID_i to RA.
  Also provides $PASS_i$ to OBU.
  OBU calculates $SPASS_i = h(PASS_i \| \eta_i)$.
  OBU sends $SPASS_i$ and η_i to RA.
  RA selects δ_i and calculates $PVID_i = h(VID_i \| \delta_i)$.
  Also calculates $\varepsilon_i^1 = h(VID_i \| \delta_i \| PVID_i \| ID_{RA})$.
  Encryption of Key_{gp}: $G_i = Key_{gp} \oplus ck_i \oplus \varepsilon_i^1 \oplus SPASS_i \oplus \eta_i$.
  RA sends $\{Pub_{sign}, Pri_E, G_i, PVID_i, VID_i, ID_{RA}\}$ and $\{\varepsilon_i^1, ck_i, InPad, OutPad\}$ to OBU.
  OBU calculates $\xi_i = h(PASS_i \| VID_i) \oplus \eta_i$,
  $\varepsilon_i^2 = h(VID_i \| ck_i \| SPASS_i \| ID_{RA})$,
  $VID_i^{'} = VID_i \oplus h(PASS_i)$,
  $PVID_i^{'} = PVID_i \oplus SPASS_i$,
  $\varepsilon_i^{1'} = \varepsilon_i^1 \oplus SPASS_i$,
  $ck_i^{'} = ck_i \oplus SPASS_i$,
  $Pri_E^{'} = Pri_E \oplus SPASS_i$,
  $\varepsilon_i^3 = h(\varepsilon_i^1 \| \varepsilon_i^2)$,
  $\varepsilon_i^{3'} = \varepsilon_i^3 \oplus SPASS_i$,
  OBU updates the calculated data in its storage table.
end for

Algorithm 4 Message verification to maintain the integrity of NoMAS scheme

Input: Number of vehicles N, Key_{gp},λ_1, λ_2, τ_M
Output: Verified message acceptance status for each vehicle

for $i = 1$ to N **do**
  Generate message M.
  TPD calculates $PPVID_i = h(\tau_M \| Key_{gp}) \oplus PVID_i$.
  TPD generates $sign_i$ for the message M.
  OBU broadcasts message $\{M, sign_i, PPVID_i, \tau_M\}$.
  After receiving the broadcast message, the vehicle checks its timestamp.
  TPD calculates:

$$sign_i^* = h(\lambda_1 \| h(\lambda_2 \| M \| \tau_M \| PPVID_i)) = \text{MAC}_{Key_{gp}}(M \| \tau_M \| PPVID_i).$$

  if $sign_i = sign_i^*$ **then**
    The vehicle accepts the broadcast message.
  end if
end for

Here, λ_1 and λ_2 are used to generate the HMAC function. Now, OBU broadcasts the message having $\{M, sign_i, PPVID_i, \tau_M\}$ in the IoV network. Whenever another OBU/ RSCN or other components of IoV receives that broadcast message, they first check the freshness of the message using the timestamp τ_M. After that, the TPD of the concerning component calculates $sign_i^*$ as shown in the Figure 3.3d and verifies whether $sign_i == sign_i^*$. If yes, then the concerned component accepts the broadcast message; otherwise discards it.

Password and key update stage: When a new user wants to access the TPD, the only way to update his password is that the initially authenticated user can request the OBU. Let's say; the authenticated user is $U_{i,a}$ and new user $U_{i,n}$ wants to update the password, then the following steps, shown in the Algorithm 5,

Algorithm 3 Privacy-preserving authentication process for NoMAS scheme

Input: Number of vehicles N, $PASS_i^*$
Output: Key_{gp}, λ_1 and λ_2, Updated parameters in the TPD storage

for $i = 1$ to N **do**
  User provides the password $PASS_i^*$ to TPD.
  TPD calculates $VID_i^* = VID_i^{'} \oplus h(PASS_i^*)$,
  $\eta_i^* = \xi_i \oplus h(PASS_i^* \| VID_i^*)$,
  $SPASS_i^* = h(PASS_i^* \| \eta_i^*)$,
  $ck_i^* = ck_i^{'} \oplus SPASS_i^*$,
  $\varepsilon_i^{1*} = \varepsilon_i^{1^{'}} \oplus SPASS_i^*$,
  $Pri_E^* = Pri_E^{'} \oplus SPASS_i^*$,
  $PVID_i^* = PVID_i^{'} \oplus SPASS_i^*$,
  $\varepsilon_i^{3*} = \varepsilon_i^{3^{'}} \oplus SPASS_i^*$,
  $\varepsilon_i^{2*} = h(VID_i^* \| ck_i^* \| SPASS_i^* \| ID_{RA})$,
  $\varepsilon_i^{3**} = h(\varepsilon_i^{1*} \| \varepsilon_i^{2*})$,
  if $\varepsilon_i^{3*} = \varepsilon_i^{3**}$ **then**
    Successfully log in to the TPD.
  end if
  TPD updates τ_{login} and $PVID_i^*$ in the table.
  TPD calculates $Key_{gp} = G_i \oplus ck_i^* \oplus \varepsilon_i^{1*} \oplus SPASS_i^* \oplus \eta_i^*$,
  $\lambda_1 = Key_{gp} \oplus OutPad$,
  $\lambda_2 = Key_{gp} \oplus InPad$.
end for

are performed. $U_{i,a}$ logs into the TPD using his password $PASS_{i,a}$. After login, TPD authenticates the user $U_{i,a}$. If the user is valid, then OBU requests to enter the password related to the new user $U_{i,n}$ as $PASS_{i,n}$. When TPD gets the new password, it calculates $SPASS_{i,n}, \xi_{i,n}, \varepsilon_{i,n}^2, VID_{i,n}^{'}, PVID_{i,n}^{'}, \varepsilon_{i,n}^{1^{'}}, ck_{i,n}^{'}, Pri_{E,n}^{'}, \varepsilon_{i,n}^3, \varepsilon_{i,n}^{3'}$, and $G_{i,n}$. At last, TPD updates already stored values with the newly calculated values.

NoMAS performs two types of key updates to enhance the security and privacy of the IoV network. SKU is done periodically, and HKU is done only in case of the revelation of the keys by the adversary.

1. **Soft Key Update:** The process of soft key update is described in the following steps:

   - Initially, RA creates new group key Key_{gp}^{new} and private key Pri_E^{new}. RA then computes $\lambda_1^{new} = Key_{gp}^{new} \oplus OutPad$ and $\lambda_2^{new} = Key_{gp}^{new} \oplus InPad$ and stores all the new parameters in the table. RA performs encryption and signing of these values such as

   $$\sigma = Sign_{Pri_{sign}}(Enc_{Pri_E^{old}}(Key_{gp}^{new} || Pri_E^{new} || \tau_2)), \tag{3.2}$$

   Here, τ_2 is the current timestamp. Now, RA sends the encrypted message along with the signature σ to the OBU.
   - When a vehicle receives σ, Attached TPD validates the signature using the public key Pub_{sign} and decrypts the message as $Dec_{Pri_E^{old}}(Key_{gp}^{new} \| Pri_E^{new} \| \tau_2)$. After this TPD calculates $G_i^{new} = G_i^{old} \oplus Key_{gp}^{old} \oplus Key_{gp}^{new}$, and $Pri_E{}'^{new} = Pri_E{}'^{old} \oplus Pri_E^{old} \oplus Pri_E^{new}$.

2. **Hard Key Update:** If a malicious user anyhow accesses the private key for encryption, SKU fails despite having low communication cost because using this private key, and he may access all the data. In such a scenario, HKU is performed using the following steps:

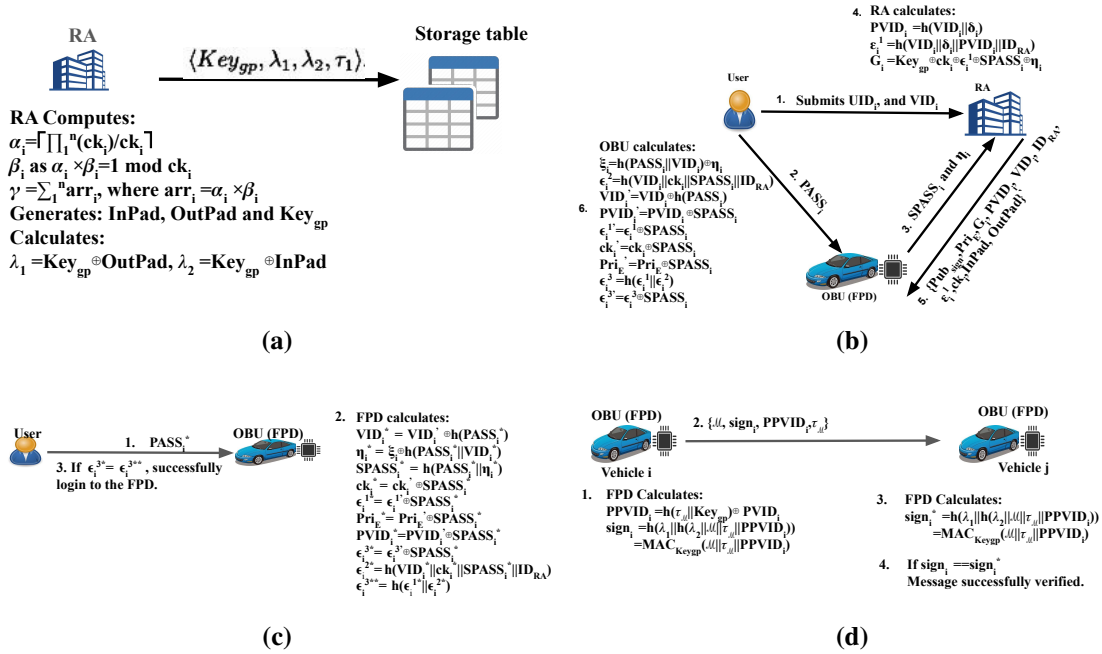


Figure 3.3: Various phases of the NoMAS scheme: (a) Network initialization and key generation stage, (b) User and vehicle registration stage, (c) Anonymous authentication of the user, (d) Secure communication and message verification between two vehicles

- Initially, a new group key Key_{gp}^{new} and a private key Pri_E^{new} are generated by RA. After that, RA computes $\lambda_1^{new} = Key_{gp}^{new} \oplus OutPad, \lambda_2^{new} = Key_{gp}^{new} \oplus InPad, Q_1 = \gamma \times Key_{gp}^{new}, Q_2 = \gamma \times Pri_E^{new}$ and signs the obtained data as $\sigma = Sign_{Pri_{sign}}(Q_1||Q_2||\tau_3)$, where τ_3 is the current timestamp. RA now sends the data along with the signature to the vehicles.
- When a vehicle receives σ, Attached TPD validates the signature using the public key Pub_{sign} and calculates the new group key as $Key_{gp}^{new} = Q_1 \mod ck_i$ and new private key for encryption as $Pri_E^{new} = Q_2 \mod ck_i$. Using these updated values, TPD further computes $G_i^{new} = G_i^{old} \oplus Key_{gp}^{old} \oplus Key_{gp}^{new}, Pri_E{}'^{new} = Pri_E{}'^{old} \oplus Pri_E^{old} \oplus Pri_E^{old}$.

NoMAS ensures Perfect Forward Secrecy (PFS) using these key updates (SKU and HKU). The general meaning of PFS is that the keys used for the encryption and decryption process are periodically updated such that if the recent key is disclosed or compromised, it does not affect the entire sensitive data.

3.1.3 Security Analysis

This section discusses in detail why an attack is not feasible against the proposed NoMAS mechanism. Table 3.3 presents a detailed security analysis of the NoMAS scheme by examining various potential attack vectors, describing each attack, and outlining the specific protection mechanisms employed by NoMAS to mitigate these threats. This comprehensive analysis highlights the robustness of NoMAS against a range of adversarial strategies, ensuring the integrity, confidentiality, and availability of the IoV network.

Algorithm 5 Password update process to prevent the unauthorised access

Input: Number of vehicles N, $U_{i,a}$ with current password $PASS_{i,a}$, New password $PASS_{i,n}$
Output: Updated security parameters in TPD

for $i = 1$ to N **do**
  $U_{i,a}$ is an authenticated user with password $PASS_{i,a}$.
  TPD authenticates $U_{i,a}$.
  if $U_{i,a}$ is valid **then**
    OBU requests for $PASS_{i,n}$.
  end if
  if $PASS_{i,n}$ is valid **then**
    OBU calculates:
    $SPASS_{i,n} = h(PASS_{i,n} \| \eta_i)$,
    $\xi_{i,n} = h(PASS_{i,n} \| VID_i) \oplus \eta_i$,
    $\varepsilon_{i,n}^2 = h(VID_i \| ck_i \| SPASS_{i,n} \| ID_{RA})$,
    $VID_{i,n}^{'} = VID_i \oplus h(PASS_{i,n})$,
    $PVID_{i,n}^{'} = PVID_i \oplus SPASS_{i,n}$,
    $\varepsilon_{i,n}^{1'} = \varepsilon_i^1 \oplus SPASS_{i,n}$,
    $ck_{i,n}^{'} = ck_i \oplus SPASS_{i,n}$,
    $Pri_{E,n}^{'} = Pri_E \oplus SPASS_{i,n}$,
    $\varepsilon_{i,n}^3 = h(\varepsilon_i^1 \| \varepsilon_{i,n}^2)$,
    $\varepsilon_{i,n}^{3'} = \varepsilon_{i,n}^3 \oplus SPASS_{i,n}$,
    $G_{i,n} = G_i \oplus SPASS_i \oplus SPASS_{i,n}$.
  end if
  TPD updates these values.
end for

Table 3.3: Detailed security analysis of NoMAS

Attack vector	Description and protection mechanism
Replay attack	$\mathbb{A}$ may stop the message and try to update the timestamp value. Timestamps are used in each transmission so that $\mathbb{A}$ cannot update the timestamp value. When the vehicular node transmits data, it includes a timestamp factor τ_M. The freshness of the message is maintained by using timestamp calculations at the receiver side as $\tau(\text{receive}) - \tau_M < \Delta$, where Δ is the predefined threshold value depending on the communication environment. In delay-sensitive dense environments like IoV, Δ is slightly more than the worst-case average packet transmission time.
Modification	If $\mathbb{A}$ accesses the message information that is passed by any vehicular node through an insecure medium and wants to change the information illegally, NoMAS prevents this using the highly secure cryptographic calculations of Key_{gp}, Pri_E, Pri_{sign}. NoMAS uses three types of messages: alert/warning messages, SKU messages, and HKU messages. To access information from these messages, $\mathbb{A}$ requires Key_{gp}, Pri_E, Pri_{sign}, which is not feasible.
Repudiation	When OBUs send messages, a primary pseudo ID $PPVID$ is calculated for each OBU as $PPVID = h(Key_{gp} \| \tau_M) \oplus PVID$. The RA can retrieve the original pseudo ID as $PVID = h(Key_{gp} \| \tau_M) \oplus PPVID$ in the situation of repudiation, then find out the original identity VID from the table. This process protects NoMAS from repudiation attacks, and the attached timestamp restricts the vehicle from denying the time of message generation.
Impersonation	$\mathbb{A}$ can send a message request to nearby components and assume the identity of any legitimate user. It can calculate fake values and send them illegally. NoMAS prevents this by using the primary pseudo ID $PPVID$ instead of the original ID VID in transferred messages. $\mathbb{A}$ cannot find out VID of vehicles because $VIDs$ are preserved in the TPD using XOR operation and hash functions.
Location tracing	Each message transmitted to IoV components contains a unique primary pseudo ID calculated as $PPVID = h(Key_{gp} \| \tau_M) \oplus PVID$. These IDs are only accessed using the timestamp, and Key_{gp}, making it impossible for $\mathbb{A}$ to trace the vehicles.

Attack vector	Description and protection mechanism
Stolen OBU	Even if an OBU is stolen, accessing all stored information from TPD is difficult. If $\mathbb{A}$ gets information from TPD such as $PVID_i'$, ξ_i, ID_{RA}, VID_i', G_i, $\varepsilon_i^{1\prime}$, $\varepsilon_i^{3\prime}$, Pri_E', ck_i', $InPad$, $OutPad$, it cannot access sensitive information like VID_i, Pri_E, ck_i, Key_{gp} because they are preserved using hash function and XOR operations. Thus, NoMAS is protected from stolen OBU attacks.

Further, Table 3.4 provides a comprehensive security comparison of various authentication schemes against a set of potential attacks. The comparison includes the proposed NoMAS scheme and state-of-the-art schemes[101, 48, 113, 118, 119, 120, 121]. Each row corresponds to a specific type of attack, denoted as A1 through A9, and each column indicates whether a particular scheme provides protection against that attack. A1: Replay Attack, A2: Modification Attack, A3: Repudiation Attack, A4: Impersonation Attack, A5: Location Tracing Attack, A6: Stolen OBU Attack, A7: Sybil Attack, A8: Man-in-the-Middle Attack, A9: DoS Attack. ✓indicates that the scheme provides protection against the specified attack and × indicates that the scheme does not provide protection against the specified attack.

Table 3.4: Security comparison of NoMAS with state-of-the-art schemes

Attacks	[101]	[48]	[113]	[118]	[119]	[120]	[121]	NoMAS
A1	✓	✓	✓	✓	✓	✓	✓	✓
A2	✓	✓	✓	✓	✓	✓	✓	✓
A3	✓	✓	✓	✓	✓	✓	✓	✓
A4	✓	✓	✓	✓	✓	✓	×	✓
A5	×	✓	×	✓	✓	×	×	✓
A6	×	×	×	×	×	×	×	✓
A7	×	×	×	×	×	×	×	✓
A8	×	×	×	×	×	×	×	✓
A9	×	×	×	×	×	×	×	✓

Formal security proof based on BAN logic: BAN logic is a set of standards for describing and evaluating communication protocols[78]. BAN logic explicitly lets its users decide if the information shared is reliable and shielded from different security attacks or not. BAN logic uses some special symbols/notation and primary rules for security proof[78]. In NoMAS, let's say vehicle V_i and V_j are in the range of RSCN R_j and V_i broadcast a warning message/ alert M. The following objectives have to be satisfied to ensure the validity of a vehicle and messages. To understand these objectives, BAN logics are described in the subsection 1.5.3.

- **Objective 1:** $V_i|\equiv V_i \overset{Key_{gp}}{\longleftrightarrow} V_j/R_j$
- **Objective 2:** $V_i|\equiv V_j/R_j|\equiv V_i \overset{Key_{gp}}{\longleftrightarrow} V_j/R_j$
- **Objective 3:** $V_j/R_j|\equiv V_i \overset{Key_{gp}}{\longleftrightarrow} V_j/R_j$
- **Objective 4:** $V_j/R_j|\equiv V_i|\equiv V_i \overset{Key_{gp}}{\longleftrightarrow} V_j/R_j$

Now, the standard messages transmitted in the proposed method are represented as follows:

1. **message1:** $V_i \rightarrow V_j/R_j : \{PPVID_i, G_i, \tau_M\}_{PVID_i}$

2. **message2:** $V_j/R_j \rightarrow V_i : \{PPVID_i, ck_i, \tau_j\}_{PVID_i}$

Following is the list of assumptions that are considered to prove the validity of the proposed protocol:

1. L_1: $V_j/R_j| \equiv \#(\tau_M)$
2. L_2: $V_i| \equiv \#(\tau_j)$
3. L_3: $V_j/R_j| \equiv V_i \implies (V_i \overset{Key_{gp}}{\longleftrightarrow} V_j/R_j)$
4. L_4: $V_i| \equiv V_j/R_j \implies (V_i \overset{Key_{gp}}{\longleftrightarrow} V_j/R_j)$
5. $L_5 : V_j/R_j| \equiv (V_i \overset{PVID_i}{\longleftrightarrow} V_j/R_j)$
6. $L_6 : V_i| \equiv (V_i \overset{PVID_i}{\longleftrightarrow} V_j/R_j)$

Using the above-mentioned assumption$(L_1 - L_6)$ and the BAN logic's primary rules[78], a strong proof of the proposed authentication method is given below:

- **Step 1:** Initially V_j/R_j receives $message_1$: $\mathbb{D}_1 : V_j/R_j \lhd \{PPVID_i, G_i, \tau_M\}_{PVID_i}$
- **Step 2:** BAN logic's message meaning rule[78] can be considered using L_5 and $\mathbb{D}_1$. Then, V_j/R_j believes, $message_1$ is transmitted by V_i: $\mathbb{D}_2 : V_j/R_j| \equiv V_i| \sim (PPVID_i, G_i, \tau_M)$
- **Step 3:** By considering the freshness rule[78], L_1 and $message_1$, it is concluded that the $message_1$ is fresh: $\mathbb{D}_3 : V_j/R_j| \equiv \#(PPVID_i, G_i, \tau_M)$
- **Step 4:** By considering nonce verification rule[78], $\mathbb{D}_2$, and $\mathbb{D}_3$, V_j/R_j believes that V_i believes $message_1$: $\mathbb{D}_4 : V_j/R_j| \equiv V_i| \equiv (PPVID_i, G_i, \tau_M)$
- **Step 5:** The belief rule[78] can be considered using $\mathbb{D}_4$, hence, V_j/R_j believes G_i that is used in $message_1$: $\mathbb{D}_5 : V_j/R_j| \equiv V_i| \equiv (G_i)$
- **Step 6:** Now, V_i receives the $message_2$: $\mathbb{D}_6 : V_i \lhd \{PPVID_i, ck_i, \tau_j\}_{PVID_i}$
- **Step 7:** Again, by considering the message meaning rule[78], L_6 and $\mathbb{D}_6$, V_i concludes that the $message_2$ is sent by the V_j/R_j: $\mathbb{D}_7 : V_i| \equiv V_j/R_j| \sim (PPVID_i, ck_i, \tau_j)$
- **Step 8:** By considering the freshness rule and assumption L_2, V_i checks the freshness of the $message_2$: $\mathbb{D}_8 : V_i| \equiv \#(PPVID_i, ck_i, \tau_j)$
- **Step 9:** Now, using the nonce verification rule, $\mathbb{D}_7$ and $\mathbb{D}_8$, V_i finds that V_j/R_j also belives $message_2$: $\mathbb{D}_9 : V_i| \equiv V_J/R_j| \equiv (PPVID_i, ck_i, \tau_j)$
- **Step 10:** By considering the belief rule[78] and $\mathbb{D}_9$, V_i believes ck_i that is used in $message_2$: $\mathbb{D}_{10} : V_i| \equiv V_j/R_j| \equiv (ck_i)$

- **Step 11:** V_j/R_j and V_i consider each other to be capable of calculating the group key Key_{gp} using the shared parameters:
  $\mathbb{D}_{11} : V_j/R_j| \equiv V_i| \equiv V_i \overset{Key_{gp}}{\longleftrightarrow} V_j/R_j$ **(Objective 4)**
  $\mathbb{D}_{12} : V_i| \equiv V_j/R_j| \equiv V_i \overset{Key_{gp}}{\longleftrightarrow} V_j/R_j$ **(Objective 2)**
- **Step 12:** By considering $\mathbb{D}_{11}$, $\mathbb{D}_{12}$, L_3, L_4 and jurisdiction rule[78], the BAN logic proof is completed.
  $\mathbb{D}_{13} : V_j/R_j| \equiv V_i \overset{Key_{gp}}{\longleftrightarrow} V_j/R_j$ **(Objective 3)**
  $\mathbb{D}_{14} : V_i| \equiv V_i \overset{Key_{gp}}{\longleftrightarrow} V_j/R_j$ **(Objective 1)**

Hence, $\mathbb{D}_{11}$, $\mathbb{D}_{12}$, $\mathbb{D}_{13}$ and $\mathbb{D}_{14}$ are the desired objectives that are mentioned initially. The proposed algorithm is also verified using ProVerif software based code and Pi calculus.

Correctness proof using ProVerif tool: ProVerif[182] is verification or testing software that uses Horn Theory[183] for formal analysis of security protocols. The software is a fully automatic analyzer that performs the possible attacks (that an attacker can perform) based on the recurring parallel executions of the security protocol. This software's primary advantage is that there is no upper bound for the number of parallel execution of the protocol that an attacker may perform. Thus, it can discover those attacks that need an unspecified number of parallel executions of the mechanism. ProVerif is based on Pi calculus. Using the ProVerif tool, the scheme is turned into a set of Horn clauses. In these clauses, different security properties in terms of queries are used. $\mathbb{A}$ is treated as an agent in ProVerif that can access the data from the network. For the accomplishment of the authentication process, an assertion based on some event is considered, which is "if event q has been executed, then event p has been executed." Thus the query can be reformed as "$event(q) \implies event(p)$." The truth table for this query is explained in Table 3.5. When a

Table 3.5: Truth table for proVerif tool

event(p)	event(q)	event(q) $\implies$ event(p)
0	0	1
0	1	0
1	0	1
1	1	1

security scheme is verified using ProVerif, different security properties in terms of queries are fed as input, and the code is then executed. If the query testing is failed, there is an attacker, and ProVerif starts tracing that attacker.

For the execution of the NoMAS algorithm using the ProVerif tool, the optional parts are eliminated and continued with the necessary code for the sake of the simplicity of the code. For this, the rules and query equations are reconstructed according to the scheme. For example, the term $sign(M, MAC_{Key_{gp}})$ refers to the authentication function with the respective MAC code using the key, and this function's output will be the sign that is used to verify the message. The equation for this is written below: $sign(M, MAC_{Key_{gp}}) \rightarrow sign$. As the communication between network entities is performed via an unsecured public channel, so that channel has to be declare free using the rule: **free** $\mathbb{C} : channel$, where $\mathbb{C}$ denotes the communication channel. Similarly, the private keys for RA and vehicle have to be declared. So the rule to declare these private keys are given below: **free** $Pri_{sign}, Pri_E, Key_{gp} : keys$. An authentication process is performed using the ProVerif tool and verified it using the truth table of the events given in Table 3.5. As of now, only authentication process of the vehicle is discussed, and the authentication of the rest of the entities will also be performed

similarly to this only. In this scheme, two components are involved in authentication, RA and vehicle. So there are four possible events named $event(p), event(q), event(r)$, and $event(s)$. The explanation for these events is as follows:

- $event(p)$ denotes the session request from RA to the vehicle.
- $event(q)$ represents the acknowledgment of the acceptance of the session request.
- $event(r)$ is the respective response from the vehicle, and
- $event(s)$ denotes that the RA can now start the process for authentication.

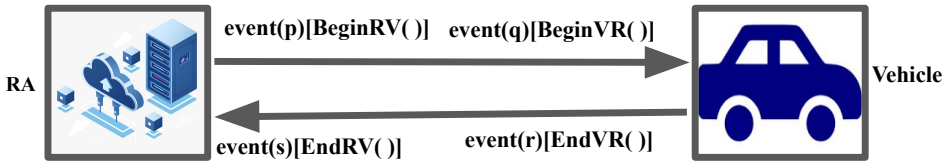


Figure 3.4: Process to create a session for authentication of vehicle

As shown in Figure 3.4, the functions for the above events are given as $BeginRV()$, $BeginVR()$, $EndVR()$, and $EndRV()$ and the query equations for these events are written as:
$event(EndVR()) \implies event(BeginVR())$
$event(EndRV()) \implies event(BeginRV())$
$event(EndRV()) \implies event(BeginRV()) \;\&\&\; event(BeginVR())$
Using these basic queries and the ProVerif tool code, confidentiality results are achieved that are shown in Table 3.6. It is shown in the table that the result for initial identities of the network components are mentioned as "No input" because, those parameters are ignored for simplicity. Similar to the confidentiality

Table 3.6: Confidentiality verification results using ProVerif tool

Test parameters	Initial status	Results from ProVerif
Pri_{sign}, Pri_E	Secure	Secure
ck_i	Secure	Secure
Key_{gp}	Not defined	Secure
UID, VID, ID_{RA}	Not defined	No input
$MAC_{Key_{gp}}$	Secure	Secure
G_i	Secure(encrypted)	Secure

check, the authentication queries of the scheme were tested using the ProVerif tool, and the results obtained are mentioned in Table 3.7. The results from ProVerif for the mentioned queries were obtained as true, indicating that the condition created for authentication was successfully verified by the ProVerif tool.

Table 3.7: Authentication test results using ProVerif tool

Query	Output
$event(BeginVR()) \implies event(BeginRV())$	True
$event(EndRV()) \implies event(EndVR())$	True
$event(EndRV()) \implies event(BeginRV())\&\&event(BeginVR())$	True

3.1.4 Performance Analysis and Simulation Results

NoMAS is based on the HMAC with less computation and communication overhead compared to identity-based authentication, session key-based, or public key-based authentication schemes. Consequently, NoMAS is having better performance in terms of computation and communication overhead. The following subsections provide a performance analysis along with the simulation results.

Performance Analysis

In this section, the performance of NoMAS is compared with state-of-the-art schemes[101, 48, 113, 116, 118, 119, 120, 121]. NoMAS uses 32-bit timestamps, 160-bit primary pseudo IDs ($PPVID_i$), 160-bit group key, SHA-1 for a hash function, AES-128 encryption, and HMAC for the MAC function. The scheme's different calculations include a one-way hash function, MAC function, one encryption function, point multiplication, mod function, and pairing operation. Computation time for these operations are symbolized as $T_h, T_{mac}, T_E, T_m, T_a, T_{mod}, T_p$ respectively. The code was executed on both a Desktop System and a Raspberry Pi 4, and this subsection describes the specifications of both systems.

- **Desktop system:** NoMAS is implemented on a Windows 10 machine with the hardware configuration as Intel 64-bit Core i5 at a frequency of 4.1 GHz and 8GB DDR4 RAM, and software configuration as Netbeans 8.2 and Java 8. 20 RSCN, 8 controllers, and 300 vehicles were considered.
- **Raspberry Pi 4:** NoMAS is also executed on the Raspberry Pi 4 chip having a lightweight and high computation power. The hardware configuration of Raspberry Pi is Broadcom BCM2711, Quad-core Cortex-A72 1.5GHz, and 4GB RAM, and software configuration is a Linux-based operating system called Noobs v3.3.1, Netbeans IDE 11, and java 8. The simulation environment for NoMAS is similar to the desktop system. Table 3.8 shows the time taken for the execution of each operation on both the Desktop and Raspberry Pi system.

Table 3.8: Execution time of different cryptographic functions

Operation	On Desktop (ms)	On Raspberry Pi (ms)
T_h	0.15	1.82
T_E	45.64	49.13
T_{mac}	10.34	18.47
T_m	4.75	7.68
T_a	7.54	10.86
T_{mod}	20.32	50.23
T_p	30.78	35.45
T_{M2P}	4.48	6.54

Computation cost: The performance of NoMAS is compared with the existing state-of-the-art methods. Some of them are MAC-based and others are using different methodologies for authentication. The efficiency of proposed scheme is that it uses only five hash functions and the cryptographic equation for authentication in NoMAS is obtained as $5T_h$ which is very less compared to the state-of-the-art schemes. The cryptographic equations in Table 3.9 are derived by analyzing the number of particular cryptographic operations in the algorithm. The total computation cost is calculated by simplifying the generated cryptographic equations using the execution time given in Table 3.8 for both Desktop and Raspberry Pi systems. The total computational cost is calculated by putting each cryptographic operation's values in the given

Table 3.9: Cryptographic equations having number of operations required for NoMAS and state-of-the-art schemes

Schemes	Cryptographic Equations
Zhang et al.[101]	$5T_m + 3T_h + 2T_a$
Jenefa et al.[48]	$2T_m$
Sutrala et al.[113]	$T_a + 5T_m + 3T_h$
Bagga et al.[116]	$10T_h + 9T_m + 2T_a$
Lyu et al.[118]	$3T_E + 3T_h + T_{mac}$
Wang et al.[119]	$8T_h + 2T_{mac}$
Asl et al.[120]	$4T_h + 3T_m + T_{mac} + 2T_p$
Li et al.[121]	$2T_E + 2T_h + 2T_{mac}$
NoMAS	$5T_h$

cryptographic equations. For example, the total execution time for Lyu et al. scheme is calculated as $3T_E + 3T_h + T_{mac} = 3 \times 59.64 + 3 \times 0.15 + 10.34 = 189.71ms$ on the Desktop system. Similarly, the same calculations can be performed for all the schemes on both systems. The comparison of computation cost between existing schemes and NoMAS is shown in Figure 3.5. According to Figure 3.5a and 3.5b, It can be clearly stated that NoMAS has a significantly lower computation cost compared to the other schemes on both the Desktop (0.75 ms) and Raspberry Pi (9.1 ms) environments.

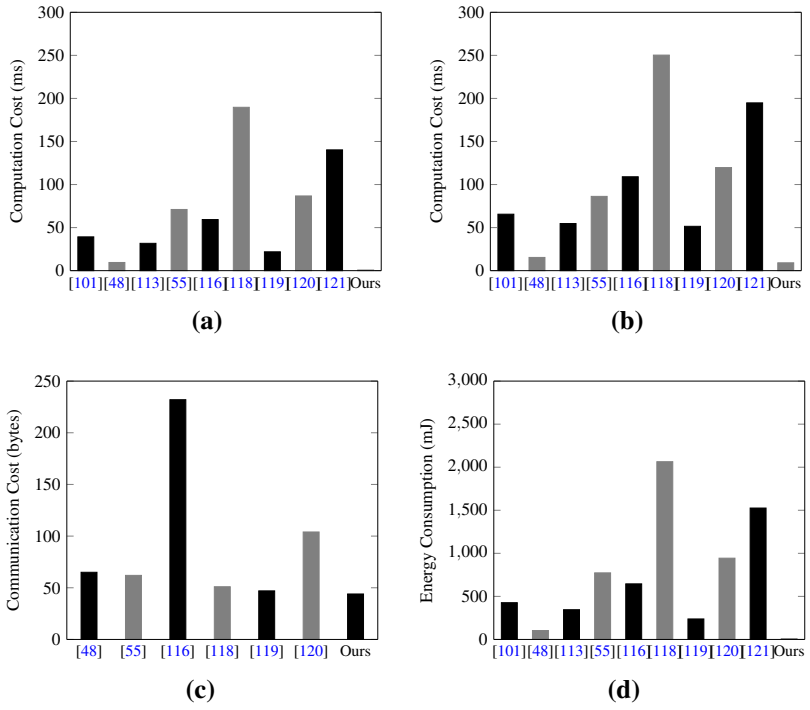


Figure 3.5: Comparison of NoMAS with state-of-the-art schemes in terms of: (a) Computation cost (Desktop), (b) Computation cost (Raspberry Pi), (c) Communication cost (bytes), (d) Energy consumption on Desktop (mJ)

Communication overhead: The communication overhead for Lyu et al.[118], Wang et al.[119], Asl et al.[120], Jenefa et al.[48], Bagga et al.[116], and the NoMAS scheme is considered. The rest of the state-of-the-art schemes do not provide information about communication overhead, hence it is quite difficult to compare the proposed scheme with those schemes. [118] requires 160-bit (20 bytes) for MAC, 184-bits (23

bytes) for pseudo-identity, and 64-bit (8 bytes) for timestamp (Total of 51 bytes). Similarly, [119], [120], [48], and [116] require 47 bytes, 104 bytes, 65 bytes, and 232 bytes respectively for authentication. NoMAS requires 44 bytes (20 bytes for two MAC functions and 4 bytes for timestamps). NoMAS reduces the communication overhead up to approximately 81%. Figure 3.5c depicts the comparison of communication overhead and shows that NoMAS is better than other existing schemes.

Energy consumption: In message exchange, some energy is consumed to perform all the calculations. This consumption can affect the performance of the technique. The formula to calculate energy consumption is:

$$Energy = Total_{CC} \times Power_{max}$$

, Where, $Total_{CC}$ is the total computational cost and $Power_{max}$ is the maximum processing power. For the wireless transmission, max processing power is considered 10.88 W[179]. Here is the sentence revised in the third person: The computational costs given in Table 3.9 for the Desktop system were used to calculate the consumed energy. The effect of energy consumption on different schemes can be analyzed through the graph representation in Figure 3.5d.

Complexity: The worst time complexity for a universal one-way hash function is $O(n)$, where n denotes the size of the massage. The complexity for NoMAS is $O(n)$ for both the vehicle and the RA. Table 3.10 shows the worst-case time complexity for NoMAS and other state-of-the-art schemes. Here, f^{κ} and $f^{\kappa+1}$ represent ECC-point multiplication and point addition for a finite f-bit field, while $M(l)$ denotes the MAC operation on finite l-bits. It is generally considered that the complexity for encryption is $O(1)$. Therefore, it can be concluded that NoMAS is efficient in terms of complexity.

Table 3.10: Complexity comparison of NoMAS with state-of-the-art schemes

Schemes	**Time complexity**
Zhang et al.[101]	$O(n+f^{\kappa}+f^{\kappa+1})$
Jenefa et al.[48]	$O(f^{\kappa})$
Sutrala et al.[113]	$O(n+f^{\kappa}+f^{\kappa+1})$
Sikarwar et al.[55]	$O(n^2+f^{\kappa})$
Bagga et al.[116]	$O(n+f^{\kappa}+f^{\kappa+1})$
Lyu et al.[118]	$O(n+M(l))$
Wang et al.[119]	$O(n+M(l))$
Wang et al.[120]	$O(n+f^{\kappa}+M(l)+n^2)$
Li et al.[121]	$O(n+M(l))$
Ours	$O(n)$

Simulation Environment and Case Study

Simulation of the NoMAS approach is performed using the SUMO and NS-2 simulators. The traffic scenario for an urban city in India (Jaipur) is generated in the form of a .osm file from an open street map. With the help of the SUMO simulator, the generated .osm file is converted to an .xml file and the simulation is performed using the NS-2 simulator. Three scenarios are considered for the simulation; CASE 1: Initially, 50 vehicles with a range of 200 m are taken, and the number of RSCN is 3, CASE 2: For this scenario, 100 vehicles and 7 RSCN are considered. The communication range of the vehicles is 300 m. CASE 3: for the third case, 150 vehicles and 10 RSCNs are considered.

Table 3.11: Configuration of the network

Parameters	**Values**
Network simulation range (Highway)	$1000m * 500m$
Network simulation range (City)	$2500m * 2500m$
Vehicle's communication range	$200m, 300m, 400m$
Number of vehicles	50, 100, 150
Number of RSCN	3, 5, 10
Simulation time	$100s$
Average speed of the vehicle	$10m/s, 15m/s, 20m/s$
Communication protocol	DSRC (IEEE 802.11p)
Data size	256 bytes
Encryption	AES-128
Group key size	160 bit
Buffer size of NoMAS	60 packets
Number of simulation	5

The communication range of vehicles is 400 m. Along with the above cases, vehicles with different speeds (10 m/s, 15 m/s, and 20 m/s) are considered. Five simulations, each with a duration of 100 seconds, are performed. The communication range for the RSCN is set to 1000 m. For routing, a well-known reactive routing protocol, the Adhoc On-demand Distance Vector (AODV) protocol, is used. The connection setup delay for the AODV protocol is lesser than other reactive and proactive routing protocols. Other network parameters used for the simulation of NoMAS are configured as shown in Table 3.11. From the .tcl script, the physical/MAC layer overhead due to message transmission is derived. Figure 3.6a indicates the PHY/MAC layer overhead caused by message transmission in the IoV environment. It can be seen that the overhead continues to decrease over time. The decrease in the PHY/MAC layer overhead indicates higher efficiency and scalability of the scheme. When the NoMAS scheme is simulated on the NS-2 simulator using predefined parameters for CASE 1, CASE 2, and CASE 3, one-way delay, packet delivery ratio, packet loss ratio, and throughput for the proposed scheme are also examined.

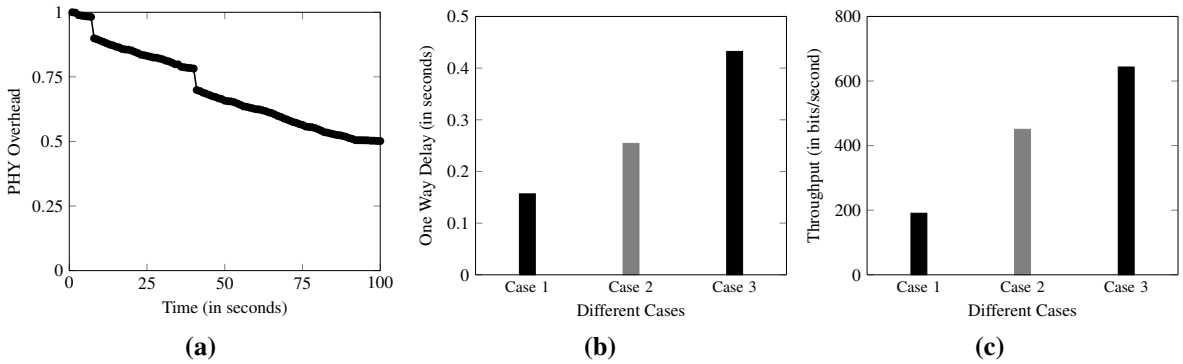


Figure 3.6: Simulation results of NoMAS in terms of: (a) Physical layer overhead during message transmission, (b) Average one way delay (seconds), (c) Average throughput (bits/second)

One-way delay or end-to-end delay: The definition of One-Way Delay (OWD) is the average travel time of a message from a source to a specific destination. It can be formulated as:

$$OWD = \sum_{i=0}^{N_m} (T_R - T_S)/N_m, \tag{3.3}$$

Here, T_R and T_S denote the time at the receiver side and source side respectively. N_m is the number of messages sent from source to destination. In the context of authentication schemes like NoMAS, OWD can be defined as the total time required for authentication, message communication, and key establishment. So for NoMAS, Three different values are obtained considering the above-mentioned three cases. The average OWD values for the three different cases are 0.15665, 0.25438, and 0.43234 seconds. The increase in the average OWD values in Figure 3.6b is because of the increasing number of vehicles in different cases.

Packet delivery ratio :PDR is the ratio of successfully delivered packets to the total number of packets sent for communication. For a reliable scheme, the value of PDR should be higher. Figure 3.7a shows the PDR values for the application packets in 100 seconds.It is observed that PDR values for NoMAS range between approximately 70% to 90% for all three cases. The PDR value decreases with the increased number of vehicles due to traffic and congestion.

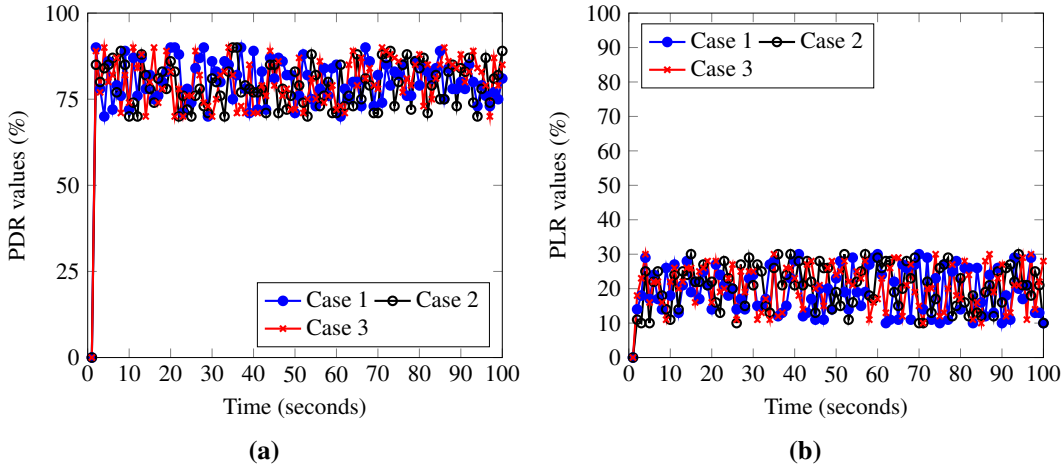


Figure 3.7: Simulation results of NoMAS in terms of: (a) Packet delivery ratio for different cases (b) Packet loss ratio for different cases

Packet loss ratio: PLR can be considered as the number of lost packets per second. The formula for PLR can be written as: $PLR = \frac{N_{lp}}{T_s}$. Here, N_{lp} denotes the number of lost packets, and T_s is the simulation time. The PLR values for the proposed NoMAS scheme are also examined with respect to these three different cases. Figure 3.7b shows the obtained PLR values for 100 seconds in the above-mentioned three cases. The PLR values lie between approximately 10 seconds to 30 seconds.

Network throughput: The last simulation parameter considered is the throughput, calculated as the total number of bits transmitted per second. The formula for this can be written as:

$$Throughput = \frac{(Total no. of packets \times packet size)}{Simulation time}. \tag{3.4}$$

The average network throughput is obtained as 190.69, 450.55, and 643.43 bits/second for CASE 1, CASE 2, and CASE 3, respectively, as shown in Figure 3.6c. It can be seen that the throughput slightly increases from CASE 1 to CASE 3 due to the increased number of messages exchanged.

3.1.5 Implementation Considerations in Diverse IoV Environments

In real-world situations, the IoV includes many different types of vehicles, communication methods, and regions with their own rules and infrastructure. This kind of environment is called a diverse IoV environment. It includes cars from different manufacturers, communication technologies like DSRC, 5G, or C-V2X, and vehicles moving through areas with different network coverage and legal systems. Although the NoMAS scheme is designed to be lightweight and fast, putting it into practice in such a varied setup may face a few important challenges. First, there are legal and privacy concerns. For example, NoMAS uses pseudo-identities and key updates, but laws like General Data Protection Regulation (GDPR) or California Consumer Privacy Act (CCPA) might treat those identities as personal information, and there may be legal rules about how those keys should be shared or stored. Second, there is the issue of compatibility. Since vehicles and systems might use different communication standards or security methods, making NoMAS work smoothly across all of them may require extra adjustments or standardization. Third, there is complexity in deployment. NoMAS needs regular key updates and consistent operation across all components like OBUs, RSUs, and central authorities. In areas with poor connectivity or fast-moving vehicles, keeping everything updated and in sync can be difficult. These challenges do not affect the core design of NoMAS but do show that extra planning and support will be needed to use it effectively in real-world, diverse IoV environments.

3.2 STATIC BATCH VERIFICATION

DSRC is a key enabler for IoV, providing a reliable protocol for V2X communication[184]. DSRC operates within the 5.8 to 5.9 GHz frequency band, specifically allocated for vehicular communication. This band offers seven 10 MHz channels and a single 20 MHz channel, supporting low-latency, high-speed communication suitable for vehicular environments. The communication timeframe for DSRC typically ranges from 50 milliseconds to 100 milliseconds, necessitating real-time processing and response. In this context, ensuring the security of messages while maintaining the efficiency of communication is paramount[185]. Traditional security mechanisms[186, 61, 187], which often involve heavy cryptographic operations, introduces significant computational and communication overhead. This overhead is not desirable in the DSRC-enabled IoV environment, where real-time communication is critical[188]. Therefore, there is a urgent need for security solutions that can provide robust protection without compromising the efficiency of communication. One of the primary challenges in this domain is the verification of digital signatures of the transferred messages. Verifying each message individually, though secure, can lead to substantial delays and increased computational load, making it impractical for real-time applications. To address this issue, batch verification techniques have been proposed[178], allowing multiple signatures to be verified simultaneously, thereby reducing the overall computational cost. However, traditional batch verification methods[189, 122, 123, 61, 190, 47] often depend on the size of the batch, meaning the efficiency gains diminish as the batch size increases. This dependency on batch size can be problematic in dynamic vehicular environments where the number of messages to be verified can vary significantly. Additionally, these

methods can still impose considerable communication overhead due to the aggregation of signatures and the need for complex cryptographic operations.

To overcome these limitations, the proposed methodology[55] introduces a static batch verification technique for IoV that is independent of batch size, maintaining constant computation cost with a verification equation of $2T_p + 2T_m$ regardless of the number of messages. By employing lightweight cryptographic operations, such as one-way hash function[71] and XOR operations, it significantly reduces computation cost, making it suitable for real-time applications. The use of bilinear pairing[191] establishes secure connections and a robust cryptographic framework, enhancing the security of the verification process. Key contributions for the proposed methodology is described below:

- The methodology employs lightweight cryptographic operations, including the use of one-way hash functions instead of more complex Map-To-Point (M2P)[122] hash functions[192]. These one-way hash functions significantly reduce computation costs, enhancing the system's suitability for real-time IoV applications. This focus on lightweight operations ensures that the scheme can handle the high-speed demands of vehicular communication without sacrificing security.
- Bilinear pairing is utilized to establish secure connections between cryptographic groups, providing a robust mathematical foundation for the verification process. The properties of bilinear maps, such as non-degeneracy and symmetry, are leveraged to create a secure and efficient cryptographic framework that supports batch verification.
- The security analysis demonstrates that the proposed scheme can withstand various attacks[14], including replay, modification, impersonation, non-repudiation, man-in-the-middle, and collision induction attacks. This robustness is achieved through the use of pseudo-identity ($PVID_i$), private keys derived from these IDs, and secure timestamp mechanisms. These components ensure the integrity and authenticity of communications, providing a strong defense against security threats in IoV.
- The performance analysis highlights the method's efficiency in terms of computation cost, time complexity, transmission overhead, and energy consumption. The proposed scheme outperforms existing methods, offering lower delays and reduced overheads, making it highly suitable for the real-time requirements of IoV networks.

3.2.1 Proposed Scheme: Static Batch Verification

Network and adversary models described in subsection 3.1.1 are utilized to explain the proposed methodology. Before explaining the secure and efficient batch verification scheme, a parallel layered architecture for the IoV network is proposed, as described in Figure 3.8. The proposed layered architecture is designed to ensure robust security while facilitating efficient and reliable communication. This architecture includes five layers, each with distinct responsibilities, and integrates an additional parallel validation layer that addresses the primary security requirements: confidentiality, authentication, and integrity. The detailed description of each layer is given below:

1. **Connection layer:** At the foundation of the IoV architecture lies the connection layer, which encompasses all the physical and data link components necessary for establishing communication between vehicles, infrastructure, and other devices. This layer supports V2V communication for sharing information such as speed, position, and other telemetry data, utilizing IEEE 802.11p standards for

WAVE[193]. It also includes V2I communication for receiving traffic updates, road conditions, and other critical information, and V2X communication, extending interactions to pedestrians, cyclists, and other road users. The robust physical layer includes high-frequency transceivers and advanced modulation schemes to ensure reliable communication under various environmental conditions.

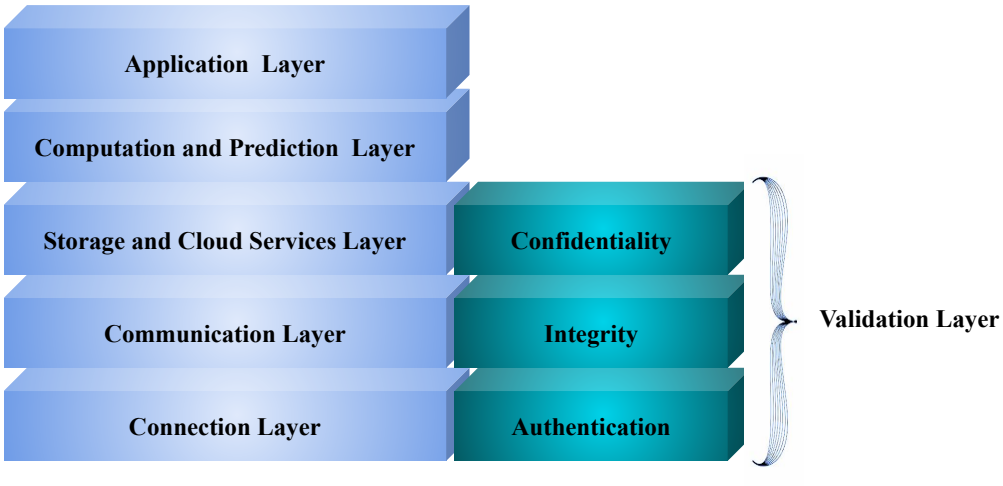


Figure 3.8: Proposed layered architecture for the secure IoV

2. **Communication layer:** Above the connection layer is the Communication Layer, responsible for handling the protocols and technologies required for data exchange. It ensures efficient and reliable message transmission across the network using protocols like DSRC[6] for high-speed, low-latency communication, along with 5G and LTE for broader coverage and higher bandwidth[8]. The communication layer also integrates Wi-Fi and Bluetooth for short-range communication, leveraging frequency hopping and adaptive modulation to mitigate interference and enhance link reliability. Additionally, this layer employs network slicing and edge computing[72] to optimize resource allocation and minimize latency, ensuring that critical IoV applications receive the necessary network performance.

3. **Storage and cloud services layer:** It manages the storage of data collected from various IoV components and provides cloud-based services for data processing and analytics. This layer includes distributed databases capable of handling large volumes of data generated by IoV devices, using technologies such as NoSQL databases for scalability and flexibility. Cloud computing services offer elastic processing power and sophisticated analytics platforms, enabling complex data analysis and machine learning model training[75]. Edge computing infrastructure complements cloud services by processing data locally to reduce latency and bandwidth usage, employing micro-services architectures to enhance scalability and fault tolerance[62]. This layer ensures data availability and durability through redundant storage and data replication strategies, leveraging cloud-native technologies for seamless scalability.

4. **Computation and prediction layer:** It processes data and makes predictions to enhance decision-making capabilities within the IoV system. This layer utilizes advanced data analytics techniques to extract actionable insights and patterns from IoV data, such as traffic flow analysis, anomaly detection, and predictive maintenance[44]. It employs machine learning models for predictive analytics, like traffic congestion prediction and potential accident detection, using frameworks like TensorFlow and

PyTorch for model development and deployment. Real-time processing systems within this layer leverage in-memory computing and stream processing frameworks, such as Apache Flink and Apache Kafka[194], to provide immediate feedback and control, ensuring that latency-sensitive applications, such as collision avoidance systems, operate efficiently. This layer employs parallel processing and hardware acceleration, including GPUs and FPGAs[195], to enhance computational efficiency and support real-time analytics.

5. **Application layer:** It provides the interface for various IoV applications and services, including navigation and routing for real-time route planning and navigation based on current traffic conditions, safety applications for collision avoidance and emergency response services, and infotainment services for media streaming and internet access[55]. This layer employs context-aware computing and user-centric design principles to deliver personalized and adaptive services, utilizing artificial intelligence to enhance user experience and system performance.

6. **Validation layer:** A validation Layer is integrated across the architecture to address confidentiality, authentication, and integrity. Confidentiality is ensured through existing encryption techniques, such as AES[196] and RSA[197], and access control mechanisms that enforce strict data access policies. Authentication is managed through a robust PKI[100] and Multi-Factor Authentication (MFA)[113], combining factors such as passwords, biometrics, and hardware tokens to verify the identities of devices, users, and services. Integrity is maintained using cryptographic hash functions like SHA-256[71] to generate unique hashes for data verification, and digital certificates issued by trusted Certificate Authorities (CAs) to validate the integrity and origin of data. The security layer employs advanced Intrusion Detection and Prevention Systems (IDPS)[14] and Anomaly Detection Algorithms (ADA) to safeguard against cyber threats.

Table 3.12: Notations used in batch verification scheme for IoV

Notation	**Explanation**
V_i	The i^{th} vehicle of the network
$\mathbb{H}_i, \mathbb{H}_j$	Cyclic additive Group
$\mathbb{H}_T$	Cyclic multiplicative group
g_1, g_2	generator of cyclic group $\mathbb{H}_i, \mathbb{H}_j$
e	A bilinear map: $\mathbb{H}_i \times \mathbb{H}_j \rightarrow \mathbb{H}_T$
o	The order of $\mathbb{H}_i, \mathbb{H}_j$
$\{Mkey_1, Mkey_2\}$	A Master keys for the vehicle
$\{Pubkey_1, Pubkey_2\}$	A set of public keys for RA
VID_i	i^{th} vehicle's registration ID
$PASS$	Password to get access to the TPD
$\{PVID_1^i, PVID_2^i\}$	A set of pseudo-IDs for V_i
$\{PriKey_1^i, PriKey_2^i\}$	A set of private keys for V_i
M_i	Generated message of V_i
$nonce$	Randomly generated nonce
$secret$	Secret key produced by TPD
$h()$	SHA-1 hash function
$\|\|$	Concatenation operation
Δ	Predefined threshold value

Further, to understand the motivation behind the proposed scheme, consider a scenario in which an RSCN interacts with hundreds of vehicles, each of which sends the RSCN a safety warning every 100-300

milliseconds[6]. Now, RSCN must verify each signature individually, causing a bottleneck at the network and theoretically taking a long time. Adopting a digital signature approach to preserve the integrity of transmitted messages can not be considered a feasible option, keeping in mind the overall network's performance. RSCN needs to verify 600-700 messages/sec in a dense IoV environment, compromising approximately 180-200 vehicles[23]. The computing capacity of RSCN prohibits the verification of such a large number of messages through a digital signature scheme.

An effective batch verification method[55] is being suggested in this thesis for heterogeneous communication to overcome the problem, as mentioned above. All the notations that are utilized to define the proposed scheme and their definitions are mentioned in Table 3.12. This scheme is divided into four following stages: (1) Key definition and allocation stage, (2) A process of generating a psuedo-ID and a private key, (3) The signature process of the message, and (4) Message verification stage.

Key definition and allocation stage: First, let $\mathbb{H}_i$ and $\mathbb{H}_j$ be cyclic additive groups of order p, generated by g_1 and g_2, respectively. Similarly, let $\mathbb{H}_T$ be a cyclic multiplicative group of the same order. A bilinear map e is defined as follows: $e : \mathbb{H}_i \times \mathbb{H}_j \rightarrow \mathbb{H}_T$. In the initial step, the RA randomly chooses $Mkey_1$ and $Mkey_2$ as the master keys for the device. Using these master keys, the RA calculates its own public keys as follows:

$$Pubkey_1 = Mkey_1 \cdot (g_1 \oplus g_2). \tag{3.5}$$

$$Pubkey_2 = Mkey_2 \cdot (g_1 \oplus g_2). \tag{3.6}$$

Here, the symbol "$\cdot$" represents the ECC-based scalar multiplication. $Mkey_1$ and $Mkey_2$, are then securely loaded into the TPD of the particular vehicle. The public parameters $\{\mathbb{H}_i, \mathbb{H}_j, \mathbb{H}_T, p, g_1, g_2, Pubkey_1, Pubkey_2\}$ are distributed to all components of the IoV to facilitate secure communication and operations across the network. Additionally, the RA is responsible for assigning a unique registration identity, VID, and a corresponding password, $PASS$, to each vehicle at the time of purchase. The vehicle's TPD is then activated using these ID-password credentials, ensuring that each vehicle is securely integrated into the IoV network with a unique cryptographic identity.

A process of generating a pseudo-ID and a private key: Vehicle's authentication should be performed at the initial level to achieve security and privacy. According to the given architecture in this paper, the initial layer is the connection layer, so authentication of vehicles at the connection layer is required to be performed. TPD, attached to the particular vehicle, generates a pseudo-ID ($PVID$) and private keys ($PriKey$) for that particular vehicle. Three following steps are considered for generating the $PVID$:

- **Log-in module:** For the log-in process, the user enters VID and $PASS$ given by RA. For TPD activation, these ID-password should be matched. After the activation, VID is now accessed by the following module, "pseudo-ID generation." This log-in module helps to secure the TPD from the adversary in case of a stolen vehicle.
- **Pseudo-ID generation:** After log-in, VID is used to generate the set of pseudo-IDs $\{PVID_1, PVID_2\}$. For this purpose, TPD facilitates the generation of a secret key $secret$ in a random manner, and that secret key is further distributed among the components of IoV. The set of pseudo-IDs $\{PVID_1, PVID_2\}$ are now calculated as follows:

$$PVID_1 = nonce \cdot (g_1 \oplus g_2), \tag{3.7}$$

$$PVID_2 = VID \oplus h(nonce \cdot Pubkey_1 || secret), \tag{3.8}$$

Here, *nonce* is a random nonce. The extra secret key is used to increase the security level.

- **Generation of private key:** The output of the previously discussed module is now served as the input for the current module, and therefore set of pseudo-IDs are accessible to generate the private keys related to the corresponding vehicle. Given below are descriptive details of the calculation involved in the generation of the private keys for the vehicle:

$$PriKey_1 = Mkey_1 \cdot (PVID_1), \tag{3.9}$$

$$PriKey_2 = Mkey_2 \cdot h(PVID_1||PVID_2||\tau_i)(g_1 \oplus g_2), \tag{3.10}$$

Here, τ_i is considered as the timestamp value that is used to avoid replay attack.

Signature generation: In a typical IoV environment, vehicular units broadcast alert messages or safety messages that influence network performance. Hence the message integrity should be maintained during the transmission. For this purpose, a message signature is generated. The signature for M_i is calculated as follows:

$$sign_i = PriKey_1 + h(M_i) \cdot PriKey_2. \tag{3.11}$$

V_i sends generated $sign_i$ along with the M_i and other necessary parameters such as $PVID$, and τ_i to all the neighboring components of environment.

Message verification phase: When a final message $\langle PVID, M_i, sign_i, \tau_i \rangle$ is received by nearby components of IoV, timestamp of the message τ_i is checked by the receivers. The message is valid, if $\tau_r - \tau_i < \Delta$, where τ_r is receiving timestamp and Δ is the predefined threshold value. On the other hand, if the above condition is not satisfied, the message is no longer valid and discarded.

- **Single message verification**: When the final message $\langle PVID, M_i, sign_i, \tau_i \rangle$ is received by the network components, $sign_i$ is now verified at the receiver end using the following equations. The received message is considered to be valid if the following equation is satisfied:

$$e\,(sign_i, g_1 \oplus g_2) = e(PVID_1, Pubkey_1) \cdot e(h(M_i(h(PVID_1||PVID_2||\tau_i))(g_1 \oplus g_2), Pubkey_2)). \tag{3.12}$$

Given below is the proof of the above equation:

$$e(sign_i, g_1 \oplus g_2),$$

$$= e(Pubkey_1 + h(M_i)PriKey_2, g_1 \oplus g_2),$$

$$= e(PriKey_1, g_1 \oplus g_2) \cdot e(h(M_i)PriKey_2, g_1 \oplus g_2),$$

$$= e(Mkey_1 PVID_1, g_1 \oplus g_2) \cdot e(h(M_i)Mkey_2 h(PVID_1||PVID_2||\tau_i)(g_1 \oplus g_2), g_1 \oplus g_2), \tag{3.13}$$

$$= e(PVID_1, Mkey_1(g_1 \oplus g_2)) \cdot e(h(M_i)h(PVID_1||PVID_2||\tau_i)(g_1 \oplus g_2), Mkey_2(g_1 \oplus g_2)), \tag{3.14}$$

$$e(sign_i, g_1 \oplus g_2) = e(PVID_1, Pubkey_1) \cdot e(h(M_i)h(PVID_1||PVID_2||\tau_i)(g_1 \oplus g_2), Pubkey_2). \tag{3.15}$$

- **Batch message verification**: Let's consider a situation where n messages from n different vehicles $V_1, V_2 ..... V_n$ are sent to the neighboring components of IoV. These messages are represented as $\langle PVID^1, M_1, sign_1, \tau_1 \rangle$, $\langle PVID^2, M_2, sign_2, \tau_2 \rangle ..... \langle PVID^n, M_n, sign_n, \tau_n \rangle$ respectively. The

received messages are considered to be valid if the following equation is satisfied:

$$e(\sum_{i=1}^{n} sign_i, g_1 \oplus g_2) = e(\sum_{i=1}^{n} PVID_1^i, Pubkey_1) \cdot e(\sum_{i=1}^{n} h(M_i).h(PVID_1^i||PVID_2^i||\tau_i)(g_1 \oplus g_2), Pubkey_2)). \tag{3.16}$$

Given below is the proof for the above equation:

$$e(\sum_{i=1}^{n} sign_i, g_1 \oplus g_2),$$

$$= e(\sum_{i=1}^{n} PriKey_1^i + h(M_i)PriKey_2^i, g_1 \oplus g_2),$$

$$= e(\sum_{i=1}^{n} PriKey_1^i, g_1 \oplus g_2) \cdot e(\sum_{i=1}^{n} h(M_i)PriKey_2^i, g_1 \oplus g_2),$$

$$= e(\sum_{i=1}^{n} Mkey_1 PVID_1^i, g_1 \oplus g_2) \cdot e(\sum_{i=1}^{n} h(M_i)Mkey_2(h(PVID_1^i||PVID_2^i||\tau_i))(g_1 \oplus g_2), g_1 \oplus g_2), \tag{3.17}$$

$$= e(\sum_{i=1}^{n} PVID_1^i, Mkey_1(g_1 \oplus g_2)) \cdot e(\sum_{i=1}^{n} h(M_i)h(PVID_1^i||PVID_2^i||\tau_i)(g_1 \oplus g_2), Mkey_2(g_1 \oplus g_2)), \tag{3.18}$$

$$= e(\sum_{i=1}^{n} PVID_1^i, Pubkey_1) \cdot e(\sum_{i=1}^{n} h(M_i)h(PVID_1^i||PVID_2^i||\tau_i)(g_1 \oplus g_2), Pubkey_2)). \tag{3.19}$$

3.2.2 Security Analysis

This section discussed how an attack is not feasible against the proposed batch verification mechanism in detail. For proving the security of the algorithm following implication represents the computation hardness of the algorithm:

$$\sim Secure(Communication) \implies V_{i=1}^{n} \sim hard(Assumptions)$$

. A strong security algorithm is one where the secret key's identity is not revealed to the adversary. The probability of breaking the messages should be negligible. As defined in security key generation, 2 abelian groups $\mathbb{H}_i$ and $\mathbb{H}_j$ are used. The public key generation requires the multiplication of the secret key and the generator key of the Abilene group, which constitute a hard logarithm problem. As the master key generated by this algorithm stored in secured TPD, which is assumed to be tamper-proof. It would be difficult for $\mathbb{A}$ to solve this computation in polynomial time. The detailed security analysis is described in the Table 3.13 that ensure the robustness of the proposed scheme towards certain types of attacks.

Table 3.13: Security analysis of the proposed scheme

Attack type	**Description and protection mechanism**
Replay attack	Whenever V_i sends M_i, it includes a timestamp factor of τ_i. The freshness of M_i is validated by using the timestamp calculations at the receiver side as $\tau_r - \tau_i < \Delta$, where, τ_r is the timestamp at the reciever end, and Δ is the predefined threshold value.

Continued on next page

Attack type	Description and protection mechanism
Modification	In the proposed method, this attack is not possible due to the following reasons: (1). The pseudo-ID for vehicles $PVIDs$ are generated using a secret key $secret$, which is not known to the public and not easy to derive as well. (2). Private keys for V_i are derived using $PVIDs$ and the current timestamp τ_i. These private keys are unknown to the public and cannot be easily guessed.
Impersonation attack	In the proposed method, TPD has master keys $Mkey_1$ and $Mkey_2$ and $Pubkey_1, Pubkey_2$ are the public keys. If it is assumed that $\mathbb{A}$ obtains these parameters, then it must find $PVIDs$ where $PVID_1 = \text{nonce}(g_1 \oplus g_2)$ and $PVID_2 = VID \oplus h(\text{nonce.Pubkey}_1 \| \text{secret})$. Here, the $secret$ and nonce $nonce$ are unknown parameters, and guessing both values is not possible.
Repudiation attack	The proposed scheme uses a vector v in the verification method to achieve non-repudiation. Assume that a suspicious vehicle is attempting to disprove the message sign by switching some messages in the batch. If RSCN performs batch verification of several messages to validate them, such changed messages may fail to be validated. As a result, vehicles cannot deny signatures on delivered messages until they have been signed and sent. Non-repudiation is achieved as a consequence.
Man-in-the-middle attack	V_i sends $\langle PVID, M_i, sign_i, \tau_i \rangle$ to the neighbouring RSCN or other connected nodes over a public transmission channel and thus $\mathbb{A}$ can access these parameter along with the public parameter $\{\mathbb{H}_i, \mathbb{H}_j, \mathbb{H}_T, o, g_1, g_2, Pubkey_1, Pubkey_2\}$ and $\{Mkey_1, Mkey_2\}$. However, V_i performs encryption, pairing operations, hashing, XOR operations, and concatenations of secret parameters. Hence, the attacker needs to perform some required calculation to access the message.
Location privacy preservation	For each communication, OBU generates a set of psuedo-IDs, which is calculated as: $PVID_1 = nonce.(g_1 \oplus g_2)$ and $PVID_2 = VID \oplus h(nonce.Pubkey_1 \| secret)$. When a message is sent to the other components of IoV, each message contains these $PVIDs$ that are generated using a one-way hash function and the randomly generated $secret$ key. Hence, it is quite challenging for an adversary to access VID because it is calculated as: $VID = PVID_2 \oplus h(nonce.Pubkey_1 \| secret)$. Thus, vehicles' location privacy is preserved using the proposed scheme.

Further, the proposed scheme is compared with the existing state-of-the-art schemes[189, 122, 123, 190, 61, 47] for security. Table 3.14 shows the security comparison of the proposed scheme with the existing schemes. Some attack vectors and features are abbreviated as follows: S1: Batch message verification; S2: Replay attack; S3: Man-in-the-middle attack; S4: Impersonation; S5: Non-repudiation of single message; S6: Non-repudiation of batch message; S7: Tracing; S8: Collision induction attack. The symbols used are: ✓: Secure; ×: Vulnerable.

Table 3.14: Security comparison of proposed scheme with state-of-the-art schemes

Attacks	[189]	[122]	[123]	[190]	[187]	[47]	Ours
S1	✓	✓	✓	✓	×	✓	✓
S2	×	✓	✓	×	✓	✓	✓
S3	✓	×	×	NA	×	×	✓
S4	✓	×	×	NA	×	×	✓
S5	×	✓	×	NA	✓	✓	✓
S6	×	×	×	×	✓	✓	✓
S7	×	×	×	✓	✓	✓	✓
S8	NA	NA	NA	NA	NA	NA	✓

3.2.3 Performance Analysis and Simulation Results

The performance of any validation or verification method can be measured as computation cost. The different calculations in the scheme include point multiplication over an elliptic curve, M2P hash function, one-way hash function, and pairing operation. The obtained value of the average execution time for these cryptographic operations are notated as T_m, T_{M2P}, T_h, and T_p respectively. The execution time for each cryptographic operation is given in Table 3.8. Three performance parameters for the proposed scheme are introduced: computation cost, time complexity, and transmission overhead. These parameters help to analyze and evaluate the proposed methodology's efficiency.

Table 3.15: Cryptographic equations for the proposed scheme and state-of-the-art schemes

Schemes	Single verification	Batch verification
Boneh et al.[189]	$4T_p + 2T_{M2P}$	$(2n+2)T_p + 2nT_{M2P}$
Zhang et al.[122]	$3T_p + T_{M2P} + T_m$	$3T_p + nT_{M2P} + nT_m$
Lee et al.[123]	$3T_p + T_m$	$3T_p + T_m$
Liu et al.[61]	$T_p + 5T_{M2P}$	NA
Pournaghi et al.[190]	$3T_p + T_{M2P} + T_m$	$3T_p + nT_{M2P} + nT_m$
Bayat et al.[47]	$3T_p + T_{M2P} + T_m$	$3T_p + nT_{M2P} + nT_m$
Ours	$2T_p + 2T_m$	$2T_p + 2T_m$

computation cost: The execution time for each cryptographic operation, given in Table 3.8, is used to calculate the computation cost for the proposed scheme and existing schemes. The scheme is compared with the existing schemes[189, 122, 123, 61, 190, 47]. The cryptographic equations using total number of operations used in both single and batch message verification are given in Table 3.15. For the proposed

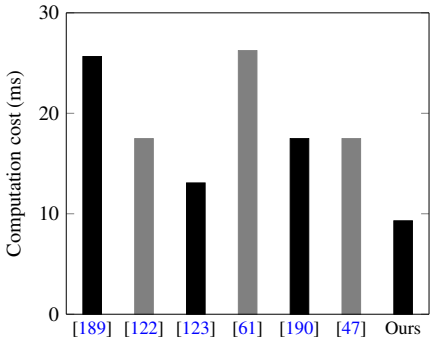


Figure 3.9: Comparison of computation cost (ms) between proposed scheme and state-of-the-art schemes

verification scheme, it is provided that the time to verify n signatures from the batch verification equation is $2T_p + 2T_m$. The computation cost for the proposed scheme is calculated as 9.306 ms ($2T_p + 2T_m = 2 \times 4.211 + 2 \times 0.4420 = 9.306$). Similarly, the computation cost for [189], [122], [123], [61], [190], and [47] are obtained as 25.66 ms, 17.48 ms, 13.07 ms, 26.24 ms, 17.48 ms, and 17.48 ms respectively. Figure 3.9 depicts the comparison of computation cost between the proposed scheme and the existing schemes. In Table 3.15, it is observed that the verification time to verify n number of signatures using the batch verification equation is $2T_p + 2T_m$ for the proposed scheme, which is not affected by n(i.e., batch size).

Time complexity: The time complexity of various verification schemes are discussed in Table 3.16. The signature verification complexity for the proposed method is $O(1)$, which is similar to Lee et al. schemes. For the rest of the schemes, the time complexity for signature verification is $O(n)$. The time complexity for signature generation is the same for all the schemes, which is $O(n)$. Despite being similar in terms of time complexity, the Lee et al. method's computation cost is more than the proposed scheme. Therefore, the proposed batch verification method is more efficient and effective in both signature generation and signature verification.

Table 3.16: Time complexity comparison

Schemes	Signature verification
Zhang et al.[122]	$O(n)$
Bayat et al.[47]	$O(n)$
Boneh et al.[189]	$O(n)$
Pouranghi et al.[190]	$O(n)$
Lee et al.[123]	$O(1)$
Proposed Scheme	$O(1)$

Transmission overhead: It is assumed that $\mathbb{H}_1$ and $\mathbb{H}_2$ are represented by 128 bits. Accordingly, each point of these groups is represented by 21 bytes. The SHA-1 hash function, which has a 160-bit (20 bytes) representation, is used. Consequently, the lengths of $PVID_1$ and $PVID_2$ are 21 bytes and 20 bytes, respectively, using the SHA-1 hash function. The signature $sign_i$ is 21 bytes. Thus, the overall transmission overhead for a signed message transferred by a vehicle is 62 bytes. The state-of-the-art schemes adopted for the comparison of computation cost do not provide sufficient data for transmission overhead comparison. Hence, other state-of-the-art schemes[122, 189, 124, 198, 103] that provide a detailed explanation for the transmission overhead are considered. Table 3.17 describes the transmission overhead for all the existing schemes along

Table 3.17: Transmission overhead comparison of proposed scheme with state-of-the-art schemes

Schemes	Transmission overhead
Zhang et al.[122]	21 bytes + 42 bytes
Boneh et al.[189]	21 bytes + 125 bytes
Tzeng et al.[124]	21 bytes + 45 bytes
Huang et al.[198]	40 bytes + 40 bytes
Malhi et al.[103]	20 bytes + 128 bytes
Ours	21 bytes + 41 bytes

with proposed scheme. The transmitted packet size for [122], [189], [124], [198], and [103] are 63 bytes, 146 bytes, 67 bytes, 80 bytes, 148 bytes, respectively. It demonstrates that the suggested batch verification scheme has the lowest transmission overhead of all the schemes.

Energy consumption: In message exchange, some energy is consumed to perform all the calculations. This consumption can affect the performance of the technique. The formula to calculate energy consumption is:

$$Energy = Total_{CC} \times Power_{max},$$

Where, $Total_{CC}$ is the total computation cost and $Power_{max}$ is the maximum processing power. For the wireless transmission, max processing power is considered 10.88 W[179]. The computation cost given in

Table 3.15 for Desktop system are used to calculate the consumed energy. Table 3.18 shows the calculated energy consumption for the proposed scheme and other state-of-the-art schemes.

Table 3.18: Energy consumption (mJ) for different schemes

Schemes	Energy consumption
Boneh et al.[189]	279.14
Zhang et al.[122]	190.19
Lee et al.[123]	142.25
Liu et al.[61]	285.50
Pournaghi et al.[190]	190.19
Bayat et al.[47]	190.19
Proposed Scheme	101.25

3.3 DYNAMIC BATCH VERIFICATION

In 2019, there were 6.8 million police-reported crashes resulting in over 36,000 deaths and 2.7 million injuries[199], emphasizing the urgency for efficient and quick real-time IoV communication to decrease traffic congestion and accidents. The increase in vehicle density leads to communication lags, despite a reserved 5.9 GHz band for transportation communications[199] and the DSRC protocol's requirement for vehicles to broadcast Basic Alert Messages (BAMs) every 100-300ms[200]. These BAMs often go unverified within the allotted timeframe and are consequently discarded, which is a critical issue in dense traffic scenarios where reducing message loss and verification delays is critical. Existing batch message verification schemes[127, 128, 129, 130, 131] are predicated on the assumption that BAMs have a 500ms window, drivers react within 700ms, which are time-consuming, may lead to message drops. However, their major limitation is the use of fixed batch sizes and lack of BAMs' prioritization, which can result in unverified emergency messages or delayed detection of malicious users in larger batches[55, 126].

This thesis presents a Message Prioritization and Priority-driven Dynamic Batch Verification named as DyBatch scheme devised to reduce verification delays and communication overhead. DyBatch prioritizes the BAMs using a flag-based mechanism, adjusts batch sizes to streamline message verification, and transform RSCNs into intelligent Edge Nodes (ENs), enhancing the efficiency of message verification in Edge-computing enabled IoV (e-IoV's). The following are the key contributions of DyBatch:

1. **Adaptability and responsiveness using dynamic batch adjustment:** DyBatch utilizes a real-time adaptive batching algorithm, dynamically adjusting batch sizes based on current network conditions and message priority levels. The mechanism operates on principles derived from dynamic system theory and real-time computing, ensuring that critical messages are processed with minimal delay. By continuously monitoring network traffic and message queues, DyBatch adapts batch sizes to improve the responsiveness, reducing the likelihood of message drops and verification delays.

2. **Real-time priority classification using flag-Based mechanism:** The flag-based priority classification mechanism in DyBatch assigns flags to messages based on their urgency, using a combination of content analysis and predefined priority rules. High-priority messages are tagged with a 'High Priority (HP)' flag, ensuring they are processed ahead of lower-priority messages. This classification

system is integrated into the batch verification process, allowing for efficient prioritization and fast-tracking of critical messages. The approach leverages real-time data analytics and machine learning techniques to dynamically adjust priority levels based on evolving network conditions.

3. **Distributed verification framework using Verification-Proxy Vehicles (VPVs):** The distributed verification framework involves the deployment of VPVs, which assist in verifying messages when the verification requests on ENs are overflow. VPVs are selected based on their computational capacity and geographic proximity to the message source. The framework uses distributed computing principles to parallelize the verification process, reducing bottlenecks and enhancing scalability. By distributing the verification load across multiple VPVs, DyBatch ensures that the network can handle high message volumes without significant delays.

4. **Optimized resource allocation using VPV selection algorithm:** The VPV selection algorithm optimizes resource allocation by balancing the computational load and geographic distribution of verification tasks. The algorithm calculates the optimal number of VPVs needed to handle the current message load, taking into account factors such as processing power, network latency, and traffic density. By dynamically adjusting the number of VPVs and their allocation, the algorithm ensures efficient use of network resources, minimizing verification delays and enhancing overall system performance.

3.3.1 Network and Delay Model

The network model of DyBatch adopts a comprehensive three-layered architecture considering the Vehicular Layer, Edge Computing Layer, and Cloud Layer, as depicted in Figure 3.10. This multi-tier design

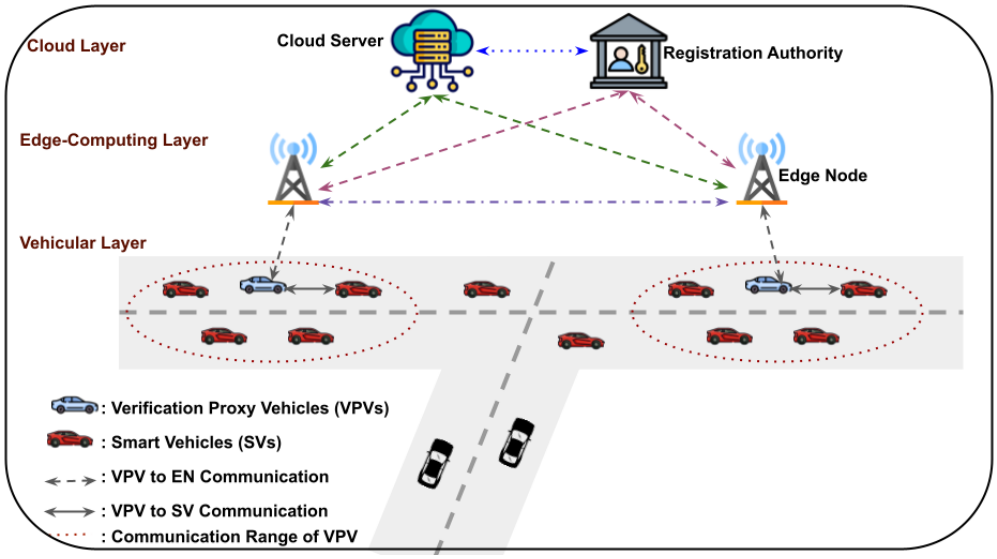


Figure 3.10: Network architecture of DyBatch for e-IoV networks

integrates intelligent edge-computing features to create a distributed enhanced e-IoV environment, thereby enhancing both the efficiency and robustness of vehicular communications and data processing. The vehicular layer primarily consists of Smart Vehicle (SV), which are equipped with advanced capabilities for sensing, communication, computation, and data storage. Each vehicle in this layer is outfitted with a TPD integrated with an OBU[201], ensuring secure data handling and robust communications. Additionally, this

layer includes VPVs that play a crucial role in assisting ENs by reducing the verification delay during the verification process. Specific ENs select a subset of VPVs within their communication range to distribute verification tasks, thereby reducing the computational load on ENs[45]. In the DyBatch architecture, it is assumed that the communication between OBUs and TPDs is secure, ensuring the integrity and confidentiality of the transmitted data.

The edge-computing layer consists of RSCNs that are converted into the intelligent edge nodes having superior storage, computational, and communication capabilities compared to vehicular nodes. These ENs transform the system by providing mechanisms for computation offloading and data caching, crucial for managing data-intensive operations typical in IoV environments. ENs are responsible for collecting data from SV s and VPVs, processing this data to extract relevant information, and forwarding the processed data to the cloud server. By performing these tasks locally, ENs significantly reduce the data processing burden on the cloud, mitigate potential bottlenecks, and enhance overall system performance and reliability. The cloud layer considers a centralized cloud server that offers high computational power and extensive storage capabilities, far better than edge and vehicular nodes[110]. This server has global access to the information aggregated from ENs, SV s, and VPVs, allowing for comprehensive data analysis and rapid decision-making processes. The central role of the cloud server includes not only storing vast amounts of data but also performing complex computations beyond the capacity of edge and vehicular nodes[77]. Additionally, the cloud server facilitates global coordination and management of the IoV network, ensuring that all components operate efficiently. The DyBatch network model, with its hierarchical structure and distributed computing approach, effectively addresses the challenges of data verification, processing delays, and system scalability in the IoV network.

A network delay model is crucial for simulating real-world network conditions, enabling accurate assessments of system performance metrics such as latency and throughput[104]. In the dynamic and dense e-IoV network, BAMs must be verified within a constrained time frame. The DyBatch scheme aims to reduce network delay, directly related to the number of selected VPVs within an EN's range and the batch size designated for VPVs. Verification of BAMs from all SV s in the e-IoV network is executed by ENs and VPVs within a specified range. Initially, VPVs are responsible for verifying the BAMs sent by SV s in their range, conducting the verification in batches. Post batch verification, VPVs compile a list of verified messages and forward it to the respective ENs. Subsequently, ENs filter out unverified BAMs due to out-of-range issues and verify the remaining BAMs directly. This entire process of BAM verification by VPVs and ENs occurs simultaneously. The delay model for the DyBatch assumes that the verification delay for a VPV, denoted as $\mathbb{T}_{VPV_i}$ where $i \in [1,k]$, and for an EN, denoted as $\mathbb{T}_{EN_j}$ where $j \in [1,n]$, contribute to the total e-IoV system verification delay. This is represented by the equation:

$$\mathbb{T}_{e-IoV} = \mathbb{T}_{EN(vpv)} + \max\{\mathbb{T}_{VPV_1}, \mathbb{T}_{VPV_2}, \ldots, \mathbb{T}_{VPV_k}, \mathbb{T}_{EN}\}. \tag{3.20}$$

The delay for EN includes the verification delay $\mathbb{T}_{EN}$ from direct BAMs received from SV s and the computation cost $\mathbb{T}_{EN(vpv)}$ for validating the list of verified BAMs sent by VPVs, calculated as:

$$\mathbb{T}_{EN} = \mathbb{T}^{sv}_{comm} + \mathbb{T}^{EN}_{comp}. \tag{3.21}$$

If N_{EN} represents the number of BAMs transmitted directly from SV s to ENs and verified by ENs, then $\mathbb{T}_{EN}$ can be further detailed as:

$$\mathbb{T}_{EN} = \mathbb{T}^{sv}_{comm} + \mathbb{T}_{\alpha} N_{EN} + \mathbb{T}_{\beta}, \tag{3.22}$$

Here, $\mathbb{T}_{\alpha}$ is the computation cost coefficient related to the BAMs in the dynamic batch verification process, and $\mathbb{T}_{\beta}$ signifies the computation cost for two dot products in the verification process. The batch verification delay $\mathbb{T}_{EN(vpv)}$ on EN, when verifying BAMs forwarded by VPVs, is formulated as:

$$\mathbb{T}_{EN(vpv)} = \mathbb{T}_{\alpha} k + \mathbb{T}_{\beta}, \tag{3.23}$$

Where k denotes the number of VPVs within a specific EN's range.
The verification delay for VPVs includes the communication delay $\mathbb{T}^{sv}_{comm}$ for transmitting BAMs from SV s to VPVs, the computation cost $\mathbb{T}^{vpv}_{comp}$ for verifying BAMs at VPVs, and the communication delay $\mathbb{T}^{vpv}_{comm}$ for sending verified BAMs lists to ENs, given by:

$$\mathbb{T}_{VPV_i} = \mathbb{T}^{sv}_{comm} + \mathbb{T}^{vpv}_{comp} + \mathbb{T}^{vpv}_{comm}. \tag{3.24}$$

Assuming that N_i BAMs are verified in batch by VPV_i, the computation cost $\mathbb{T}^{vpv}_{comp}$ is calculated as:

$$\mathbb{T}^{vpv}_{comp} = \frac{\Phi_{EN}}{\Phi_{vpv}} (\mathbb{T}_{\alpha} N_i + \mathbb{T}_{\beta}), \tag{3.25}$$

Where Φ_{EN} and Φ_{vpv} denote the computing power of EN and VPV nodes, respectively.
For calculating the communication delay of the transmitted list of verified BAMs from VPV to EN, it is assumed that the data communication rate is δ_{comm} and the length of pseudo ID $PVID$ is $\mathbb{L}_{PVID}$. Additionally, data length for timestamps, signatures, pseudo IDs for SV, and other information is considered fixed and denoted as $\mathbb{L}_{other}$. The data length of the list of verified N_i BAMs can be calculated as $[\mathbb{L}_{PVID} N_i + \mathbb{L}_{other}] \backslash \delta_{comm}$.
Here, the communication delay of PVIDs is defined as:

$$\mathbb{T}_{\mu} = \mathbb{L}_{PVID} \backslash \delta_{comm}, \tag{3.26}$$

and communication delay for other data as:

$$\mathbb{T}_{\nu} = \mathbb{L}_{other} \backslash \delta_{comm}. \tag{3.27}$$

Hence, the total communication delay in transmitting the list of verified BAMs from VPVs to EN is simplified as follows:

$$\mathbb{T}^{vpv}_{comm} = \mathbb{T}_{\mu} N_i + \mathbb{T}_{\nu}. \tag{3.28}$$

By simplifying the Eq. (3.24), (3.25), and (3.28), the total verification delay for transmitting the verified BAMs list from VPVs to EN is summarized as:

$$\mathbb{T}_{VPV_i} = \mathbb{T}^{sv}_{comm} + \frac{\Phi_{EN}}{\Phi_{vpv}} (\mathbb{T}_{\alpha} N_i + \mathbb{T}_{\beta}) + \mathbb{T}_{\mu} N_i + \mathbb{T}_{\nu}. \tag{3.29}$$

3.3.2 Proposed Method: Dynamic Batch Verification (DyBatch)

DyBatch provides the solution for the high message loss and high verification delay occur due to high traffic density. DyBatch uses the concept of dynamic batch verification and improves the efficiency, throughput, and latency of e-IoV. The proposed protocol is described in three phases: flag-based message prioritization and batch formation, selection of VPVs, and priority-driven dynamic batch verification.

Message prioritization and batch formation: When a large number of BAMs encounter at VPVs or ENs and need to be verified within a limited time frame, there may be a situation in which these BAMs may not be verified before the expiration of BAMs. Keeping in mind this issue, the focus is on efficiently managing message traffic at VPVs and ENs. The core idea revolves around classifying messages into two categories:

Algorithm 6 Flag-based message prioritization and priority-driven batch formation

Input: A queue of messages with content and type (Emergency or Ordinary)
Output: High priority batch having flag 'HP', low priority batch having flag 'LP'.
Procedure: CreateMsg(content, type)
$msg \leftarrow$ new message($content$)
if type = "Emergency" **then**
    $msg.flag \leftarrow$ "HP"
else
    $msg.flag \leftarrow$ "LP"
end if
return msg
End Procedure
Procedure: ProcessQueue(queue)
$HP_{Batch} \leftarrow$ new Queue()
$LP_{Batch} \leftarrow$ new Queue()
for each msg **in** $queue$ **do**
    if $msg.flag =$ "HP" **then**
        $HP_{Batch}.enqueue(msg)$
    else
        $LP_{Batch}.enqueue(msg)$
    end if
end for
ProcessBatch(HP_{Batch})
ProcessBatch(LP_{Batch})
End Procedure
Procedure: ProcessBatch(batch)
while not $batch.isEmpty()$ **do**
    Batch Verification (Note: High priority batch will be verified earlier than low priority batch will be processed.)
end while
End Procedure

'Emergency' and 'Ordinary'. This classification is crucial as it allows for prioritized processing of critical messages, essential in IoV scenarios where timely delivery of certain information is very important. Algorithm 6 defines the step-by-step procedure for message classification and batch formation. Initially, when a message is created, the flag associated with it is generated either as 'Emergency' or 'Ordinary'. This flag are generated based on the content and urgency of the message. The 'CREATEMSG' procedure encapsulates this process. It takes the content and type as input and returns a message object flag accordingly. The core of the algorithm lies in the 'PROCESSQUEUE' procedure. The procedure takes a queue of messages and segregates them into two separate batches based on their tags. Emergency messages are enqueued into the high priority queue 'HP_Batch', while 'Ordinary' messages go into the low priority Batches 'LP_Batch'. This segregation is crucial as it sets the stage for prioritized processing. Once segregated, the batches are processed separately by the 'PROCESSBATCH' procedure. Priority is given to the emergency batch, which has the tag 'HP' to ensure these messages are dealt with promptly. The 'Ordinary' batch is processed subsequently. This step ensures that while emergency messages are fast-tracked, 'Ordinary' messages are not neglected. The responsibility of classifying messages and forming batches isn't covered here.

Selection of verification-proxy vehicles: In a highly dynamic e-IoV environment with high traffic conditions, verifying BAMs on ENs may lead to a bottleneck issue. If BAMs are not verified within a time frame, critical BAMs will be dropped, affecting the entire system's safety. For this specific high-traffic scenario, DyBatch proposes an idea of proxy vehicles VPVs that assist ENs in the BAMs verification process. The issue with the above idea is to decide the suitable number of VPVs in the range of specific EN. DyBatch provides the solution for this issue. The total network delay is shown in Eq. (3.20), where it is stated that the number of VPVs affects the overall network delay. For example, if the selected VPVs increase, the number of BAMs to be verified at each VPV will decrease. Consequently, the BAM's verification delay for VPV will automatically decrease. On the other hand, the computation costs $\mathbb{T}_{EN(vpv)}$ on EN for the verified BAMs list sent by VPVs will increase. Its vice versa is also possible. So the fair number of VPVs in the range of specific EN plays an important role in optimizing overall network delay. To decide the initial number of VPVs in the range of particular EN, let's consider an ideal network e-IoV'. The network model for e-IoV' is the same as the original one, and the processing power of each SV is the average processing power of SV s in the original e-IoV, represented as: $\phi^{'} = \Sigma_{j=1}^{n}(\Phi_j \backslash n)$. In the e-IoV, VPVs and EN verify BAMs simultaneously. It is assumed that the computing power of each SV is the same, and the computation cost of BAMs verified by VPV is only affected by the number of SVs verified by the VPV. Hence, the number of ordinary SVs verified by each VPV in e-IoV should be the same, represented as $N_{\phi'}$.It is assumed that the number of VPVs is k. From Eq. (3.29), the BAMs verification delay for VPV in e-IoV is:

$$\mathbb{T}_{\Phi'} = \frac{\Phi_{EN}}{\Phi_{'}}(\mathbb{T}_{\alpha}N_{\Phi'} + \mathbb{T}_{\beta}) + \mathbb{T}_{\mu}N_{\Phi'} + \mathbb{T}_{\nu} + \mathbb{T}_{comm}^{sv}. \tag{3.30}$$

According to the Eq. (3.20), the total verification delay for e-IoV can be written as:

$$\mathbb{T}_{e-IoV'} = \mathbb{T}_{EN(vpv)} + max\{\mathbb{T}_{\Phi'}, \mathbb{T}_{\Phi'}, ...\mathbb{T}_{\Phi'}, \mathbb{T}_{EN}\}. \tag{3.31}$$

The aim is to optimize the verification delay of the overall network, so the verification delay on VPVs will be the smallest only when $\mathbb{T}_{\Phi'} = \mathbb{T}_{EN}$, which is $\mathbb{T}_{\alpha}N_{EN} + \mathbb{T}_{\beta} + \mathbb{T}_{comm}^{sv}$. And the total optimized verification delay for overall e-IoV network is as follows:

$$min\{\mathbb{T}_{\alpha}k + \mathbb{T}_{\beta} + \mathbb{T}_{\alpha}N_{EN} + \mathbb{T}_{\beta} + \mathbb{T}_{comm}^{sv}\}. \tag{3.32}$$

Further, the total number of SVs in the system is the sum of the number of SVs verified by VPVs and the number of SVs verified by EN, calculated as follows:

$$N = kN_{\Phi'} + N_{EN}. \tag{3.33}$$

By simplifying Eq. (3.32) and Eq. (3.33), the optimized number of VPVs in the e-IoV system is obtained as follows:

$$k = \frac{\sqrt{(\mathbb{T}_{\alpha}\Phi'N + \mathbb{T}_{\beta}\Phi' - \mathbb{T}_{\beta}\Phi_{EN} - \mathbb{T}_{\nu}\Phi')(\mathbb{T}_{\alpha}\Phi_{EN} + \mathbb{T}_{\mu}\Phi')}}{\mathbb{T}_{\alpha}\Phi'} - \frac{(\mathbb{T}_{\alpha}\Phi_{EN} + \mathbb{T}_{\mu}\Phi')}{\mathbb{T}_{\alpha}\Phi'}. \tag{3.34}$$

The number of SV s verified by each VPVs is calculated as follows:

$$N_{\Phi'} = \sqrt{\frac{(\mathbb{T}_{\alpha}\Phi'N + \mathbb{T}_{\beta}\Phi' - \mathbb{T}_{\beta}\Phi_{EN} - \mathbb{T}_{\nu}\Phi')}{(\mathbb{T}_{\alpha}\Phi_{EN} + \mathbb{T}_{\mu}\Phi')}}. \tag{3.35}$$

Dynamic batch verification: Although researchers[127, 128, 129, 130, 131, 55] proposed some efficient batch verification algorithms for the IoV communication environment, these algorithms have a significant drawback of the fixed batch size resulting in a higher message drop rate that is not desirable for the real-time communication system keeping in mind the safety of the human being involved. The above issue can be explained with a scenario where let's suppose, the batch size is larger, having emergency BAMs; it may lead to the message drop due to the limited duration of expiration of BAMs. Similarly, if the batch size is smaller, having a ordinary BAMs, some VPVs and ENs will usually go into the idle state while other ENs and VPVs face the bottleneck issue. To avoid such issues, DyBatch prioritizes the BAMs according to their generated flags based on the criticality of the content and forms batches according to the priority of the BAMs. Minimizing delay is the primary optimization objective. To achieve this, the focus extends beyond just the security robustness of the chosen encryption-decryption algorithm; it also prioritizes ensuring that the computation cost is relatively minimal. This is crucial as different encryption and decryption methods come with varying levels of computational overhead. Therefore, selecting an encryption and decryption algorithm that not only provides high security but also incurs lower computation cost is essential to meet this goal. DyBatch uses the ECC algorithm for the highly dynamic real-time e-IoV environment due to its low security-efficiency trade-off. Along with the ECC algorithm, one-way hash functions are used to maintain the integrity of the BAMs. The execution time for a one-way hash function is significantly less than that of the mainly used M2P hash functions. The variables used in the DyBatch methods and their descriptions are given in Table 3.19. The dynamic batch message verification is explained as follows:

Table 3.19: Variables used in DyBatch and their descriptions

Variables	Descriptions
p, q	Prime numbers used in the cryptographic system.
a, b	Coefficients of the elliptic curve equation.
P	Base point on the elliptic curve.
$Pubkey$	Public key of the system.
$h()$	Cryptographic hash function.
V_j	SV with index j.
$PVID_1^j, PVID_2^j$	Pseudo IDs generated for V_j.
x_j	Random number generated by TPD, belongs to Z_p^*.
VID_j	Vehicular identity of V_j.
$sign_j$	Signature generated by TPD for BAM_j.
$Prikey$	Private key of the system.
τ_j	Current timestamp.
BAM_j	Basic Alert Message generated by V_j.
VPV_i	Verification-Proxy Vehicle with index i.
EN	Edge Node in the IoV network.
B_j	Batch size for the j^{th} set of messages.
f_j	Random factor used in batch verification, a small integer.
g	Small integer determining the range of f_j.

- **System initialization:** In the initial stage, the system establishes the environment by generating global public parameters which underpin the ECC operations. These parameters include two large prime numbers p and q, where q divides $p-1$, coefficients a and b that define the elliptic curve equation, and a base point P on the curve. Additionally, a secure hash function h (e.g., SHA-256) is chosen for hashing operations. Each participant, whether a SV or an EN, generates a unique cryptographic key pair: a private key ($Prikey$) for securing operations and a corresponding public key ($Pubkey = Prikey \cdot P$) for public verification.

- **Secure message generation and transmission:** During this phase, the sender vehicle, V_j undertakes several cryptographic steps to prepare a message for secure transmission. The detailed procedure of secure communication using DyBatch is described in the Algorithm 7.
  - Pseudonym generation: For enhancing privacy, each message is tagged with pseudonyms ($PVID_1^j$ and $PVID_2^j$) derived from ECC operations and hashing, respectively. These pseudonyms mask the true identity of the sender.
  - Message signing: The message, alongside its metadata, is signed using the sender's private key to generate $sign_j$, ensuring the message's integrity and authenticity.
  - Message encryption: The message is encrypted using the recipient's public key to ensure confidentiality during transit.
  - Broadcasting: The prepared message, including its pseudonyms, encrypted content, and signature, is broadcast to the intended recipient.

- **Dynamic batch verification:** The verification process involves aggregating signatures of multiple messages and verifying their collective authenticity and integrity efficiently. This is accomplished through a weighted sum of the signatures and corresponding message hashes, using randomly chosen small integers, f_j as coefficients. The messages considered to be verified, if the following equation is satisfied. Otherwise, the messages are considered as corrupt messages and hence discarded.

$$\text{Verify} \left(\sum_{j=1}^{B_j} f_j \cdot sign_j \right) \cdot P = \left(\sum_{j=1}^{B_j} f_j \cdot h(PVID_1^j \parallel PVID_2^j \parallel BAM_j \parallel \tau_j) \right) \cdot Pubkey. \tag{3.36}$$

Proof: Starting from the definition of $sign_j$,

$$sign_j = Prikey \cdot h(PVID_1^j \parallel PVID_2^j \parallel BAM_j \parallel \tau_j). \tag{3.37}$$

Multiplying both sides by f_j and summing over all B_j messages, the result is obtained as

$$\sum_{j=1}^{B_j} (f_j \cdot sign_j) = \sum_{j=1}^{B_j} (f_j \cdot Prikey \cdot h(PVID_1^j \parallel PVID_2^j \parallel BAM_j \parallel \tau_j)). \tag{3.38}$$

Now, multiplying both sides by the base point P, and recalling that $Pubkey = Prikey \cdot P$, the result is obtained as:

$$\left(\sum_{j=1}^{B_j} f_j \cdot sign_j \right) \cdot P = \left(\sum_{j=1}^{B_j} f_j \cdot Prikey \cdot h(PVID_1^j \parallel PVID_2^j \parallel BAM_j \parallel \tau_j) \right) \cdot P. \tag{3.39}$$

Simplifying further using the distributive property of scalar multiplication over addition in ECC and the definition of $Pubkey$,

$$= \left(\sum_{j=1}^{B_j} f_j \cdot h(PVID_1^j \parallel PVID_2^j \parallel BAM_j \parallel \tau_j) \right) \cdot Pubkey. \tag{3.40}$$

Algorithm 7 DyBatch secure communication procedure

Input: Number of vehicles N, Public parameters (p,q,a,b,P,h), VID_j, BAM_j, Timestamps τ_j, Keys $(Key_{pri_j}, Key_{pub_{rec}}, Prikey, Pubkey, Key_{pri_{SV}}, Key_{pub_{VPV}})$, Nonce n_{VPV}

Output: $\{PVID_1^j, PVID_2^j, Encrypted, sign_j\}$, Authentication status, Verification results

System Initialization:
Generate public parameters: p,q,a,b,P,h.
for each SV and EN **do**
    Generate $Prikey$ and $Pubkey = Prikey \cdot P$.
end for
Secure Message Generation and Transmission:
Function GeneratePseudonyms(V_j)
$x_j \leftarrow$ random element in $\mathbb{Z}_p^*$.
$PVID_1^j \leftarrow x_j \cdot P$.
$PVID_2^j \leftarrow VID_j \oplus h(x_j \cdot Pubkey)$.
End Function
Function SignMessage(BAM_j, τ_j, Key_{pri_j})
$sign_j \leftarrow Key_{pri_j} \cdot h(PVID_1^j \| PVID_2^j \| BAM_j \| \tau_j)$.
return $sign_j$.
End Function
Function EncryptMessage(BAM_j, τ_j, $Key_{pub_{rec}}$)
$Encrypted \leftarrow \text{AES}_{Key_{pub_{rec}}}(BAM_j \| \tau_j)$.
return $Encrypted$.
End Function
Function Broadcast($PVID_1^j$, $PVID_2^j$, $Encrypted$, $sign_j$)
Broadcast $\{PVID_1^j, PVID_2^j, Encrypted, sign_j\}$.
End Function
Dynamic Batch Verification with Secure Authentication:
Function DecryptMessage($Encrypted$, $Prikey$)
$Decrypted \leftarrow Dec_{Prikey}(Encrypted)$.
return $Decrypted$.
End Function
Function MutualAuthentication(n_{VPV}, $Key_{pri_{SV}}$, $Key_{pub_{VPV}}$)
$resp_{SV} \leftarrow h(n_{VPV} \| Key_{pri_{SV}})$.
Send $resp_{SV}$ to receiver.
receiver computes $expected_{resp} \leftarrow h(n_{VPV} \| Key_{pub_{SV}})$.
if $resp_{SV} = expected_{resp}$ **then**
    Authentication success.
else
    Authentication fail.
end if
End Function
Verification Process:
Function VerifyBatch($\{sign_j\}$, $\{BAM_j\}$, $\{\tau_j\}$, $Pubkey$)
for each j in batch **do**
    Compute $h(PVID_1^j \| PVID_2^j \| BAM_j \| \tau_j)$.
    Generate random f_j.
end for
Compute $\Sigma_j f_j \cdot sign_j$ and $\Sigma_j f_j \cdot h(\ldots)$.
Verify equation holds with $Pubkey$.
End Function

3.3.3 Security Analysis

This section discusses in detail why an attack is not feasible against the proposed DyBatch mechanism. Table 3.20 shows the security comparison of the DyBatch with existing state-of-the-art methods[127, 128, 129, 130, 131, 55]. The comparison shows that DyBatch is resilient against replay attack, modification, repudiation, impersonation, location tracing while providing confidentiality, anonymity, authentication, forward and backward secrecy. The followings are the detailed discussion to prove the robustness of DyBatch against the mentioned attacks:

Confidentiality: In DyBatch, confidentiality is achieved through symmetric encryption using AES, where each message BAM_j is encrypted with the recipient's public key, $Pubkey_{rec}$:

$$Encrypted \leftarrow \text{AES}_{Pubkey_{rec}}(BAM_j \| \tau_j).$$

Anonymity and unlinkability: Anonymity and unlinkability are achieved via the pseudonym mechanism:

$$PVID_1^j \leftarrow x_j \cdot P.$$
$$PVID_2^j \leftarrow VID_j \oplus h(x_j \cdot Pubkey).$$

The first pseudonym, $PVID_1^j$, provides anonymity by using ECC to transform a random scalar x_j without revealing the actual identity. The second pseudonym, $PVID_2^j$, uses a hash-based transformation XORed with the vehicle's identifier, enhancing unlinkability as x_j shields the real identity. An adversary cannot link $PVID_2^j$ to $PVID_1^j$ or the actual identity without the private key associated with $Pubkey$, under the assumption that the ECC discrete logarithm problem is hard.

Table 3.20: Security comparison of DyBatch with the state-of-the-art schemes

Attacks	[127]	[128]	[129]	[130]	[131]	[55]	DyBatch
A1	✓	✓	✓	✓	✓	✓	✓
A2	✓	✓	✓	✓	✓	✓	✓
A3	✓	✓	✓	✓	✓	✓	✓
A4	✓	✓	NA	NA	✓	✓	✓
A5	✓	✓	NA	NA	✓	✓	✓
A6	×	×	NA	NA	✓	✓	✓
A7	✓	×	NA	NA	×	×	✓
A8	✓	×	NA	NA	×	×	✓
A9	×	×	NA	NA	×	×	✓

Note: A1: Replay attack, A2: Modification, A3: Non-repudiation, A4: Impersonation, A5: Location tracing, A6: Stolen OBU, A7: Perfect forward secrecy, A8: Backward secrecy, and A9: System key disclosure.

Mutual authentication: Mutual authentication is implemented using a nonce-based challenge-response mechanism:

$$resp_V \leftarrow h(n_{VPV} \| Prikey_V).$$

receiver computes $expected_{resp} \leftarrow h(n_{VPV} \| Pubkey_V)$ and checks if $resp_V = expected_{resp}$.
The process ensures that both parties prove their identities by demonstrating knowledge of their private keys without transmitting them. The use of a nonce n_{VPV} prevents replay attacks, and the integrity and authenticity of the response are verified through a hash function, assuming the pre-image resistance of h.

Unforgeability: Unforgeability of messages is ensured through digital signatures:

$$sign_j \leftarrow Prikey_j \cdot h(PVID_1^j \| PVID_2^j \| BAM_j \| \tau_j).$$

The signature cannot be forged without access to $Prikey_j$, under the assumption that the ECC-based digital signature scheme (e.g., ECDSA) is secure. The scheme's security relies on the infeasibility of solving the ECC discrete logarithm problem.

Forward secrecy and backward secrecy: Forward secrecy and backward secrecy are provided by regenerating x_j for each session/message and potentially rotating *Prikey* and *Pubkey* periodically or per session. Compromise of long-term keys does not compromise past or future session keys, assuming each x_j is kept secret and discarded securely after use. New session keys ensure that even if a long-term key is compromised, previous communications remain secure (backward secrecy) and future keys cannot be predicted (forward secrecy).

Replay attack: A timestamp is incorporated in every transmission to prevent adversaries from altering the timestamp value. Each time a vehicular node sends data, it incorporates a timestamp element of τ_j. In Dybatch, message freshness can be sustained by employing timestamp computations at the receiving end, following the formula $\tau(receive) - \tau_j < \Delta$, where Δ represents a predetermined threshold value. The value of the threshold is depend upon the communication environment. In a dense, delay-sensitive context such as e-IoV, the threshold value is taken to be marginally greater than the maximum average packet transmission duration (i.e., the time it takes for a message to be sent from the sender to the recipient).

Modification: In a message modification attack, the adversary $\mathbb{A}$ intercepts $sign_j$ and $PVID_j$ for a legitimate BAM_j and attempts to modify the content of BAM_j before sending $sign_j$ and $PVID_{j'}$ in a new BAM. Let's consider the verification equation for the BAM with modified content:

$$\left(\sum_{j=1}^{B_j} f_j \cdot sign_j\right) \cdot P = \left(\sum_{j=1}^{B_j} f_j \cdot h(PVID_1^j \parallel PVID_2^j \parallel modified_BAM_j \parallel \tau_j)\right) \cdot Pubkey, \tag{3.41}$$

Here, $Modified_BAM_j$ represents the content of BAM_j after being modified by $\mathbb{A}$. Since $\mathbb{A}$ cannot possess the legitimate private key *Prikey* used by V_j to generate $sign_j$, $\mathbb{A}$ cannot produce a valid $sign_j$ or $PVID_j$ that would satisfy the verification equation. Specifically, the following mathematical reasons apply:

- $\mathbb{A}$ cannot generate a valid $sign_j$ without knowing the private key *Prikey*. Therefore, the LHS of the equation will not match the RHS for the modified BAM.
- $\mathbb{A}$ cannot produce a valid $PVID_j$ without knowing the corresponding x_j. The secure x_j is required to compute $PVID_j$, and $\mathbb{A}$ does not have access to it. This means the $PVID'_j$ on the RHS will not match the $PVID_j$ computed by $\mathbb{A}$.
- If the content of BAM_j is modified by $\mathbb{A}$, the hash value $h(Modified_BAM_j||\tau_j)$ will be different from the original hash value $h(BAM_j||\tau_j)$, resulting in a mismatch in the equation.

Repudiation: Repudiation occurs when a legitimate sender V_j attempts to deny their involvement in generating a specific BAM_j after the BAM has been broadcast. To prove the prevention of repudiation, the properties of digital signatures and the verification equation are considered.

- A legitimate sender V_j generates a digital signature $sign_j$ using their private key $Prikey$ for a specific BAM_j.
- The verification equation ensures that the signature $sign_j$ corresponds to the pseudo ID $PVID_j$ and the content of BAM_j.
- The verification equation also verifies the legitimacy of BAM_j based on the values of $PVID_j$ and the content of BAM_j.
- As a result, any attempt by V_j to deny their involvement in BAM_j is mathematically implausible because only V_j possesses the private key required to generate $sign_j$.

To deny involvement in BAM_j, V_j would need to repudiate the signature $sign_j$ generated using their private key $Prikey$. However, this is mathematically infeasible because:

- Digital signatures generated with a private key are cryptographically bound to that private key.
- Verifying the signature $sign_j$ against the corresponding public key $Pubkey$ ensures that it was generated using the legitimate private key $Prikey$.
- Therefore, the verification equation confirms the association between the signature $sign_j$, the sender V_j, and the content of BAM_j.

Impersonation: During this attack, $\mathbb{A}$ may transmit a message request to neighboring components and impersonate the identity of a valid user within the network. It can compute fraudulent values and unlawfully dispatch them to other components. Preventive measures can help avoid this attack. In DyBatch, when communicating, exchanged messages carry the pseudo ID $PVID$ instead of the original ID VID_j. $\mathbb{A}$ is unable to determine the VID_j of vehicles, as VID_j is secured in the TPD using XOR operations and hash functions. Consequently, DyBatch remains protected against impersonation attacks.

Location tracing: As a message is conveyed to other IoV components, it includes pseudo ID computed using the formula $PVID_1^j = x_j.P, PVID_2^j = VID_j \oplus h(x_j.Pubkey)$. Access to these IDs is exclusively granted through the timestamp. Consequently, an adversary cannot track the vehicles.Thus DyBatch remains impervious to location tracing attacks.

3.3.4 Performance Analysis and Simulation Results

The performance analysis of DyBatch is provided with a comparison to the state-of-the-art schemes in terms of computation cost, verification delay, storage overhead, and the number of verified BAMs. Some of the recent works considered for comparison include [127], [128], [129], [130], [131], and [55]. In DyBatch, two types of devices are considered for the experiments: NVIDIA Jetson Nano as ENs and Raspberry Pi 4 as SVs and VPVs. Table 3.21 provides detailed specifications of the experimental devices. The Multi-precision Integer and Rational Arithmetic Cryptographic Library (MIRACL) and Java 8 are used to execute different cryptographic operations on Jetson Nano and Raspberry Pi 4 systems. Table 3.22 shows the execution time for each cryptographic operation on the said platform having different environments. In table, T_h, T_p, T_m, T_a, T_e, and T_{sm} represents the execution time for hash functions, pairing operations, ECC-point

Table 3.21: Specifications of NVIDIA Jetson Nano 4 GB and Raspberry Pi 4 4 GB

Feature	Jetson Nano 4 GB	Raspberry Pi 4 4 GB
CPU	Quad-core ARM Cortex-A57 @ 1.43 GHz	Quad-core ARM Cortex-A72 @ 1.5 GHz
GPU	128-core NVIDIA Maxwell architecture-based GPU	None
RAM	4 GB 64-bit LPDDR4 @ 1600 MHz	4 GB LPDDR4-3200 SDRAM
Storage	microSD slot (up to 256 GB), M.2 Key M slot for NVMe SSD	microSD slot (up to 2 TB)
Connectivity	Gigabit Ethernet, 802.11ac wireless, Bluetooth 4.1, HDMI 2.0, DisplayPort 1.2, 4 USB 3.0 ports, CSI camera connector, 3.5 mm audio jack, IR receiver	Gigabit Ethernet, 802.11ac wireless, Bluetooth 5.0, 2 USB 3.0 ports, 2 USB 2.0 ports, 2 micro-HDMI ports (supports up to 4Kp60)
GPIO	Not specified	40-pin GPIO header with support for Raspberry Pi HATs
Software	Supports Ubuntu Linux and NVIDIA JetPack SDK, which includes CUDA, cuDNN, TensorRT, VisionWorks, and other libraries and tools for AI and machine learning	Supports Raspberry Pi OS, Ubuntu, and other Linux-based distributions

Table 3.22: Execution time (ms) of each cryptographic operation on different platforms

Cryptographic operations	Jetson Nano (ENs)	Raspberry Pi 4 (VPVs)
T_h	0.23	1.82
T_p	10.34	35.45
T_m	3.27	7.68
T_a	3.43	10.86
T_e	2.67	10.56
T_{sm}	1.37	4.86

multiplication, ECC-point addition, encryption, and scalar point multiplication. The followings are the detailed discussions for the considered performance measures:

Computation cost : To calculate the computation cost for the DyBatch scheme, the focus is primarily on the time-consuming processes to minimize the execution times. All the cryptographic operations are executed using the MIRACL cryptographic library on the Jetson Nano and Raspberry Pi 4. Table 3.23 shows the cryptographic equations calculated by the different state-of-the-art schemes for single and batch message verification. The cryptographic equation for DyBatch is $T_h + T_m$ for single message verification and $(n+2)T_h + nT_m$ for batch message verification. Using the cryptographic equation given in Table 3.23 and the execution time for each cryptographic operation given in Table 3.22, the total computation cost for the DyBatch scheme is achieved. Figure 3.11a describes the comparison of computation cost for DyBatch and the state-of-the-art schemes on both the Jetson Nano and Raspberry Pi 4 systems.

Table 3.23: Comparison of DyBatch based on the computation cost

Schemes	Single verification	Batch verification
[127]	$T_e + 6T_m + 3T_p$	$nT_m + 3T_p$
[128]	$4T_m + 3T_a + 3T_h$	$(2n+2)T_m + 3nT_a + 3nT_h$
[129]	$3T_m + T_{sm} + T_p + 2T_h$	$2T_m + (T_m + T_{sm} + T_p + 2T_h)n$
[130]	$2T_m$	$(n+4)T_m$
[131]	$4T_p + T_h + 2T_e$	$4T_p + nT_h + 2nT_e$
[55]	$2T_p + 2T_m$	$2T_p + 2T_m$
[132]	$T_p + T_m$	$T_p + nT_m$
[133]	$2T_p + T_m$	$T_p + nT_m + 2(n-1)T_a$
DyBatch	$T_h + T_m$	$(n+2)T_h + nT_m$

Verification delay: In DyBatch, the total verification delay is considered as the initial batch verification delay and the verification delay in re-batch formation. The impact of the number of VPVs on the verification delay is analyzed. Figure 3.11b shows the variation in the verification delay with respect to the number of selected VPVs when the different numbers of SVs are considered as 50, 100, 150, 200, 250, and 300. It is observed that the verification delay first increases and then decreases after a certain period with the increase in the number of VPVs. This is due to the parallel verification of BAMs at VPVs and ENs. Further, Figure 3.11c shows the impact of the number of SVs on the number of optimal VPVs. It is observed that as the number of vehicles in the e-IoV environment increases, the message verification requirement also increases, thus increasing the number of VPVs. The verification delay for DyBatch is also calculated without the selection of VPVs and with the selected VPVs. Figure 3.11d shows the comparison for this analysis. The figure depicts that when the batch verification is done by the ENs only, the verification delay increases drastically with the increasing number of SVs in the e-IoV environment. With the use of VPVs, the verification delay can be reduced to some extent.

Number of verified BAMs: In this section, the impact of dynamic batch verification on the number of verified BAMs is analyzed. Two cases are considered: without re-batch formation and with re-batch formation. Figure 3.11e shows the number of verified BAMs in the case without re-batch formation. It is observed that DyBatch provides a higher number of BAMs compared to other state-of-the-art schemes due to the lower message drop within a specific time frame. The number of verified BAMs is observed to be 126, 130, 204, 420, 340, 380, and 430 for [127], [128], [129], [130], [131], [55], and DyBatch schemes, respectively, at a specific time of 700 ms. The impact of re-batch formation on the number of verified BAMs is also analyzed. Since only Ferng et al.[130] uses the re-batch formation concept, the DyBatch scheme is compared with the existing scheme in [130]. Figure 3.11f depicts the comparison of the number of verified BAMs with respect to time in the case of re-batch formation. It is observed that DyBatch performs significantly better than Ferng et al.'s scheme. At 800 ms, the verified BAMs are 375 and 400 for [130] and DyBatch, respectively.

Simulation Results and Case Study

For the simulation of the DyBatch scheme, the SUMO[202] and NS-2 simulators are used. The .osm file of an urban city in India (Jaipur) is generated from an Open Street Map (OSM). With the help of the SUMO simulator, the generated .osm file is converted to an .xml file, and the simulation is performed using the NS-2 simulator. Heavy traffic road-cross scenarios are considered for the simulation, and at every junction, an edge node is deployed, having a communication range of 1000 m. The communication range for SV is set to 300 m. The data transmission rate for the 802.11p communication protocol is set to 12 Mb/s. The other network parameters are described in Table 3.24. Further, Table 3.25 shows the comparison of the proposed DyBatch scheme with the above-mentioned schemes based on the basic features such as dynamic batch verification, message classification, re-batch formation, edge computing and verifier unit (who is responsible for the verification), etc. The performance of DyBatch is also analyzed in terms of PDR, PLR, latency and jitter, and throughput. The detailed description is provided in the following points:

Packet delivery ratio v/s Packet loss ratio: In IoV, PDR and PLR are important metrics that reflect the reliability and efficiency of the communication system. Achieving a high PDR and low PLR in IoV can be challenging due to the dynamic nature of vehicular networks, which can be affected by factors such as mobility, interference, and network congestion.

$$PDR = (N_R/N_S) * 100\%, \tag{3.42}$$

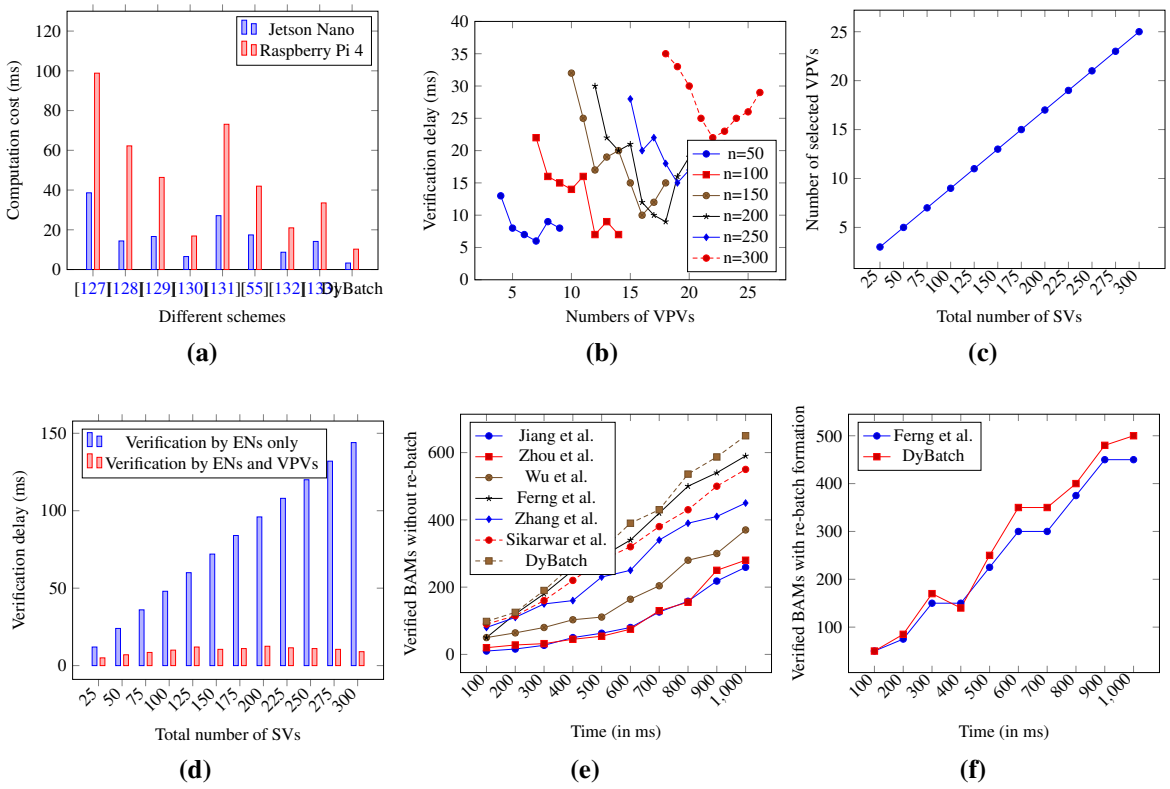


Figure 3.11: Performance evaluation of DyBatch: (a) Comparison of computation cost on different platforms, (b) The impact of the number of VPVs on the overall verification delay, (c) The variations in the number of optimal VPVs in case of increase in the number of SVs, (d) Comparison of verification delay in the case of verification done by EN only, and verification done by both EN and VPVs, (e) Number of verified BAMs for different schemes without re-batch formation, (f) Comparison between DyBatch and Ferng et al.'s scheme based on the number of verified BAMs in case of re-batch formation

Table 3.24: Network parameters and their values

Network parameters	**Value**
Network simulation tool	NS-2
Mobility tool	SUMO
Channel model	two-ray interference model
Simulation area	2000 m x 2000 m
Communication range of EN	1000 m
Communication range of SV	300 m
Communication protocol	IEEE802.11p
Data transmission rate	12 Mb/s
Simulation time	100 seconds
Buffer size for DyBatch	50 packets
Number of simulation	5

$$PDR = (N_L/N_S) * 100\%, \tag{3.43}$$

Here, N_R, N_L, and N_S are the total number of received packets, lost packets, and sent packets. In DyBatch, PDR and PLR values are improved in the e-IoV environment by using an error control mechanism of re-

Table 3.25: Comparison of DyBatch with state-of-the-art schemes based on the different factors

Features	**[127]**	**[128]**	**[129]**	**[130]**	**[131]**	**[55]**	**DyBatch**
Dynamic batch verification	Yes	No	No	Yes	No	No	Yes
Message classification	No	No	No	Yes	No	No	Yes
Re-batch formation	No	No	No	Yes	No	No	Yes
Edge computing	No	No	No	No	No	No	Yes
Verification done by	RSCNs	RSCNs	RSCNs	RSCNs	RSCNs	RSCN	EN & VPVs

batch formation in case of message drops. Figure 3.12a shows the PDR and PLR values (in %) and the total number of sent and received packets.

Latency and jitter: Latency is an important factor in determining the performance of a network. In general, lower latency is better, as it means that data is being transmitted more quickly and efficiently[203]. However, in the case of an e-IoV communication network, all affecting factors, such as the mobility of SVs, network congestion, traffic density, and communication protocols, are considered while calculating the latency of the e-IoV system.

$$Latency = RTT - (PT + QD + TT), \tag{3.44}$$

Here, RTT stands for Round-Trip Time, PT is for Processing Time, QD is for Queueing Delay, and TT is for Transmission Time. Figure 3.12b shows the variation of latency with respect to time for the e-IoV environment. It is observed that the latency of the network varies up to a maximum of 20 ms. Furthermore, for real-time applications, the jitter factor is also considered and analyzed using the standard deviation of the packet delay. Figure 3.12c shows the latency versus jitter for every 10 seconds and the error calculated based on the standard deviation of the latency and jitter.

Throughput: The throughput of the DyBatch scheme for the e-IoV environment is also analyzed, with measurements taken every 10 seconds over a 100-second period, considering the transmission of 208-byte messages. Figure 3.12d shows the data transferred in bytes and the throughput in Kilo-bits/second.

3.4 SUMMARY OF THE WORK

The chapter presents a comprehensive analysis of lightweight authentication and verification schemes using tamper-proof devices for IoV. The focus is on balancing security, privacy, and efficiency in vehicular communications. This chaper introduces three significant contributions: conditional privacy preserving MAC based authentication (Novel MAC-based Authentication Scheme (NoMAS)), static batch verification, and dynamic batch verification. NoMAS addresses the critical need for confidentiality in IoV by employing conditional privacy preservation. This approach secures user identities without compromising system integrity, ensuring that sensitive information is exchanged securely. NoMAS also incorporates hard and soft Key Updates to address the key disclosure problem, guaranteeing continuous key integrity through periodic modifications or complete replacements. The static batch verification scheme optimizes the verification process by allowing multiple messages to be verified simultaneously using lightweight cryptographic operations such as one-way hash functions and XOR operations. This method is particularly effective in scenarios with predictable communication patterns, significantly reducing computational overhead and enhancing network efficiency. Dynamic batch verification builds on the principles of static batch verification and introduces message prioritization. This technique adjusts the verification process based on message

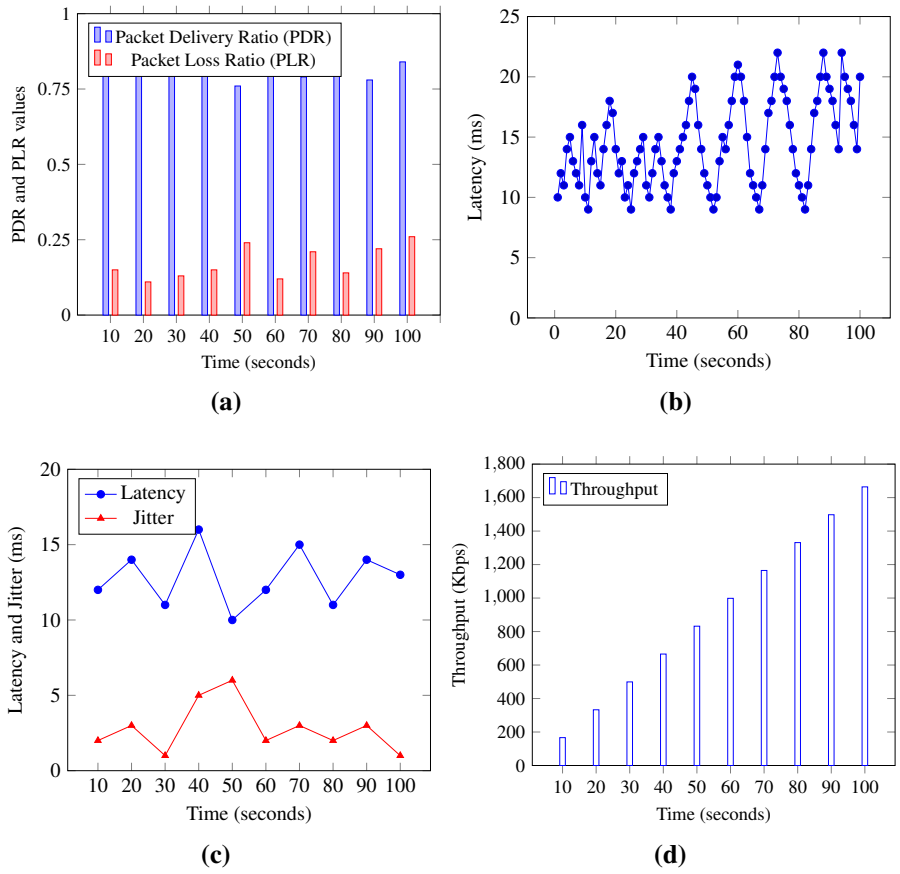


Figure 3.12: Simulation results of DyBatch: (a) Packet delivery ratio v/s Packet loss ratio, (b) Latency analysis for 100 seconds, (c) Latency and Jitter comparison for every 10 seconds interval, (d) Throughput analysis

criticality, ensuring that high-priority messages are authenticated promptly. This approach improves the system's responsiveness and reliability, which is crucial for applications like emergency response and traffic management.

The proposed schemes achieve significant reductions in communication overhead, which is essential for maintaining high-speed communication standards in IoV networks. The schemes require minimal computational resources, making it highly efficient compared to existing state-of-the-art schemes. The security aspect is validated using BAN logic and the ProVerif tool, confirming the robustness of the proposed schemes against potential security threats. Performance analysis and simulation results demonstrate substantial improvements in computation cost, communication overhead, and energy consumption.