5

Lightweight Authentication using Physical Unclonable Functions

In the continuously growing field of vehicular technology, securing IoV networks presents considerable security and privacy challenges due to its highly dynamic and heterogeneous communications (i.e., V2V, V2R, V2I, and V2X, etc.)[99]. Previous chapters of this thesis have explored the deployment of TPD[111] as a foundational security measure. These devices have proven to be highly effective in protecting vehicular networks by ensuring data integrity and preventing unauthorized access. The inherent robustness of tamper-proof mechanisms has thus far provided a substantial layer of security against a wide range of cyber threats[55, 42, 45]. However, as vehicular technology advances, the security frameworks designed to protect it must also evolve. Building upon the established foundation of TPDs, this chapter transitions to examining PUF[104]. In contrast to conventional tamper-proof mechanisms, PUFs introduce an approach to security by exploiting the intrinsic physical variations that arise during semiconductor manufacturing processes. These microscopic differences, which are impossible to replicate even by the original manufacturer, offer a robust means of generating unique cryptographic keys and authenticating devices within the IoV network[80]. PUF works on CRP mechanism where a challenge (a specific input) is applied to a PUF, it produces a corresponding response that is unique and unpredictable. This CRP pair is fundamental to the PUF's functionality[67]. The unpredictability and uniqueness of the CRPs make it extremely difficult for an attacker to replicate or predict the PUF's behavior, thereby enhancing the security of the IoV network. Additionally, PUFs can be used to generate cryptographic keys on-the-fly, eliminating the need to store keys in non-volatile memory, which is susceptible to physical attacks. The integration of PUF devices within IoV introduces a unique paradigm for secure authentication[81]. The utilization of PUFs in vehicular networks enhances security by enabling unique, device-specific keys that are inherently resistant to cloning and physical attacks[14].

This chapter provides the mechanisms using PUFs that facilitate anonymous authentication, ensuring that communication within the network remains secure and private. PUF generates unique responses to authentication challenges, which are used to verify the vehicle's identity without the need to store sensitive information. The authentication involves the exchange of CRP between the vehicle and the RSCN, with the edge computing layer[72] managing the computational load. Further, by integrating edge computing with PUF-based authentication, the system reduces latency and enhances the efficiency of real-time data processing, which is crucial for rapid decision-making in vehicular environments. The edge nodes, equipped with advanced computational and storage capabilities, handle the majority of data processing tasks, thereby reducing the load on central servers and minimizing network delays[149, 63].

The research is further expanded to explore the role of drones in the IoV network to provide PUF-based lightweight authentication. Drones[208], equipped with advanced sensing and communication capabilities, function as dynamic nodes in vehicular networks. Their ability to collect, process, and transmit data significantly enhances the overall functionality and security of drone-based IoV systems. In situations where the buffer of RSCNs becomes overloaded due to high data traffic, drones can act as auxiliary nodes to offload the excess data. These drones can dynamically position themselves in areas with high traffic density, offering additional storage and processing capabilities. By integrating drones with PUF-based authentication mechanisms, the system ensures secure communication even in the most dynamic and distributed network scenarios. Consequently, drones contribute to a more resilient and adaptive IoV infrastructure, capable of maintaining high levels of security and efficiency. Their mobility allows for real-time adaptability to changing network conditions, ensuring continuous and reliable data flow across the vehicular network.

5.1 PUF BASED LIGHTWEIGHT AND EFFICIENT AUTHENTICATION BY INTEGRATING THE EDGE-COMPUTING

In IoV, PUFs have gained attention as a promising solution for lightweight security. PUFs leverage inherent manufacturing variations in chips to generate unique secret keys for device authentication. They are cost-effective, consume less power, and can be implemented in FPGA and ASIC [209]. The primary types of PUFs include Strong PUFs [80, 210], Controlled PUFs [211], and Weak PUFs [212], each offering different levels of challenge-response behavior and security features. Despite these advancements, IoV systems face challenges such as limited network coverage, security vulnerabilities, high communication overhead, centralized cloud dependencies, limited data storage, and moderate computing capabilities. To address these challenges, this chapter presents a PUF and edge-computing based Secure Protocol for IoV, named as PESPI, utilizing one-way hash functions and lightweight PUF functions [67]. PESPI aims to optimize communication overhead, storage constraints, computation time, and security issues. PUF devices eliminate the need to store secret parameters, generating keys and bio-metric passwords dynamically to prevent physical and side-channel attacks [213]. By incorporating edge computing with traditional IoV, the scheme minimizes latency and enhances bandwidth efficiency by processing real-time data locally. Edge computing mitigates security risks, reduces central server computation and storage costs, and ensures high throughput and data security. The primary contributions of the proposed scheme are as follows:

1. **PUF-based three-layered architecture:** PESPI introduces a hybrid three-layered architecture (refer 5.1) consisting of vehicles, edge computing nodes, and cloud servers. This architecture balances the needs of both centralized and distributed environments, thereby mitigating the bottleneck problem that arises from the failure of a central authority.

2. **On-demand generation of secret parameters using PUF:** PESPI leverages PUFs to randomly generate secret parameters on demand, eliminating the need to store sensitive data on OBUs. This approach significantly reduces the storage costs for vehicles and edge nodes, ensuring that the system remains lightweight and efficient.

3. **Enhanced security against various attacks:** PESPI uses PUF with lightweight one-way hash functions and XOR operations to address several critical security issues, including physical attack, data leakage from side-channel attack, Dos attack, man-in-the-middle attack, etc.

4. **Simulation and performance analysis:** The PESPI scheme has been simulated using SUMO and NS-2 version 2.35 simulators. The performance analysis demonstrates a reduction in computation time and communication cost by up to 84.96% and 57%, respectively.

5. **Comprehensive security analysis:** PESPI's security has been rigorously analyzed through theoretical validation using ROM and practical verification via the ProVerif tool. This dual-layered security analysis confirms the robustness of PESPI against a wide range of attacks.

5.1.1 Network Model

The network model for PESPI is designed as a three-layered architecture incorporating edge computing. This hierarchical structure as depicted in Figure 5.1, comprises the vehicle layer, the edge computing layer, and the cloud layer. The vehicle layer consists of vehicles functioning as smart nodes equipped with capabilities for sensing, communication, computation, and data storage. These vehicles are integrated with PUF chips along with OBUs. The PUF chips are crucial for generating unique, unclonable responses to security challenges, thereby enhancing the level of authentication within the network. The PUF chips ensure that even if an attacker gains access to the OBU, they cannot replicate the chip's unique response patterns, and PESPI assumes secure communication between OBUs and PUF chips to ensure that the secret parameters remain uncompromised. The edge computing layer comprises RSCNs as intelligent edge nodes positioned

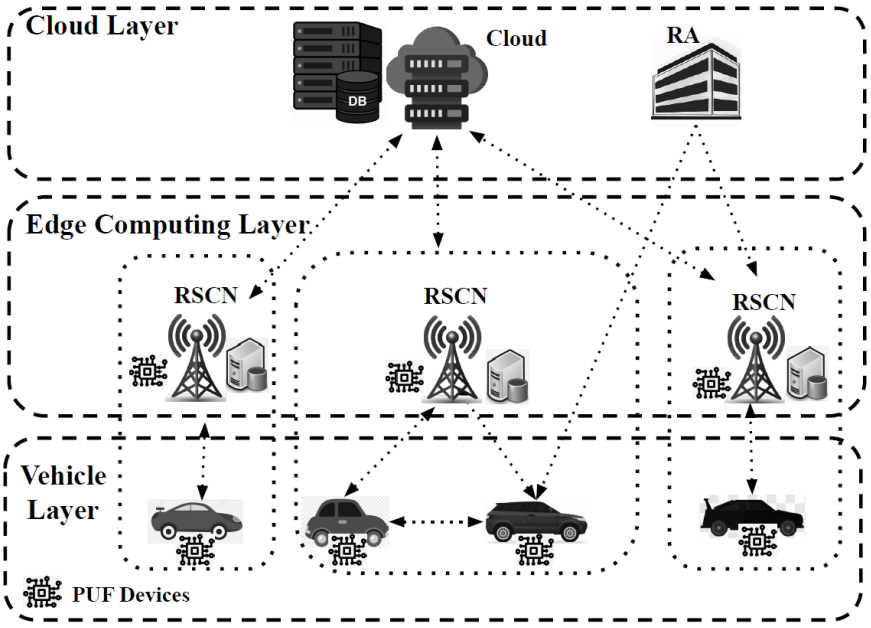


Figure 5.1: PUF and edge-computing based three layered network model for proposed PESPI scheme

strategically to provide enhanced storage, computation, and communication capabilities. These edge nodes are also equipped with the PUF chips for the authentication purpose. These nodes play a critical role in offloading computational tasks from vehicles and caching data to improve network efficiency. Intelligent edge nodes handle intensive computational tasks that are offloaded by the vehicles, such as complex data processing and analytics. These nodes collect data from multiple vehicles, process it to generate meaningful insights, and forward the processed information to the cloud server when necessary. By providing a

distributed environment, the edge computing layer reduces the dependency on the centralized cloud server, thus enhancing system reliability and resilience against central point failures.

The cloud layer is having the centralized cloud server that possesses superior computational and storage capabilities compared to both vehicle and edge nodes. This layer provides comprehensive data management and supports high-level decision-making processes. The cloud server stores vast amounts of data collected from the edge and vehicle layers, ensuring long-term data availability and historical analysis capabilities. Leveraging powerful computational resources, the cloud server performs advanced data analytics, machine learning, and big data processing tasks that are beyond the capabilities of edge nodes. It coordinates the operations of the vehicle and edge layers, manages the distribution of tasks, and ensures that data integrity and security protocols are maintained across the entire network. By integrating these three layers, PESPI achieves a robust, scalable, and efficient vehicular network that addresses the challenges of modern IoV systems, such as latency, security, and data management.

Table 5.1: Required notations and their definitions

Notation	Meaning
V_i	i^{th} smart vehicle
VID	Identity of the smart vehicle
Req_{reg}	Smart vehicle registration request
$C_V - R_V$	CRP for smart vehicle
$C_V^{sync} - R_V^{sync}$	Re-synchronization CRP for smart vehicle
$\Phi()$	PUF function
$ProVID, PVID$	Provisional and pseudo identity for smart vehicle
β_V	Bio-metric thumb impression using PUF function
$PASS$	Generated strong password
E_i	i^{th} edge node
UID_E	Unique identity of the edge node
$C_E - R_E$	CRP for the edge node
$C_E^{sync} - R_E^{sync}$	Re-synchronization CRP for the edge node

5.1.2 Proposed Scheme: *PESPI*

Before the detailed explanation for the methodology of PESPI, the proposed scheme considers the widely accepted CK-adversary model [181] described in the subsection 3.1.1, along with the Dolev-Yao adversary model (DY-model)[214], where an adversary can intercept the ongoing communication and may modify, delete, or inject wrong information. The model is mainly used to prove the properties of interactive cryptographic protocols. In PESPI, two-factor authentication is employed, combining the user's password with fingerprints generated by the PUF device. Thus, the user device cannot be accessible by someone else in case of stolen or being physically compromised. By providing this two-factor authentication, PESPI provides more secure environment while reducing the computation and storage cost. Hence, it can be used in the critical situations along with the general purpose applications in IoV environment. Along with this, it performs CRP updates periodically. This feature provides extra layer of security against the cloning attack. Table 5.1 refers the notations and their definition that are used in PESPI. The following phases are involved in the authentication procedure of PESPI:

Smart vehicle registration phase: The *Smart Vehicle Registration* algorithm, depicted in Algorithm 11, outlines the process of registering a smart vehicle, V_i, using a PUF for enhanced security. Initially, V_i sends its vehicle identification, VID, and registration request, Req_{reg}, to the RA through a secure medium. Upon receiving the request, the RA generates a random challenge, C_V, and a set of re-synchronization challenges, C_V^{sync}, to mitigate DoS attacks and desynchronization issues. These challenges are sent to V_i. The PUF device in V_i processes these challenges to produce corresponding responses, R_V and R_V^{sync}, which are then sent back to the RA. The RA, in turn, generates a provisional identity, $ProVID$, and a set of pseudo identities, $PVID$, sending these back to V_i. The RA also stores all received and generated data in database for future reference. Subsequently, V_i saves the provisional and pseudo identities in its database. The driver of V_i inputs their bio-metric thumb impression, $\mathbb{B}_{Vi}$, into the PUF device, which then generates a new bio-metric impression, β_V, using PUF. This bio-metric impression is combined with a password, pwd_V, selected by the driver to form a strong password, $PASS$, through a hash function, where $PASS = h(\beta_V || pwd_V)$. This process ensures a secure and robust registration mechanism for the smart vehicle. The whole process of V_i registration is diagrammatically shown in Figure 5.2a.

Algorithm 11 Smart vehicle registration

Input: Vehicle identity VID, registration request Req_{reg}
Output: Provisional and pseudo-identity of vehicle $\{ProVID, PVID\}$, challenge-response pairs $\{\{C_V, C_V^{sync}\} - \{R_V, R_V^{sync}\}\}$, password $\{PASS\}$

Procedure SENDREQUEST(VID, Req_{reg}) // V_i `sends registration request to RA`
V_i sends $\{VID, Req_{reg}\}$ to RA through a secure medium
end Procedure
Procedure GENERATECHALLENGES // `RA generates challenges to avoid DoS and desynchronization`
RA generates random challenge C_V
RA generates resynchronization challenges $C_V^{sync} = \{c_1, c_2, \ldots, c_n\}$
RA sends $\{C_V, C_V^{sync}\}$ to V_i
end Procedure
Procedure GENERATERESPONSES // `PUF device generates responses to challenges`
$R_V \leftarrow \Phi_V\{C_V\}$
$R_V^{sync} \leftarrow \Phi_V\{C_V^{sync}\}$
V_i sends $\{R_V, R_V^{sync}\}$ to RA
end Procedure
Procedure GENERATEIDENTITIES // `RA generates provisional and pseudo identities`
RA generates $ProVID$ and $PVID = \{pid_1, pid_2, \ldots, pid_n\}$
RA sends $\{ProVID, PVID\}$ to V_i
RA stores all relevant data in its database
end Procedure
Procedure STOREIDENTITIES // V_i `saves identities and generates strong password`
V_i saves $\{ProVID, PVID\}$ in its database
`/* Driver inserts biometric thumb impressions */`
$\beta_V \leftarrow \Phi_V(\mathbb{B}_V)$
`/* Combine biometric impression with driver-selected password */`
$PASS \leftarrow h(\beta_V || pwd_V)$
end Procedure

Edge node registration phase: The *Edge Node Registration* algorithm, illustrated in Algorithm 12, outlines the steps for registering an intelligent edge node, E, using a PUF to enhance security. Initially, RA generates a challenge, C_E, and a set of re-synchronization challenges, C_E^{sync}, to address potential DoS attacks and synchronization issues. These challenges are securely transmitted to E. The PUF device within E processes these challenges to generate corresponding responses, R_E and R_E^{sync}, which are then sent back to the RA. Subsequently, the RA generates a unique identity, UID_E, for the edge node and stores all relevant

information, including $UID_E, C_E, C_E^{sync}, R_E, R_E^{sync}$, in its database for future reference. This process ensures a secure and robust registration mechanism for the intelligent edge node. The entire procedure of edge node registration is explained in Figure 5.2b depicts the CRP generation process for edge registration.

Algorithm 12 Edge node registration

Input: Edge node E
Output: Unique identity of edge node $\{UID_E$, challenge-response pairs $\{C_E, C_E^{sync}\} - \{R_E, R_E^{sync}\}\}$
1: **Procedure** GENERATECHALLENGES // `RA generates initial and resynchronization challenges`
2: RA generates C_E and $C_E^{sync} = \{c_1, c_2, \ldots, c_n\}$ to mitigate DoS attacks and synchronization issues.
3: RA sends $\{C_E, C_E^{sync}\}$ to E through a secure medium.
4: **end Procedure**
5: **Procedure** GENERATERESPONSES // `PUF device generates responses to challenges`
6: $R_E = \Phi_E\{C_E\}$
7: $R_E^{sync} = \Phi_E\{C_E^{sync}\}$
8: E sends $\{R_E, R_E^{sync}\}$ to RA.
9: **end Procedure**
10: **Procedure** GENERATEANDSTOREIDENTITIES // `RA generates and stores unique identity`
11: RA generates unique identity UID_E for E
12: RA stores $\{UID_E, \{C_E, C_E^{sync}\}, \{R_E, R_E^{sync}\}\}$ in its database.
13: **end Procedure**

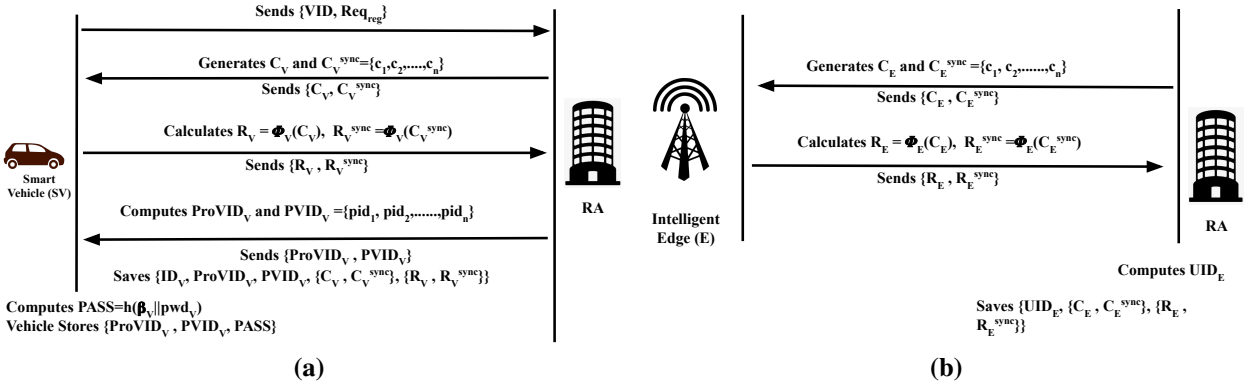


Figure 5.2: Registration process for smart vehicles and edge node using CRP mechanism

Anonymous mutual authentication and CRP update phase: The Algorithm 13 and Algorithm 14 describe an *Authentication Process* between smart vehicle, RA, and an edge node. This process ensures secure communication through the use of PUFs, nonce, and cryptographic hash functions. The first part of the algorithm (refer Algorithm 13) begins with the driver of V_i providing a bio-metric impression, $\mathbb{B}_V$, which is processed by the PUF device to compute a bio-metric-based secret, β_V. This secret is then used to generate a password, $PASS^*$. If this password matches the stored password, $PASS$, the PUF device generates a nonce, η_V and sends a login message, M_1 to the RA. The RA responds by generating its own nonce, η_{RA}, and computes a hash value, δ_1, to send an authentication request, M_2, back to the vehicle. Upon receiving M_2, the vehicle's PUF device computes its response hash, δ_1^*, and verifies it against δ_1. If they match, the PUF device updates certain values, computes a new response hash, δ_2, and sends an authentication response, M_3, to the RA. The RA verifies the response and, if valid, calculates session-related values and sends a message, M_4, containing session keys and updated values back to the vehicle.

Algorithm 13 Anonymous mutual authentication process (Part-1)

Input: Identity of vehicle VID, vehicle's registration request Req_{reg}
Output: unique identity of edge $\{UID_E$, challenge-response pairs for edge $\{C_E, C_E^{sync}\} - \{R_E, R_E^{sync}\}\}$

Driver of V_i puts his/her bio-metric impression $\mathbb{B}_V$ and PUF device computes $\beta_V = \Phi_V(\mathbb{B}_V)$ and $PASS^* = h(\beta_V || pwd_V)$.
if $PASS^* == PASS$ **then**
    PUF device generates a nonce η_V. It sends login message $M_1 = \{\eta_V, ProVID\}$ to RA through public channel.
    RA generates nonce η_{RA} and calculates: $\eta_{RA}^* = \eta_{RA} \oplus R_V$, $\delta_1 = h(\eta_{RA}^* || R_V || \eta_V)$.
    RA sends authentication request $M_2 = \{C_V, \eta_{RA}^*, \delta_1\}$ to V_i.
    PUF device of V_i calculates: $R_V = \Phi_V(C_V)$, $\delta_1^* = h(\eta_{RA}^* || R_V || \eta_V)$.
    if $\delta_1^* == \delta_1$ **then**
        PUF device calculates: $\eta_{RA} = \eta_{RA}^* \oplus R_V$, $C_V^{updated} = h(C_V || R_V)$, $R_V^{updated} = \Phi_V(C_V^{updated})$.
        PUF device gets the identity of the corresponding edge node UID_E and calculates: $R_V^{'} = h(VID || R_V) \oplus R_V^{updated}$, $UID_E^{'} = h(VID || \eta_{RA}) \oplus UID_E$, $\delta_2 = h(R_V^{'} || R_V || \eta_{RA} || UID_E^{'})$.
        PUF device sends authentication response $M_3 = \{R_V^{'}, UID_E^{'}, \delta_2\}$ to RA through public channel.
    else
        PUF device discontinues the session.
    end if
    RA calculates $\delta_2^* = h(R_V^{'} || R_V || \eta_{RA} || UID_E^{'})$.
    if $\delta_2^* == \delta_2$ **then**
        RA calculates: $UID_E = h(VID || \eta_{RA}) \oplus UID_E^{'}$. RA selects x_1 and $SKey$ and calculates: $x_1^{'} = x_1 \oplus R_E$, $SKey_E = h(R_E || x_1) \oplus SKey$, $\delta_3 = h(x_1^{'} || R_E || SKey_E || ProVID)$. RA sends $M_4 = \{ProVID, x_1^{'}, SKey_E, C_E, \delta_3\}$.
    else
        RA discontinues the session.
    end if
else
    PUF device discontinues the session.
end if

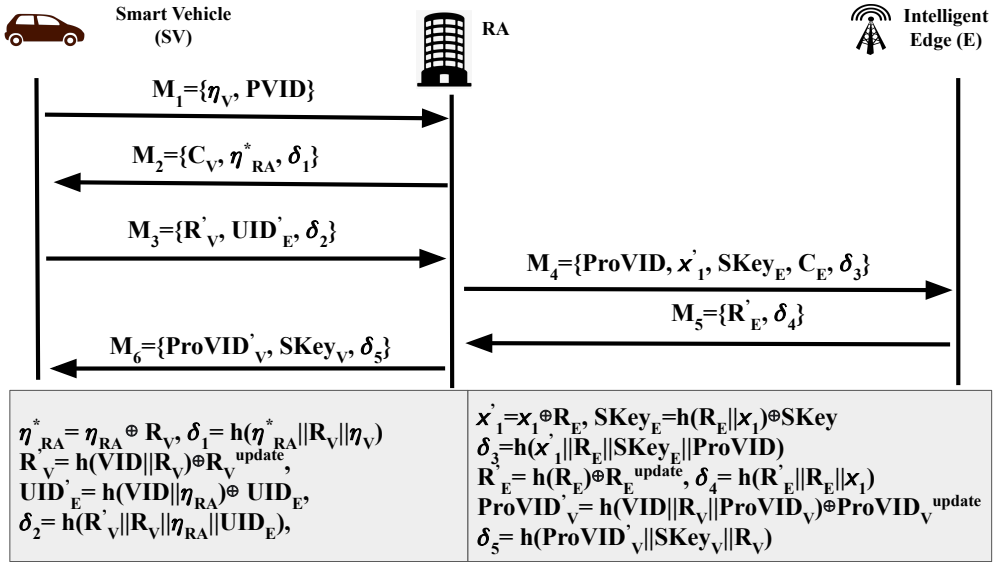


Figure 5.3: Anonymous mutual authentication

The second part of the algorithm (refer Algorithm 14) begins with edge calculating its response, R_E, and verifying the received hash, δ_3^*. If the verification is successful, the edge node calculates the original random number, (x_1, derives the session key, $SKey = h(R_E||x_1) \oplus SKey_E$, and updates its challenges, $C_E^{updated} = h(C_E||R_E)$, along with a new response, $R_E^{updated} = \Phi_E(C_E^{updated})$. Next, the edge node computes $R_E^{'}$ and a new hash, δ_4. It then sends the message, M_5 to the RA through an insecure medium. Now, RA calculates its own hash, δ_4^*, to verify the edge node's response. If they match, the RA computes updated values: C_E^{update}, R_E^{update}, C_V^{update}, and R_V^{update}. The RA then randomly generates a new provisional identity, $ProVID^{update}$ for the vehicle and calculates $SKey_V$, $ProVID^{'}$, and δ_5. The RA saves these updated values in its database for future reference and sends message, M_6, to the vehicle through an insecure medium. Finally, the vehicle's PUF device computes the hash, δ_5^*, to verify the final message. If the verification is successful, the PUF device computes the session key, $SKey$, and the updated provisional identity, $ProVID^{update}$. It saves the updated provisional identity in its database for future reference. If any verification step fails, the corresponding device (either the PUF device, RA, or edge node) discontinues the session to ensure security. For better understanding, the entire process of authentication is illustrated in the Figure 5.3.

Algorithm 14 Anonymous mutual authentication process (part-2)

Continued from Algorithm 13:
if $PASS^* == PASS$ **then**
  Edge node E calculates $R_E = \Phi_E(C_E)$ and $\delta_3^* = h(x_1^{'}||R_E||SKey_E||ProVID)$.
  if $\delta_3^* == \delta_3$ **then**
    Edge E calculates $x_1 = x_1^{'} \oplus R_E$, $SKey = h(R_E||x_1) \oplus SKey_E$, $C_E^{updated} = h(C_E||R_E)$, $R_E^{updated} = \Phi_E(C_E^{updated})$, $R_E^{'} = h(R_E) \oplus R_E^{updated}$, and $\delta_4 = h(R_E^{'}||R_E||x_1)$.
    E sends the message $M_5 = \{R_E^{'}, \delta_4\}$ to the RA through a insecure medium.
  else
    Edge E discontinues the session.
  end if
  RA calculates $\delta_4^* = h(R_E^{'}||R_E||x_1)$.
  if $\delta_4^* == \delta_4$ **then**
    RA computes: $C_E^{update} = h(C_E||R_E)$, $R_E^{update} = h(R_E) \oplus R_E^{'}$, $C_V^{update} = h(C_V||R_V)$, $R_V^{update} = h(VID||R_V) \oplus R_V^{'}$.
    RA generates $ProVID^{update}$ and calculates: $SKey_V = h(VID||R_V||\eta_{RA}) \oplus SKey$, $ProVID^{'} = h(VID||R_V||ProVID) \oplus ProVID^{update}$, $\delta_5 = h(ProVID^{'}||SKey_V||R_V)$.
    RA saves $ProVID^{update}$,C_V^{update},R_V^{update},C_E^{update},R_E^{update} in its atabase for further reference and sends a message $M_6 = \{ProVID^{'}, SKey_V, \delta_5\}$ to the vehicle V_i through an insecure medium.
  else
    RA discontinues the session.
  end if
  The PUF device of V computes $\delta_5^* = h(ProVID^{'}||SKey_V||R_V)$.
  if $\delta_5^* == \delta_5$ **then**
    PUF device compute the session key $SKey = h(VID||R_V||\eta_{RA}) \oplus SKey_V$ and $ProVID^{update} = h(VID||R_V||ProVID) \oplus ProVID^{'}$ and saves $ProVID^{update}$ in its database for further reference.
  else
    PUF device discontinues the session.
  end if
else
  PUF device discontinues the session.
end if

5.1.3 Security Analysis

In this section, a comprehensive security analysis of PESPI has been provided in different perspectives. Initially, the formal analysis using ROM model is done. A correctness proof based on ProVerif tool is also discussed. Along with these two analysis, this section explores the security comparison between PESPI and other state-of-the-art schemes. Table 5.2 shows the comparison with different types of security attacks and privacy features. The attacks are denoted as: S1: Replay attack; S2: Modification; S3: Impersonation attack; S4: Repudiation attack; S5: Man-in-the-middle attack; S6: Location privacy preservation (Anonymity); S7: Cloning attack; S8: Physical attack; S9: Side channel attack; S10: Distributed DoS attack; S11: Revocation; S12: Location tracing; ✓: Secure; ×: Vulnerable.

Table 5.2: Security comparison between different schemes

Schemes	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12
Gope et al.[106]	✓	✓	×	✓	×	×	✓	✓	×	×	×	×
Gope et al.[107]	✓	✓	✓	×	×	✓	✓	✓	×	×	✓	✓
Chatterjee et al.[108]	✓	✓	×	×	×	✓	✓	✓	×	×	✓	✓
Aman et al.[51]	✓	✓	✓	×	✓	×	✓	✓	×	×	×	×
Jiang et al.[142]	✓	✓	✓	×	✓	×	✓	✓	×	×	×	×
PESPI	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Formal analysis using ROM: PESPI mainly uses one-way hash function and PUF function. The definition is provided below:

- **One-way hash function, $h()$:** It is written as: $h : \{0,1\}^* \rightarrow \{0,1\}^n$. It means $h()$ uses variable length string as input and gives fixed n-length string as output.

  If $\mathbb{A}$ attacks on the collision resistant $h()$, it can be written in mathematical form as: $\Gamma_{\mathbb{A}}^{h()}(T_r) \leq \mu$. where, T_r is the run-time, μ is the threshold value, and $\Gamma_{\mathbb{A}}^{h()}(T_r)$ is the advantage of $\mathbb{A}$ that is represented as: $\Gamma_{\mathbb{A}}^{h()}(T_r) = P[(h(x_1) = h(x_2))], x_1, x_2 \in R, x_1 \neq x_2$.

- **Physically Unclonable Function, ($\Phi()$):** A PUF, $\Phi()$ is considered to be secure if it holds the following requirements:

  Let's say, $\Phi_1()$ and $\Phi_2()$ are two PUF functions and for any random challenge $c_1, c_2 \in \{0,1\}^k$, the following equation should be true: $P[D_h(\Phi_1(c_1), \Phi_2(c_2)) > Th] = 1 - \varepsilon$, where, D_h denotes the hamming distance and Th is the threshold value for error tolerance. The following theorems are considered to prove the security and efficiency of PESPI.

Theorem 1: The advantage of $\mathbb{A}$, to break the security of PESPI for computing the session key *SKey* in polynomial time T_p is formulated as:

$$\Gamma_{\mathbb{A}}^{PESPI}(T_p) \leq \left(\frac{Query_h^2}{|h()|} + \frac{Query_p^2}{|\Phi()|} + 2max(Z^{'}.Query_S^{S^{'}}, \frac{Query_S}{2^b})\right), \tag{5.1}$$

Where $Query_h, Query_p$, and $Query_S$ are the number of queries for hash function, PUF function, and sending/receiving a message respectively. Here, b denotes the number of bits in bio-metric impression $\mathbb{B}$. $Z^{'}$ and

S' are the Zipf's law parameters used in the distribution done using Zipf's law [204]. It is different from the uniform distribution. $|h()|, |\Phi()|$ represents the range space of hash functions and PUF functions.

Proof: To prove the security of session key, following cases are considered:

- **Case 1:** The $\mathbb{A}$ performs an attack on the PESPI protocol for guessing w bits of the password in polynomial time, T_p. Here, w bits are selected in starting of the first attempt, hence the advantage for $\mathbb{A}$ can be formulated as:
$$\Gamma_{\mathbb{A}}^{PESPI}(T_p) = |2.P[Win_1] - 1|, \tag{5.2}$$
Here, Win_1 denotes the success of $\mathbb{A}$ in guessing the w bits in case 1.

- **Case 2:** The $\mathbb{A}$ performs eavesdrop attack by accessing the transmitted messages. Consider the messages as, M_1, M_2, M_3, M_4, M_5, and M_6. It access the session key which is computed as $SKey = h(VID||R_V||\eta_{RA}) \oplus SKey_V$ and $SKey = h(R_E||x_1) \oplus SKey_E$ in case of smart vehicles and edge device respectively. Here, $\mathbb{A}$ needs to find out the secret parameters like response of V_i, R_V, response of E, R_E and identity of V, VID. These responses and identities are only known to the authenticated users and thus the probability of winning is same as the case 1 and it can be shown as:
$$P[Win_2] = P[Win_1]. \tag{5.3}$$

- **Case 3:** The $\mathbb{A}$ performs impersonation attack by pretending itself as a authorised receiver. Since all the transmitted messages have the secret parameters and the hash function is collision resistant, the difference between the probability of Win_3 and Win_2 can be formulated as:
$$|P[Win_3] - P[Win_2]| \leq \frac{Query_h^2}{|h()|}. \tag{5.4}$$

- **Case 4:** Similar to case 3, the above equation for the secure PUF function can be formulated as:
$$|P[Win_4] - P[Win_3]| \leq \frac{Query_p^2}{|\Phi()|}. \tag{5.5}$$

- **Case 5:** It is assumed that the OBU and Edge are corrupt devices, and $\mathbb{A}$ can access the secret parameters stored in these devices. Due to the PUF device, the probability of guessing the password is reduced to $\frac{1}{2^b}$, where b is the number of bits in the biometric password $\mathbb{B}_V$. Here, $\mathbb{A}$ can also try to guess the password with less entropy using Zipf's law, and those parameters can be used as well. The relation can be formulated as:
$$|P[Win_5] - P[Win_4]| \leq \max(Z'.Query_S^{S'}, \frac{Query_S}{2^b}). \tag{5.6}$$
Finally, it is assumed that $\mathbb{A}$ can also perform random guessing of w bits of the user password, and the probability of winning can be written as:
$$P[Win_5] = \frac{1}{2}. \tag{5.7}$$

Using Eq. (5.2), (5.3), and (5.4), the following is obtained:

$$\frac{1}{2}\Gamma_{\mathbb{A}}^{PESPI}(T_p) = |P[Win_2] - P[Win_5]|. \tag{5.8}$$

Using Eq. (5.4), (5.5), and (5.6), the result is obtained as:

$$|P[Win_2 - Win_5]| \leq \frac{Query_h^2}{|h()|} + \frac{Query_p^2}{|\Phi()|} + 2max(Z^{'}.Query_S^{S^{'}}, \frac{Query_S}{2^b}). \tag{5.9}$$

Finally, using Eq. (5.8) and (5.9), the desired output of Eq. (5.1) is obtained:

$$\Gamma_{\mathbb{A}}^{PESPI}(T_p) \leq \frac{Query_h^2}{|h()|} + \frac{Query_p^2}{|\Phi()|} + 2max(Z^{'}.Query_S^{S^{'}}, \frac{Query_S}{2^b}). \tag{5.10}$$

Theorem 2: The PUF responses R_V and R_E are not predictable.

Proof: The processing of a PUF chip can be represented as $\{0,1\}^n \leftarrow \{0,1\}^m$ where, n is the length of input challenge and m is the length of output response. The theorem can be proven using the following points:

- $\mathbb{A}$ sends the challenge $C_{\mathbb{A}}$ to the challenger and then challenger provide response $R_{\mathbb{A}}$ to $\mathbb{A}$ using PUF chip.
- Now, the challenger select a random challenge, C_r and using the PUF chip it produces the corresponding response, R_r as: $R_r = \Phi(C_r)$.
- $\mathbb{A}$ sends random challenges other than C_r, polynomial number of time and tries to guess the response for challenge, C_r. $\mathbb{A}$ guesses the response as $R_r^{'}$. Now, the adversary will be succeed if $R_r^{'} = R_r$.

The advantage Γ of $\mathbb{A}$ is formulated as:

$$\Gamma_{\mathbb{A}}^{\Phi()} = P[R_r^{'} = R_r]. \tag{5.11}$$

It is considered that every $\Phi()$ produces a unique response corresponding to the given challenges and is unclonable. Hence, $\mathbb{A}$ only try random guesses and the probability of winning is formulated as:

$$P[R_r^{'} = R_r] = \frac{1}{2^m}. \tag{5.12}$$

Using Eq. (5.10) and (5.11), the results is obtained as: $\Gamma_{\mathbb{A}}^{\Phi()} = \frac{1}{2^m}$.

Theorem 3: Anonymous authentication: The privacy of the driver and V_i is preserved during the entire authentication process, meaning that V_i is untraceable to $\mathbb{A}$.

Proof:

- The challenger selects randomly one of the smart vehicles, lets say, V_r. $\mathbb{A}$ performs an attack on V_r and RA in polynomial number of times.

- The probability of $\mathbb{A}$ to succeed is: $P[V_{\mathbb{A}} = V_r] = \frac{1}{2}$, where, $V_{\mathbb{A}}$ is the vehicle that is randomly accessed by $\mathbb{A}$. The advantage of $\mathbb{A}$ can be formulated as: $\Gamma_{\mathbb{A}}^{AnoAuth} = 2 * (P[V_{\mathbb{A}} = V_r] - \frac{1}{2})$. Now, it is concluded that if $\mathbb{A}$ makes random guesses, there is no advantage for $\mathbb{A}$, i.e., $\Gamma_{\mathbb{A}}^{AnoAuth} = 0$.

Theorem 4: Mutual authentication: If V_i, E, and RA successfully performs the authentication process. V_i and E are considered to be authenticated (RA is already assumed to be authenticated).

Proof:

- The challenger randomly selects V_r among all the vehicles and $\mathbb{A}$ performs impersonation attack on V_r.
- For the successful attack, $\mathbb{A}$ needs to guess the bio-metric password produced by PUF device, i.e. $\beta_V = \Phi(\mathbb{B})$. It is assumed that $\mathbb{A}$ randomly guesses $m_{\mathbb{A}}$ bits of the password; hence, the probability of winning is formulated as:
$$P[\beta_V^{\mathbb{A}} = \beta_V] = \frac{1}{2^{m-m_{\mathbb{A}}}}. \tag{5.13}$$
- Next, the advantage of $\mathbb{A}$ to be the legitimate one can be given as: $\Gamma_{\mathbb{A}}^{MuAuth} = P[\beta_V^{\mathbb{A}} = \beta_V] - \Gamma_{\mathbb{A}}^{\Phi()}$.
- By using the proof of *Theorem 1*, it can be stated that $\mathbb{A}$ can only guess the password, i.e., $m_{\mathbb{A}} = 0$. Using this relation, it can further be stated that: $P[\beta_V^{\mathbb{A}} = \beta_V] = \frac{1}{2^m} = \Gamma_{\mathbb{A}}^{\Phi()}$. Thus, it can be concluded that:
$$\Gamma_{\mathbb{A}}^{MuAuth} = 0. \tag{5.14}$$

5.1.4 Performance Analysis and Simulation Results

This section explains the simulation environment, performance factors, analysis, and efficiency of the PESPI scheme. The performance of the PESPI scheme is compared with existing state-of-the-art schemes based on IoT and IoV networks. Some recent works considered for comparison include Gope et al. [106, 107], Chatterjee et al. [108], Aman et al. [51], and Jiang et al. [142]. The code has been executed on a Desktop as well as a Raspberry Pi 4 (as an edge node). The followings are the configuration details:

- **Desktop System:** Desktop systems are considered as intelligent edge nodes. PESPI is implemented on a Windows 11 machine with the hardware configuration of an Intel 64-bit Core i5 at a frequency of 4.1 GHz, 8GB DDR4 RAM, and the software configuration of Netbeans 8.2 and Java 8. The implementation includes 20 RSCNs, 8 controllers, and 300 vehicles.
- **Raspberry Pi 4:** In PESPI, mobile vehicular nodes are represented by Raspberry Pi 4 chip. The hardware configuration of Raspberry Pi is Broadcom BCM2711, Quad-core Cortex-A72 1.5GHz, and 4GB RAM, with software configuration as a Linux based operating system called Noobs v3.3.1, Netbeans IDE 11, and java 8. The simulation environment for PESPI is similar to the desktop system.
- **PUF Chip System:** Along with the Desktop and Raspberry Pi system, a 128-bit arbiter fuzzy extractor PUF circuit integrated on the MSP430 micro-controller machine with a 798 MHz CPU is also considered.

Table 5.3: Execution time for each cryptographic operation

Operation	For edge node (ms)	For vehicle (ms)
$T_{PUF()}$	0.56	1.04
T_h	0.15	1.82
T_E	40.27	49.13
T_m	4.75	7.68
T_a	7.54	10.86

The following parameters such as computation time, complexity, and communication overhead are considered for the performance analysis of PESPI.

Computation cost: It is the time required to perform a computational process. The execution time for cryptographic operations is denoted as follows: the cryptographic one-way hash function as T_h, the one-way PUF function as $T_{PUF()}$, encryption/decryption as T_E, point multiplication as T_m, and point addition as T_a. The Table 5.3 shows the time taken for the execution of each operation on Desktop and Raspberry Pi system. The PESPI uses 2 $T_{PUF()}$ and 3 T_h for vehicle, 2 $T_{PUF()}$ and 2 T_h for E, and 5 T_h for RA.

Table 5.4: Required parameters for PESPI and other state of the art schemes

Schemes	Vehicle	Edge node	RA
Gope et al.[106]	$8T_h$	NA	$9T_h$
Gope et al.[107]	$3T_{PUF()}+6T_h$	NA	$5T_h$
Chatterjee et al.[108]	$10T_h+T_E$+T_m+T_a	NA	$8T_h+T_E+T_m+2T_a$
Aman et al.[51]	$2T_h+T_E$	NA	$4T_h+3T_E$
Jiang et al.[142]	$T_{PUF()}+13T_h+2T_m+T_E$	NA	$19T_h+T_m$
PESPI	$2T_{PUF()}+3T_h$	$2T_{PUF()}+2T_h$	$5T_h$

Table 5.4 presents the required parameters for anonymous mutual authentication in the state-of-the-art and PESPI. From the table, it can be observed that PESPI requires fewer hash and PUF operations compared to the other state-of-the-art schemes. Therefore, it can be concluded that PESPI takes less time for the execution of operations, indicating its computational efficiency. Furthermore, in terms of network scalability, PESPI provides significant improvements. Figure 5.4a depicts the comparison of scalability

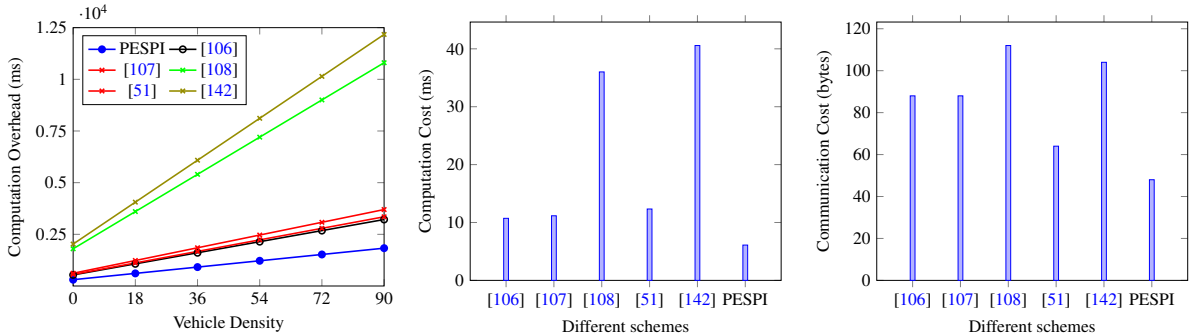


Figure 5.4: Performance evaluation of PESPI: (a) Scalability with respect to vehicle density, comparison of PESPI with state-of-the-art schemes in terms of: (b) computation cost (ms), and (c) communication cost (bytes)

between PESPI and other state-of-the-art methods for different numbers of vehicles. It is observed that the

computation overhead for PESPI is lower in all the considered cases. This is due to the reduced number of one-way hash functions and PUF operations. Therefore, it can be concluded that PESPI is more scalable than other schemes. Table 5.5 provides the values for computation cost (ms) and communication overhead (bytes). Figure 5.4b illustrates the relationship between the total computation cost and different schemes.

Communication cost: It is the total number of packets to be transferred or transmitted for the authentication purpose from one node to another node. For the comparison of PESPI with other existing schemes, 128 bits are considered for CRP, 256 bits for hash functions, 8 bits for identity, and 48 bits for timestamps. The longest message in PESPI is 192 bits long. Table 5.5 shows the communication overhead for PESPI with other state-of-the-art-schemes. It is observed that PESPI uses 48 bytes for communication which is the lowest among all of the other schemes. In Figure 5.4c, the relation between communication overhead with different state-of-the-art is depicted. Performance analysis shows that along with the negligible storage cost, PESPI also reduces the computation time and communication cost upto 84.96% and 57% respectively.

Table 5.5: Comparison of computation and communication overhead

Schemes	Computation cost (ms)	Communication cost (bytes)
Gope et al. [106]	10.72	88
Gope et al.[107]	11.16	88
Chatterjee et al.[108]	36.01	112
Aman et al.[51]	12.33	64
Jiang et al.[142]	40.57	104
PESPI	6.1	48

Complexity: The worst time complexity of a universal one-way hash function is $O(n)$, where n denotes the size of the messages. Therefore, the complexity of PESPI is $O(n)$ for the vehicle, edge node, and RA. Table 5.6 compares the worst time complexity of PESPI with other schemes. The worst time complexity of the scheme presented by Gope et al.[106] is $O(n+m(w))$, where $m(w)$ represents the general multiplication with a w-bit operand. Similarly, the complexity of the scheme by Chatterjee et al.[108] is $O(n+f^{\kappa}+f^{\kappa+1})$, where f^{κ} and $f^{\kappa+1}$ are the ECC-point multiplication and point addition for a finite f-bit field. The complexity of the scheme by Jiang et al.[142] is $O(n+m(l)+\max(\log a+\log b))$, where $\max(\log a+\log b)$ denotes the standard arithmetic binary operation with a and b as two operands. Thus, it can be concluded that PESPI is efficient in terms of complexity.

Table 5.6: Complexity for different authentication schemes

Schemes	Complexity
Gope et al. [106]	$O(n+m(w))$
Gope et al.[107]	$O(n)$
Chatterjee et al.[108]	$O(n+f^{\kappa}+f^{\kappa+1})$
Aman et al.[51]	$O(n)$
Jiang et al.[142]	$O(n+M(l)+max(loga+logb))$
PESPI	$O(n)$

The simulation of PESPI is conducted using the SUMO and NS-2 network simulators. For the case study, a .osm file of Jaipur, an urban city in India, is generated using OSMWebWizard with the help of the SUMO simulator. The generated .osm file is converted to the .cfg file that is the map configure file. Further, .xml file is generated using .cfg file. "TraceExporter" function is used to generate .tcl script for vehicle mobility

with the help of existing .cfg file. Figure5.5a depicts the extracted map segment for MNIT Institute, Jaipur, India.

Table 5.7: Configuration of the network

Parameters	**Values**
Simulation range (Highway and City)	$1000m * 500m$, $2500m * 2500m$
Vehicle's communication range	$200m, 300m, 400m$
Number of vehicles	100
Size of authentication packets	48 bytes
Transmission rate of authentication packets	8 Mbps
Size of application packets	40 bytes
Transmission rate of application packets	2.3 kbps
Number of RSCNs	3, 5, 10
Simulation time	$300s$
Average speed of the vehicle	$10m/s, 15m/s, 20m/s$
Communication protocol	DSRC
Data size	256 bytes
Group key size	160 bit
Buffer size of PESPI	60 packets
Number of simulation	5

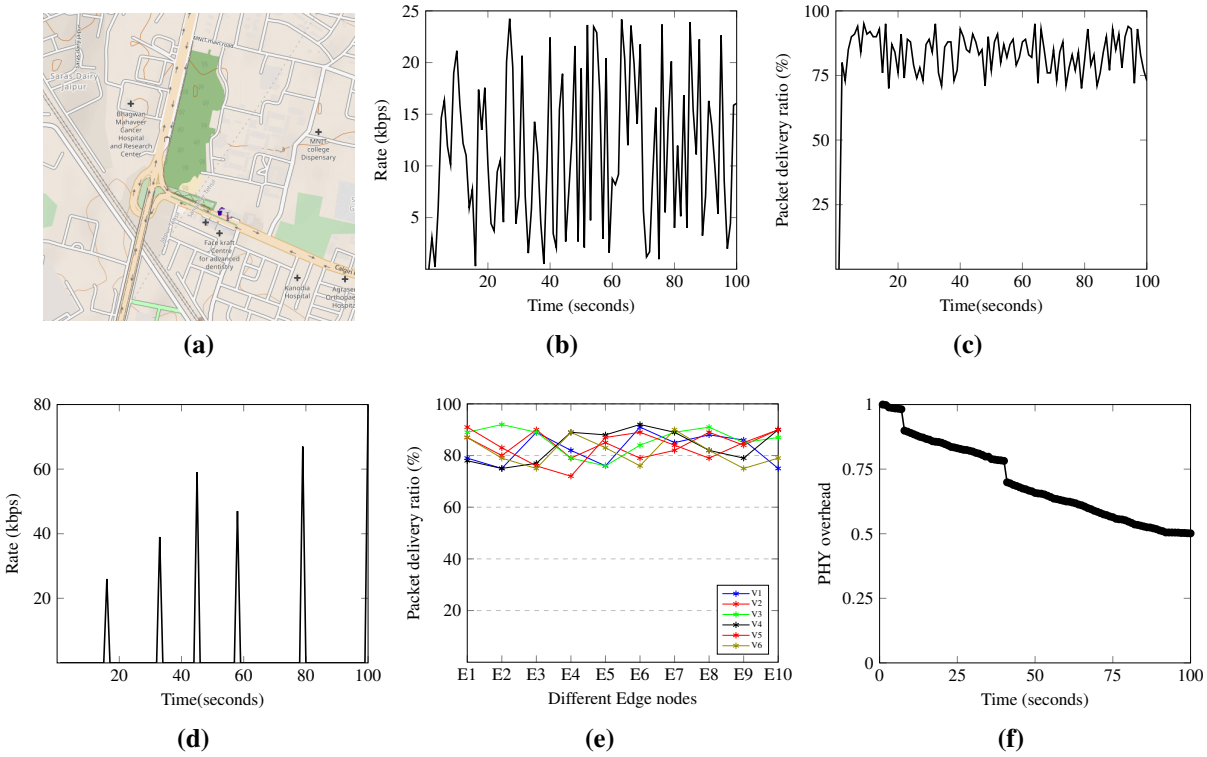


Figure 5.5: Simulation results of PESPI: (a) Selecting MNIT Jaipur region, India using OpenStreetMap, (b) Data receiving rate for application packets, (c) Packet delivery ratio for application packets, (d) Data receiving rate for authentication packets, (e) Packet delivery ratio for authentication packets, and (f) Physical layer overhead during message transmission

PESPI uses IEEE 802.11p which consider DSRC with 5.9 GHz frequency band and 10 MHz control channel. Other network parameters that are used for the simulation of PESPI are included in Table 5.7. In the simulation, 100 vehicle nodes are considered, moving according to the trace file and broadcasting two types of packets: authentication and application packets. The authentication packets are transmitted when a vehicle enters the range of a new edge, while the application packets are considered as normal message transmissions. Simultaneously, all vehicles broadcast application packets while moving within the extracted map segment. Considering two types of communication, i.e., V2V and V2E, both authentication and application packets are generated by the vehicles. Figures 5.5b and 5.5c depict the data receiving rate and PDR of application packets, respectively. The simulation was executed for 100 seconds, and the results indicate that the PDR values for application packets range between 75% and 94%. Figures 5.5d and 5.5e show the data receiving rate and PDR values for the authentication packets. In Figure 5.5d, it is observed that authentication packets are generated only when the vehicle enters the range of a new edge node. The study considered 10 edge nodes and generated the PDR values for different smart vehicles entering the range of these nodes. Figure 5.5e presents the PDR values for six smart vehicles, which lie approximately between 75% and 95% for the noisy channel. Figure 5.5f illustrates the physical layer overhead, which consistently decreases over time. This decrease in the MAC/PHY layer overhead indicates that PESPI improves the QoS and throughput, essential performance measures for communication systems like IoV. Therefore, it can be concluded that PESPI can be integrated into an IoV environment to achieve better performance.

5.2 PUF-BASED LIGHTWEIGHT AUTHENTICATION IN DRONE-ASSISTED INTERNET OF VEHICLES

The report from National Highway Traffic Safety Administration (NHTSA) states that 31,785 people died in traffic crashes in the first nine months of the year 2023 [215]. In IoVs context, researchers have developed a new concept called the Internet of Drones (IoD) [216] and it is getting more acceptance with its various potential applications, such as smart cities surveillance, navigation, military settings, civilian purposes, coordination between UAV, cloud/fog frameworks, WSN, mobile computing, business paradigms, and aerial photography. Basically, IoD consists of a layered network control architecture for coordinating the communication between UAVs in controlled airspace. It provides several generic services, such as traffic surveillance, package delivery, search, rescue, and more. The literature survey indicates that incorporating IoVs and drones can enhance V2V connectivity, infrastructure coverage, network information collection capability, and network inter-working efficiency [208]. Consequently, this thesis explores DIoV, recognizing the contemporary trend of integrating drones into IoV and its significant advantages. Besides the advantages, ensuring security is one of the primary research concerns in DIoVs. Whenever a drone is attacked, the attacker gets the important credentials, which can be used for future privacy breaches [217]. Thus, it compromises the security, privacy, and safety of DIoVs. In these scenarios, authentication plays an important role in protecting not only users' data but also in preventing attacks occurring in drones. Apart from authentication, privacy preservation is also important in these cases. For improving security, one of the recent trends that attracted both industry and academia is the use of PUFs [67].

Existing authentication schemes for DIoV [147, 148, 149, 150] ensure security against various drone attacks; however, network delay and performance issues remain a research concern. Consequently, this chapter discusses an extended version of PESPI, named *SECURE*, which uses edge computing to overcome network delays and PUF to ensure a high level of security. In *SECURE*, a three-layer network architecture

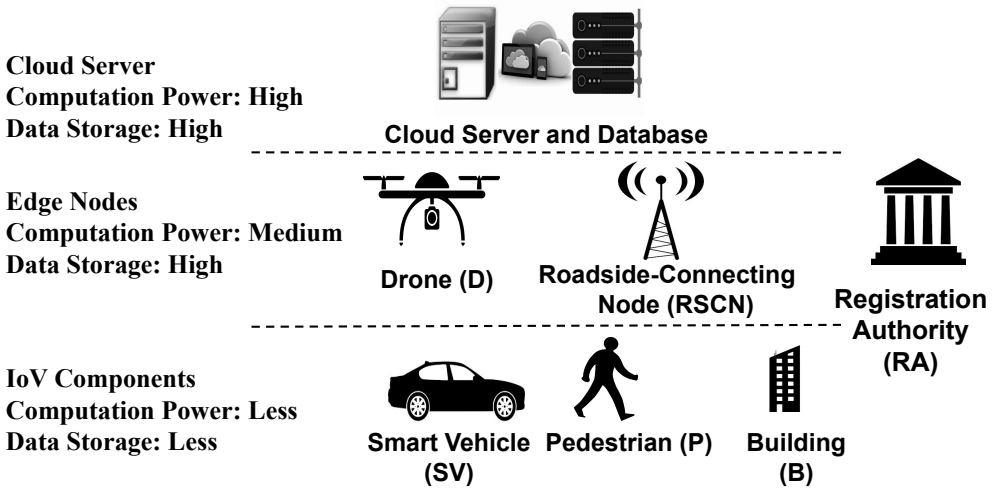


Figure 5.6: Layers and DIoV components of *SECURE* protocol

is considered, as depicted in 5.6. The base layer consists of IoV components such as smart vehicles, pedestrians, and smart buildings. The middle layer includes drones and RSCNs as edge nodes, while the upper layer comprises a cloud server. The introduction of edge computing helps to minimize time complexity by rapidly processing a large volume of real-time data at the local server. This approach reduces network delay and performance issues that arise when sending raw data directly to the centralized server without utilizing edge computing. The key contributions of this work are listed below:

1. The *SECURE* protocol integrates PUFs with one-way hash functions and pseudonymity. This combination provides anonymous secure authentication by generating unique, hardware-based cryptographic keys, leveraging intrinsic device-specific randomness, and ensuring robustness against cloning and tampering. It uses 2 PUF functions and 8 hash functions to perform mutual authentication between vehicle and RSCNs.

2. Another key contribution of the *SECURE* protocol is its architecture, which utilizes drones as mobile edge nodes to assist traditional RSCNs. This integration addresses the limitations of RSCNs, such as constrained communication range, high deployment costs, and single points of failure. By leveraging drones, the protocol enhances line-of-sight connectivity, load balancing, flexibility, and cost-effectiveness, ultimately reducing network delay and improving overall performance.

3. Practical implementation on NVIDIA Jetson Nano and Raspberry Pi 4 demonstrates the protocol's feasibility. Performance analysis indicates significant efficiency gains, with approximately $5X$ lower communication costs and $2X$ faster computation times compared to state-of-the-art methods.

4. The *SECURE* protocol ensures robust security features, including authentication, integrity, confidentiality, availability, non-repudiation, forward and backward secrecy, and resistance to brute-force and replay attacks. Edge computing reduces latency by processing large volumes of real-time data locally, enhancing performance for time-sensitive applications.

5.2.1 Network Model

The network model of *SECURE* is meticulously structured to include smart vehicles as vehicular nodes, drones and RSCNs functioning as edge nodes, RA, and a Cloud Server. This architecture ensures robust, secure, and efficient communication within the DIoV network. RA acts as the foundation of trust, responsible for the secure registration of vehicles, drones, and RSCNs. It maintains a comprehensive database of registered entities, encompassing their identities, cryptographic keys, and other pertinent data. The RA is also empowered to trace unauthorized vehicles and revoke their certificates in response to security breaches or theft, thus ensuring the integrity and security of the network. Drones and RSCNs serve as intelligent edge nodes within the *SECURE* framework. Drones are equipped with PUF devices, Inertial Measurement Units (IMUs) for precise positioning, high-capacity batteries for extended operational periods, advanced flight controllers for accurate navigation, and robust communication modules. These drones are deployed within zones to manage local connectivity. They facilitate essential functions such as bridging communication between distant RSCNs, transferring data to the cloud server for predictive analytics, and executing *SECURE* protocol. RSCNs, strategically positioned along road networks, forming regions and act as stationary edge

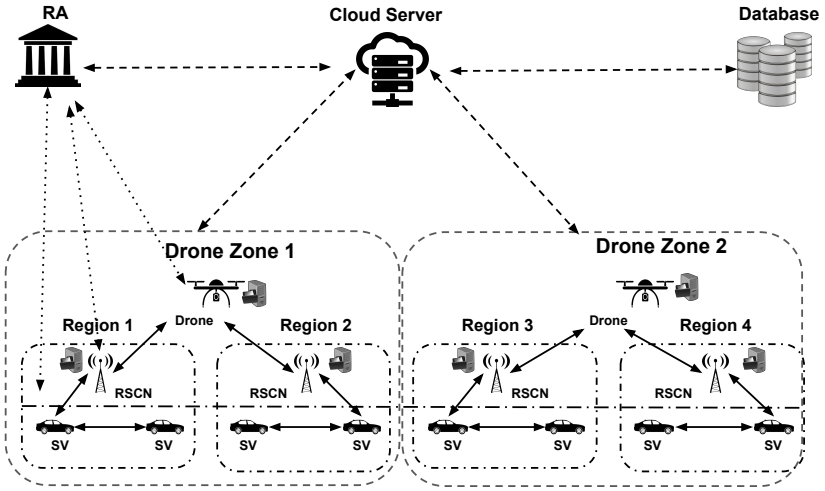


Figure 5.7: Network model of *SECURE* considering RSCN and drones as edge devices

nodes. These nodes are also having PUF chips, wireless communication modules, and substantial computational resources. RSCNs manage communication with drones, vehicles, and the cloud server, generating and storing PUF keys to secure communications. By partitioning the road into distinct regions, RSCNs enable efficient local management of the IoV network. This ensures prompt handling of communication requests from vehicles within their designated areas. Smart Vehicles, acting as dynamic nodes, are equipped with OBUs, PUF devices, wireless communication modules, and computational resources. These vehicles utilize the IEEE 802.11p protocol for secure communication and interact with the corresponding RSCNs in their regions. Each vehicle undergoes a rigorous pre-registration process. During this process, they receive a smart card embedded with a unique ID and password, which facilitates secure authentication and communication with RSCNs, drones, and other IoV components. The cloud server functions as the central entity overseeing the management of PUF keys, digital certificates, and authentication requests. This ensures centralized control and coordination of the network's security infrastructure. The hierarchical network model, depicted in Figures 5.7, illustrates the deployment and interaction of these components. This model showcases protocol's capability to enhance the security, efficiency, and resilience of the DIoVs network.

5.2.2 Proposed Scheme: *SECURE*

Before explaining the detailed procedure, the widely accepted CK-adversary model[181], described in subsection 3.1.1, is considered along with the DY-model[214]. In these models, $\mathbb{A}$ can intercept ongoing communication and may modify, delete, or inject incorrect information. In *SECURE*, the registration of each entity in the DIoV network is initially performed, followed by authentication. The required notations to explain the internal process of *SECURE* and their meanings are summarized in Table 5.8. The registration process for the vehicle, RSCNs, and drones, that is based on CRP mechanism using PUF devices, is described as follows:

Table 5.8: Notations and their explanations

Notations	Explanations
ID_d	Drone's identity
Req_d	Registration request from drone
AID_d	Unique Anonymous Identity for drone
C_d	Random challenge generated by RA for drone
R_d	Generated response corresponding to C_d
C_d^{sync}	Set of re-synchronization challenges generated by RA for drones
R_d^{sync}	Set of re-synchronization responses for C_d^{sync}
$\phi_d()$	PUF function of drone
Req_r	Registration request from RSCN
C_r	Random Challenge generated by RA for RSCN
R_r	Generated response corresponding to C_r
C_r^{sync}	Set of re-synchronization challenges generated by RA for RSCN
R_r^{sync}	Set of re-synchronization responses for C_r^{sync}
UID_r	Unique identity generated by RA for RSCN
VID	Identity of a vehicle
Req_V	Registration request from vehicle
C_V	Random challenge generated by RA for a vehicle
R_V	Generated response corresponding to C_V
C_V^{sync}	Set of re-synchronization challenges generated by RA for vehicle
R_V^{sync}	Set of re-synchronization responses for C_V^{sync}
AID_V	Unique Anonymous Identity for vehicle
$\phi_V()$	PUF Function of vehicle

Smart vehicle registration: As depicted in the Algorithm 15, vehicle V sends VID and Req_V via a secure medium to the RA. The RA then generates a random challenge, C_V, and a set of re-synchronization challenges, $C_V^{sync} = \{c_1, c_2, ..., c_n\}$, to avoid DoS attacks and synchronization issues. These challenges are sent to the corresponding vehicle. Upon receiving the challenges, V uses PUF function, $\phi_V()$, to generate the required responses: $R_V = \phi_V(C_V)$ and $R_V^{sync} = \phi_V(C_V^{sync})$. These responses are sent to the RA via a secure medium. Upon receiving the responses, the RA generates a unique anonymous identity for V, denoted as AID_V. The RA sends AID_V to V and stores the information $\{AID_V, VID, (C_V, R_V), (C_V^{sync}, R_V^{sync})\}$. Simultaneously, V saves AID_V in its database. Additionally, the driver of the V inserts his bio-metric thumb impression, $\mathbb{B}_V$, into the PUF device. The PUF function then generates a unique unclonable bio-metric impression: $\mathbf{B}_V = \phi_V(\mathbb{B}_V)$. This bio-metric impression is combined with the password pwd_V selected by the driver to create a strong combined password, expressed as: $PASS = h(\mathbf{B}_V || pwd_V)$.

RSCNs and drone registration: Initially, RSCN sends the registration request (Req_r) to RA through a secure medium. Upon receiving the registration request, RA randomly generates the challenge C_r and a set

Algorithm 15 Smart vehicle registration process

Input: Vehicle's identity VID, registration request from vehicle Req_V
Output: Anonymous identity of vehicle AID_V
1: $C_V \leftarrow$ generate random challenge.
2: $C_V^{sync} \leftarrow$ generate set of re-synchronization challenges.
3: RA sends C_V and C_V^{sync} to V.
4: $R_V \leftarrow \phi_V(C_V)$.
5: $R_V^{sync} \leftarrow \phi_V(C_V^{sync})$.
6: V sends $\{R_V, R_V^{sync}\}$ to RA.
7: $AID_V \leftarrow$ generate unique anonymous identity.
8: RA sends AID_V to V.
9: RA saves $(AID_V), VID, (C_V, R_V), (C_V^{sync}, R_V^{sync})$.
10: V saves AID_V in its database.
11: $\mathbb{B}_V \leftarrow$ bio-metric thumb impression inserted by driver.
12: $pwd_V \leftarrow$ password chosen by the driver.
13: $\mathbf{B}_V \leftarrow \phi_V(\mathbb{B}_V)$.
14: $PWD_V^{comb} = h(\mathbf{B}_V || pwd_V)$.

of re-synchronization challenges C_r^{sync}. The RA sends these challenges $\{C_r, C_r^{sync}\}$ to the RSCN via a secure medium. The RSCN computes a response and a set of re-synchronization responses corresponding to the

Algorithm 16 RSCN registration process

Input: Registration request from RSCN Req_r
Output: Unique identity of RSCN UID_r
1: $C_r \leftarrow$ generate random challenge.
2: $C_r^{sync} \leftarrow$ generate set of re-synchronization challenges.
3: RA sends C_r and C_r^{sync} to RSCN.
4: $R_r \leftarrow \phi_r(C_r)$.
5: $R_r^{sync} \leftarrow \phi_r(C_r^{sync})$.
6: RSCN sends $\{R_r, R_r^{sync}\}$ to RA.
7: $UID_r \leftarrow$ generate unique identity for RSCN.
8: RA sends UID_r to RSCN.
9: RA saves $UID_r, (C_r, R_r), (C_r^{sync}, R_r^{sync})$.
10: RSCN saves UID_r in its database.

received challenges using the PUF function as $R_r = \phi_r(C_r)$ and $R_r^{sync} = \phi_r(C_r^{sync})$. The RSCN then sends $\{R_r, R_r^{sync}\}$ to the RA via a secure medium. Subsequently, RA randomly generates a unique identity for RSCN as UID_r and sends it to the RSCN. The RA also stores $\{UID_r, (C_r, R_r), (C_r^{sync}, R_r^{sync})\}$ in its database for future reference. The registration process of RSCN briefly described in the Algorithm 16. Further, the drone sends its identity (ID_d) and registration request (Req_d) via a secure medium to the RA. The RA then generates a random challenge, C_d, and a set of re-synchronization challenges, $C_d^{sync} = \{c_1, c_2, ..., c_n\}$, to avoid DoS attacks and synchronization issues. These challenges are sent to the corresponding drone. Upon receiving the challenges, drone uses its PUF function (ϕ_d) to generate the required responses: $R_d = \phi_d(C_d)$ and $R_d^{sync} = \phi_d(C_d^{sync})$. These responses, as depicted in the Algorithm 17, are then sent to the RA via a secure medium. Upon receiving the responses, the RA generates a unique anonymous identity for drone as AID_d. The RA sends AID_d to the corresponding drone and saves $\{AID_d, ID_d, (C_d, R_d), (C_d^{sync}, R_d^{sync})\}$ in its database. Simultaneously, drone saves AID_d.

The authentication process occurs between the drone and RSCN, as well as between the RSCN and vehicle. Figure 5.8 illustrates the procedure for handling the authentication requests. Generally, drones and RSCNs have limited storage capacity. The RSCN uses its buffer to accept requests from different vehicles.

Algorithm 17 Drone registration process

Input: Drone's identity ID_d, registration request from drone Req_d
Output: anonymous identity of drone AID_d

1: $C_d \leftarrow$ generate random challenge.
2: $C_d^{sync} \leftarrow$ generate set of re-synchronization challenges.
3: RA sends C_d and C_d^{sync} to drone.
4: $R_d \leftarrow \phi_d(C_d)$.
5: $R_d^{sync} \leftarrow \phi_d(C_d^{sync})$.
6: Drone sends $\{R_d, R_d^{sync}\}$ to RA.
7: $AID_d \leftarrow$ generate unique anonymous identity.
8: RA sends AID_d to drone.
9: RA saves $\{AID_d, ID_d, (C_d, R_d), (C_d^{sync}, R_d^{sync})\}$.
10: Drone saves $\{(AID_d)\}$ in its database.

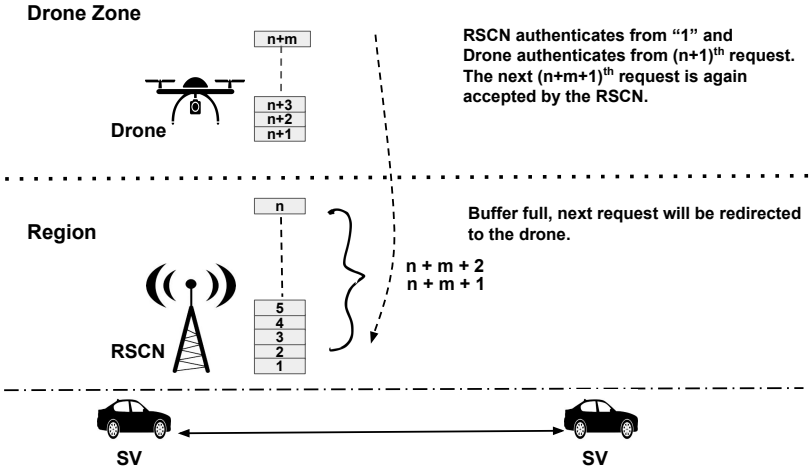


Figure 5.8: Requests handling process of RSCN and drone during authentication

Sometimes, the RSCN may receive multiple requests simultaneously. In such cases, the RSCN stores these requests in its buffer. If the number of requests exceeds the RSCN's buffer capacity, additional requests are handled directly by the drone. Once the drone's buffer is also full, subsequent requests are redirected to the RSCN, which, by then, would have processed the initial requests and freed up buffer space. This situation is an exception and occurs only when the number of vehicles on the road surpasses the maximum limit for a smooth authentication process. The scheme is designed to manage this exceptional case effectively. The mutual authentication processes between the drone and RSCN, and between the RSCN and vehicles, are explained as follows:

Mutual authentication between drone and RSCN: The authentication between drone and RSCN is done with the help of RA. Initially, the drone generates a nonce, η_d, and sends M_1 to RA, which consists of η_d and AID_d. Upon receiving M_1, the RA generates a nonce, η_{ra}, and computes $\eta_{ra}^{'}$ and δ_1. Then, it sends M_2 to the drone which consists of C_d, $\eta_{ra}^{'}$, and δ_1 as parameters. The drone receives M_2 and performs further computations. If δ_1 and δ_1^* are the same, as shown in the Figure 5.9, then only further procedure continues. The drone further performs some computations and sends M_3 to the RA, which consists of $R_d^{'}$, $UID_r^{'}$, and δ_2. Upon receiving M_3, RA calculates unique identity UID_r and sends M_4 with the values such as AID_d, a_1, SK_r, C_r, and δ_3 to the RSCN. After receiving M_4, the RSCN derives the session key SK along with intermediate variables and sends M_5 with $R_r^{'}$ and δ_4 to RA. The RA receives M_5 and calculates SK_d for

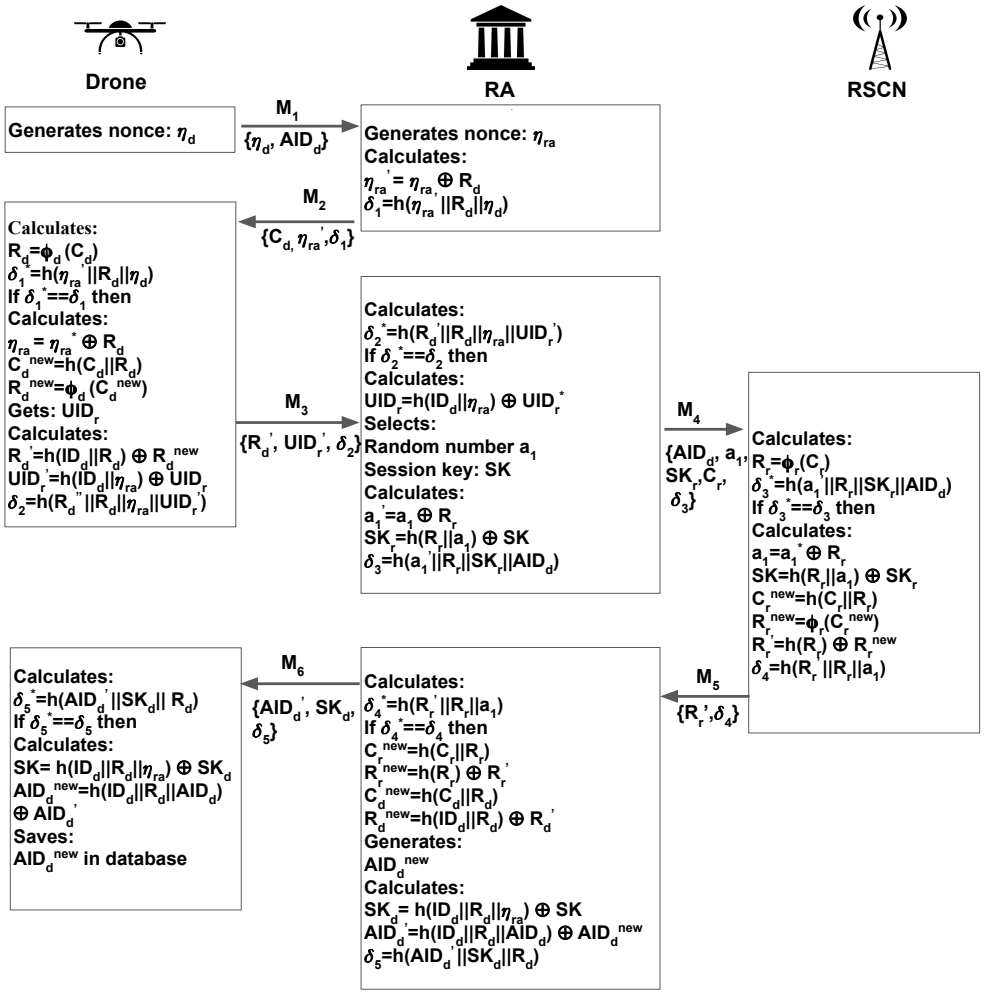


Figure 5.9: Mutual authentication between drone and RSCN

drone. After the computations, the RA sends M_6 with $AID_d^{'}$, SK_d, and δ_5 as parameters to the drone. The drone recomputes the value of δ_5^* and compares it with δ_5. If it matches then computes SK and AID_d^{new}. Finally, the drone saves the value of AID_d^{new} in the database.

Mutual authentication between RSCN and smart vehicle: The authentication between vehicle and RSCN is also done with the help of RA. Initially, the vehicle generates a nonce, η_V, and sends M_7 to RA which consists of η_V and AID_V. Upon receiving M_7, the RA generates a nonce, η_{ras} and computes $\eta_{ras}^{'}$ and δ_6. Then, it sends M_8 to the vehicle which consists of C_V, $\eta_{ras}^{'}$, and δ_6 as parameters. The vehicle receives M_8 and performs further computations as illustrated in the Figure 5.10. If δ_6 and δ_6^* are the same, then only further procedure continues. The vehicle further calculates response R_V along with other intermediate variables and sends M_9, which consists of $R_V^{'}$, $UID_r^{'}$, and δ_7 to the RA. Upon receiving M_9, RA derives unique identity for RSCN, UID_r, and sends M_{10} with the values such as AID_V, a_1, SK_r, C_r, and δ_8 to the RSCN. After receiving M_{10}, the RSCN calculates session key SK along with new CRP pairs and sends M_{11} with $R_r^{'}$ and δ_9 to the RA. The RA receives M_{11} and calculates SK_V for the corresponding vehicle. After the computations, the RA sends M_{12} with $AID_V^{'}$, SK_V, and δ_{10} as parameters to the vehicle. The vehicle

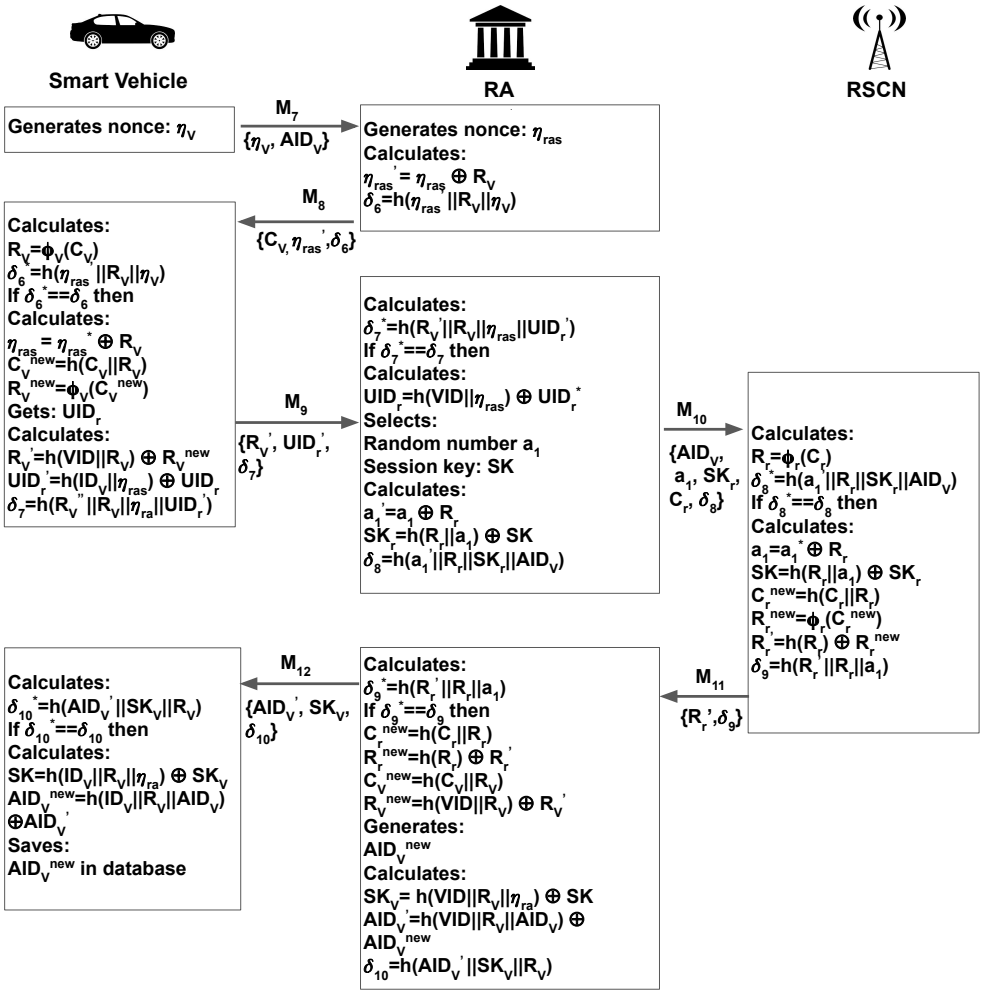


Figure 5.10: Mutual authentication between vehicle and RSCN

recomputes the value of δ_{10}^* and compares it with δ_{10}. If it matches then computes SK and AID_V^{new}. Finally, the vehicle saves the value of AID_V^{new} in the database.

Resource Optimization and Load Management in *SECURE*: To ensure reliable performance in dynamic and high-demand IoV environments, the *SECURE* protocol incorporates a multi-layered resource allocation strategy that balances load between edge nodes and the centralized server. These mechanisms help maintain consistent authentication response times and prevent bottlenecks in dense vehicular zones.

- **Adaptive Threshold-Based Authentication Offloading:** Authentication requests in *SECURE* are distributed dynamically based on a computed threshold T_{auth} that considers the real-time load at the edge nodes (L_{EN}) and centralized server (L_{CS}), along with the priority of the request (P_{auth}). The decision is governed by:

$$T_{\text{auth}} = f(L_{\text{EN}}, L_{\text{CS}}, P_{\text{auth}}) \tag{5.15}$$

If $L_{EN} \leq T_{auth}$, authentication is handled at the edge to reduce latency and conserve bandwidth. Otherwise, the request is escalated to the central server. This allows the system to prioritize critical authentication tasks while distributing computational workload intelligently.

- **Drone-Assisted Load Redistribution:** SECURE further enhances scalability through the integration of Drone-Assisted Verification Units (DAVUs). These drones serve as supplementary edge devices equipped with lightweight PUF authentication modules, allowing them to verify vehicular identities and relay only high-priority or ambiguous requests to the central server. This design minimizes communication congestion and ensures uninterrupted service in overloaded edge zones.

By integrating threshold-based offloading with mobile drone relays, SECURE achieves a high level of elasticity in resource management. This dual mechanism allows the system to sustain real-time authentication and data integrity, even during high-density vehicle surges or edge buffer overflow scenarios. These strategies make SECURE highly adaptable to both urban and highway environments, where vehicular loads fluctuate rapidly.

5.2.3 Performance Analysis and Simulation Results

MIRACL and Java 8 are used to execute different cryptographic operations on Jetson Xavier NX and Raspberry Pi 4 systems. In the *SECURE* protocol, two types of devices are considered for the experiments: NVIDIA Jetson Xavier NX and Raspberry Pi 4. According to the network model, three different entities

Table 5.9: Hardware specifications of drone, smart vehicle, and RSCN

Components	Specifications	Details
Drone and smart vehicle	**System**	Raspberry Pi 4
	Variants	4 GB (Smart Vehicles), 8 GB (Drones)
	CPU	Quad-core ARM Cortex-A72, 1.5 GHz
	RAM	4 GB / 8 GB LPDDR4-3200 SDRAM
	Storage	MicroSD card slot (up to 2 TB)
	Connectivity	Gigabit Ethernet, 802.11ac wireless, Bluetooth 5.0
	Ports	2x USB 3.0, 2x USB 2.0, 2x micro-HDMI (up to 4Kp60)
	GPIO	40-pin header, compatible with Raspberry Pi HATs
RSCN	**Device**	Jetson Xavier NX
	Performance	Up to 21 TOPS (15W), up to 14 TOPS (10W)
	CPU	6-core NVIDIA Carmel 64-bit ARMv8.2, 1400MHz
	Cache	6MB L2, 4MB L3
	GPU	384-core NVIDIA Volta, 1100MHz, 48 Tensor Cores
	RAM	8GB 128-bit LPDDR4x, 51.2GB/s bandwidth
	Storage	16GB eMMC 5.1
	Displays	Dual 4K (DisplayPort/HDMI)
	Ethernet	10/100/1000 BASE-T
	Ports and I/O	USB 3.1 (10Gbps), 3x USB 2.0, 2x PCIe Gen 3, UART, SPI, I2C, I2S, CAN, GPIOs

participate in the communication: drone, vehicle, and RSCN. The Raspberry Pi 4 with 8GB of RAM is used as the drone, while the Raspberry Pi 4 with 4GB of RAM is used as the vehicular node. The Jetson Xavier NX represented the RSCN. The hardware specifications of these devices are detailed in Table 5.9. In addition to these devices, a 128-bit strong SRAM PUF circuit is included as a PUF chip, integrated on an MSP430 micro-controller machine with a 798 MHz CPU. The experimental setup for the *SECURE* scheme is illustrated in Figure 5.11, which includes one drone node creating a zone, two RSCNs for two different

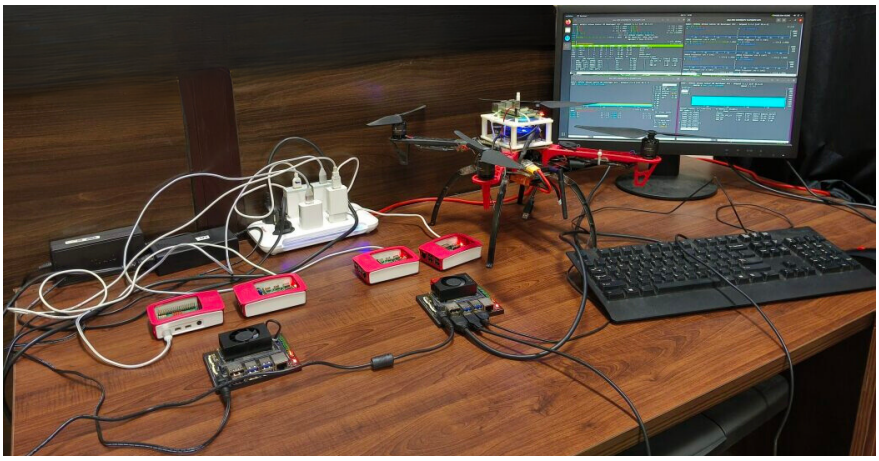


Figure 5.11: Experimental setup for the execution of *SECURE* algorithm

regions, and two vehicular nodes in each RSCN region. The performance analysis is conducted from three perspectives: computation cost, energy consumption, and communication cost. The detailed analysis for these parameters is as follows:

Computation cost: The *SECURE* protocol's computation cost can be effectively understood by analyzing the execution times of various cryptographic operations on different hardware platforms, the specific cryptographic equations used in the protocol, and a comparative analysis with other state-of-the-art schemes. The execution times for cryptographic operations such as hash operations (T_h), ECC-point multiplication (T_m), ECC-point addition (T_a), exponentiation (T_e), and PUF operations ($T_{PUF()}$) were measured on Jetson Xavier NX (RSCNs), Raspberry Pi 4-4GB (vehicles), and Raspberry Pi 4-8GB (drones). The results, in Table 5.10, indicate that the Jetson Xavier NX platform exhibits significantly lower execution times across all cryptographic operations compared to the Raspberry Pi platforms. For example, a hash operation takes 0.52 ms on

Table 5.10: Execution time (ms) of each cryptographic operation on different platforms

Cryptographic operations	**Jetson Xavier NX (RSCNs)**	**Raspberry Pi 4-4GB (vehicle)**	**Raspberry Pi 4-8GB (Drone)**
T_h	0.52	1.82	1.08
T_m	3.27	7.68	7.02
T_a	1.43	9.86	8.06
T_e	2.07	10.56	9.46
$T_{PUF()}$	0.75	2.04	1.26

Jetson Xavier NX, while it takes 1.82 ms and 1.08 ms on the Raspberry Pi 4-4GB and Raspberry Pi 4-8GB respectively. The cryptographic equations, in Table 5.11 used in the *SECURE* protocol for authentication between drones and RSCNs, and between vehicles and RSCNs, further illustrate the computation cost. The

Table 5.11: Cryptographic equations of vehicle, drone, and RSCN

Authentications	**Smart vehicle**	**Drone**	**RSCN**
Between Drone and RSCN	—	$2T_{PUF()}+8T_h$	$2T_{PUF()}+4T_h$
Between vehicle and RSCN	$2T_{PUF()}+8T_h$	—	$2T_{PUF()}+4T_h$

protocol requires $2T_{PUF()}+8T_h$ for the drone and $2T_{PUF()}+4T_h$ for the RSCN in drone-to-RSCN authenti-

cation. Similarly, vehicle-to-RSCN authentication involves $2T_{PUF()} + 8T_h$ for the vehicle and $2T_{PUF()} + 4T_h$ for the RSCN. These operations leverage the efficient execution capabilities of the hardware platforms to ensure timely processing. Further, the comparison for the computation cost of *SECURE* and state-of-the-art schemes is shown in Table 5.11. In comparison, the *SECURE* protocol stands out for its efficiency and lower computation cost. For instance, Hussain et al.[148] and Wu et al.[149] require more complex operations involving exponentiation and a higher number of hash operations, which increases their computation cost. In contrast, the *SECURE* protocol requires $8T_{PUF()} + 24T_h$, making it more efficient. Additionally,

Table 5.12: Comparison of *SECURE* with state-of-the-art schemes

Schemes	Cryptographic equations	Communication cost (bits)	Edge-computing	PUF Usage
Khan et al. [147]	$4T_m$	336	No	No
Hussain et al. [148]	$2T_{PUF()} + 2T_e + 31T_h$	2061	No	No
Wu et al. [149]	$T_m + 2T_e + 30T_h$	6720	No	No
Chaudhry et al. [150]	$13T_h + 7T_e$	2624	No	No
SECURE	$8T_{PUF()} + 24T_h$	384	Yes	Yes

the *SECURE* protocol's use of edge-computing and PUFs enhances its performance and security, providing a distinct advantage over other schemes that do not employ these technologies. Figure 5.12 graphically displays the comparison of computation cost for the vehicle, RSCN, and drone separately. It is evident that, in *SECURE*, the vehicle takes 52.32 ms, the RSCN takes 4.6 ms, and the drone takes 33.6 ms. All other competitive schemes, except for Khan et al.[147], require more time than *SECURE*. In [147], the edge computing concept is not integrated into the protocol, leading to scalability issues compared to *SECURE*. These scalability issues are a primary concern in a DIoVs environment.

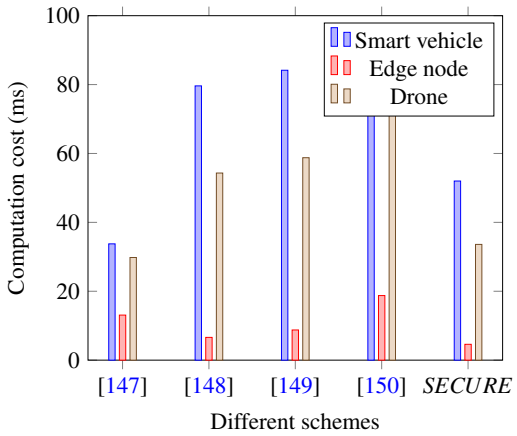


Figure 5.12: Computation cost (ms) comparison of *SECURE* with state-of-the-art scheme

Energy consumption and communication cost: Energy is one of the primary factors to be preserved when dealing with DIoVs. A small electrical UAV typically requires around 20 to 200 watts per kilogram to fly. For instance, an electrical UAV with a 5-meter wingspan may need a power of 100 watts for a Take-Off Weight (TOW) of 4.5 kg. The energy consumption is directly proportional to the computation cost of the scheme. Therefore, to calculate the energy consumption, the computation cost mentioned in Table 5.12 is used. Figure 5.13a graphically displays the energy consumption of *SECURE* compared to other competitive schemes, with energy measured in millijoules (mJ). The communication cost is the total

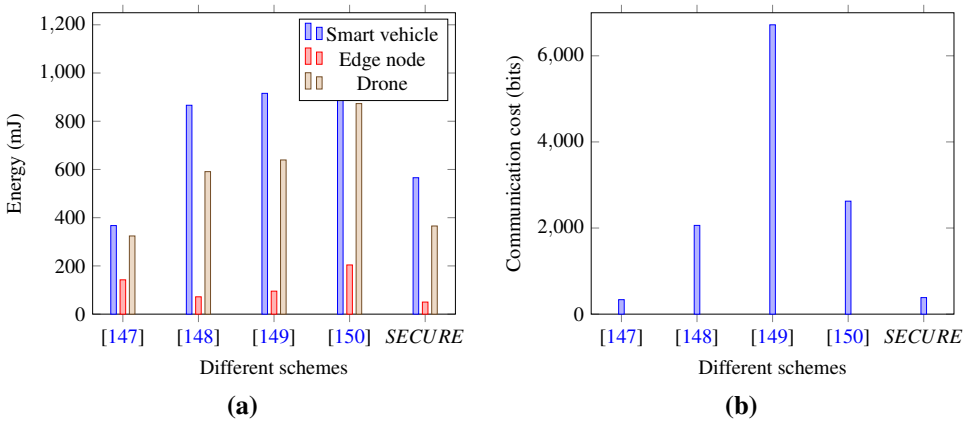


Figure 5.13: Performance comparison of *SECURE* with state-of-the-art in terms of (a) Energy consumption (mJ), and (b) Communication cost (bits)

number of packets to be transferred or transmitted from one node to another. In *SECURE*, 128 bits are considered for the challenge-response pair, 256 bits for hash functions, 8 bits for identity, and 48 bits for timestamps. Figure 5.13b shows the communication cost comparison of *SECURE* with other competitive schemes. It is evident that *SECURE* requires fewer bits than schemes presented by Hussain et al.[148], Wu et al.[149], and Chaudhry et al.[150], and only 48 bits more than Khan et al.[147].

5.3 SUMMARY OF THE WORK

In this chapter, the integration of PUFs and edge computing is explored to enhance the security and efficiency of IoV and DIoV networks. The primary focus is on the development and implementation of the SECURE and PESPI protocols, both addressing critical challenges in modern vehicular networks. PESPI extends the concept of secure vehicular networks by integrating lightweight PUF functions with edge computing. This protocol optimizes communication overhead, storage cost, and computation cost, while addressing security issues such as physical attacks, data leakage from side-channel attacks, and DoS attacks. PESPI employs a hybrid three-layered architecture that balances centralized and distributed environments, mitigating bottleneck issues associated with central authority failures. The protocol uses PUFs for the random generation of secret parameters, eliminating the need to store sensitive data on OBUs and significantly reducing storage costs. Additionally, the integration of edge computing nodes allows for the local processing of large volumes of real-time data, reducing latency and improving bandwidth efficiency. The *SECURE* protocol integrates drones as mobile edge nodes to complement traditional RSCNs. This approach addresses the limitations of RSCNs, such as constrained communication range, high deployment costs, and vulnerability to single points of failure. Drones enhance line-of-sight connectivity, load balancing, and flexibility, thereby improving overall network performance. The protocol employs PUFs to generate unique, hardware-based cryptographic keys, leveraging intrinsic device-specific randomness to ensure robust security against cloning and tampering. The three-layered architecture of SECURE, consisting of IoV components at the base layer, drones and RSCNs as intelligent edge nodes in the middle layer, and a centralized cloud server at the upper layer, ensures comprehensive coverage, reduced latency, and enhanced resilience.

Performance analysis and simulations of both protocols demonstrate significant improvements in computation time and communication cost, with PESPI achieving reductions of up to 84.96% and 57%, respectively. *SECURE* protocol reduces the communication cost maximum upto 85%. The formal and informal security analyses confirm the robustness of these protocols against various cyber attacks, including impersonation, replay, and man-in-the-middle attacks. ProVerif tool verification further validates the protocols' resilience to potential security threats.