4

Secure Data Handling using Tamper-Proof Devices

The IoV faces significant challenges due to its diverse and dynamic characteristics. One of the main challenges is the vast amount of data generated by large-scale IoV communications[74]. This data, which includes information from various sensors, GPS, and heterogeneous communications such as V2V, V2I, and V2X, etc., must be handled securely[12]. Secure data handling involves storing the generated data safely and ensuring its quick retrieval, especially in delay-sensitive networks where timely access to information is crucial[75]. To address these challenges, the concept of V-cloud has been introduced[44, 77]. V-cloud aims to provide a centralized platform for managing vast amounts of generated data. By leveraging cloud computing technologies, V-cloud offers scalable storage solutions and computational resources, enabling the efficient processing and analysis of vehicular data[62]. However, the centralized nature of V-cloud introduces its own set of challenges. These include potential bottlenecks in data transmission, latency issues, and vulnerabilities to cyber-attacks. The reliance on a central server means that any disruption can have widespread implications, affecting the entire network's ability to function effectively[76].

To mitigate these challenges, integrating edge computing with V-cloud has been proposed in this thesis as a promising solution. An edge-computing enabled secure communication, data storage and quick data retrieval method, *SecEdge*[45] has been proposed leveraging the advanced features of edge computing[72]. Edge computing involves deploying computational resources closer to the data source, thereby reducing latency and alleviating the load on central servers. By processing data at the edge of the network, closer to where it is generated, edge computing can enhance the efficiency and responsiveness of IoV systems[149]. This integration offers several advantages in terms of secure data handling. Firstly, edge computing improves data security by reducing the amount of data transmitted over long distances to central servers, thereby minimizing exposure to potential cyber threats. Data can be processed locally, ensuring that sensitive information remains within the confines of the vehicular network. Secondly, the distributed nature of edge computing enhances fault tolerance and reliability[63]. In the event of a central server failure, edge nodes can continue to operate independently, ensuring uninterrupted service. Thirdly, edge computing enables faster data retrieval and processing, which is critical in delay-sensitive applications such as emergency response systems. By reducing the dependency on centralized servers, edge computing ensures that data can be accessed and utilized promptly, facilitating timely decision-making[72]. Apart from these advantages, the primary contributions of *SecEdge* are discussed as follows:

- A distributed architecture for IoV communication has been proposed that effectively integrates the V-cloud concept with edge computing by transforming RSCN into robust and capable edge nodes[45].

This hybrid architecture facilitates faster, location-based real-time query processing, thereby enhancing the responsiveness and efficiency of the IoV network.

- The *SecEdge* scheme is introduced, which ensures secure inter-vehicle and intra-vehicle communication through the implementation of lightweight one-way hash functions[71]. Additionally, *SecEdge* scheme provides mechanisms for secure storage and retrieval of data using SSE[73] within edge nodes and the V-cloud, addressing the critical need for data confidentiality and integrity in IoV networks.
- A comprehensive security evaluation of the proposed *SecEdge* scheme is conducted, demonstrating its resilience against various security and privacy threats such as replay attacks, data modification, impersonation, repudiation, man-in-the-middle attacks, and location privacy breaches. The robustness of *SecEdge* is formally verified using the ROM[79] and ProVerif software tool[89], providing a strong theoretical foundation for its security claims.
- To assess the performance of the *SecEdge* scheme, simulations are performed using the SUMO and NS-2 tools[88]. The execution of the code is done on both Desktop and Raspberry Pi 4 to evaluate computational cost and search time complexity. The results validate the efficiency and scalability of *SecEdge* in real-world IoV scenarios.

4.1 PROPOSED HYBRID ARCHITECTURE FOR INTERNET OF VEHICLES

The proposed hybrid architecture for the IoV leverages RSCNs as highly capable nodes, structured into domains, and functioning as edge nodes. The IoV scenario under consideration consists the RA, Key Generation Center (KC), a centralized V-cloud for storage, RSCNs as edge nodes, and OBU having TPD[45]. This hybrid IoV scenario, integrating edge computing with V-cloud, is depicted in Figure 4.1. A detailed discussion for the distinct components of proposed IoV architecture is as follows:

Registration authority: RA serves as a highly secure and reliable entity responsible for the initial registration of vehicles at the point of purchase. Upon registration, the RA transmits all relevant vehicle information to both the V-cloud and edge nodes. This dual storage mechanism ensures redundancy and data availability[74]. In instances where the V-cloud fails to retrieve specific information due to data loss or corruption, it can issue a retrieval request to the RA. Additionally, the RA maintains the security by tracking and identifying malicious nodes be revealing the original identity using its comprehensive repository of vehicle data.

Key generation center: KC is a secure entity tasked with generating essential system parameters and cryptographic keys, including public and private keys, for vehicles and other IoV components. The KC ensures the secure transmission of these cryptographic parameters to the respective entities, maintaining the integrity and confidentiality of the keys during transit.

Vehicular cloud: The V-cloud operates as the centralized, robust component within the proposed network model. It is responsible for several critical functions, including validating information received from vehicles and edge nodes, securely storing vast amounts of data, and processing search queries[44]. Upon receiving a query request from a vehicle, the V-cloud searches its extensive database and returns the required information through a secure communication channel. The V-cloud's centralized nature allows for comprehensive data management and high-level oversight of the network.

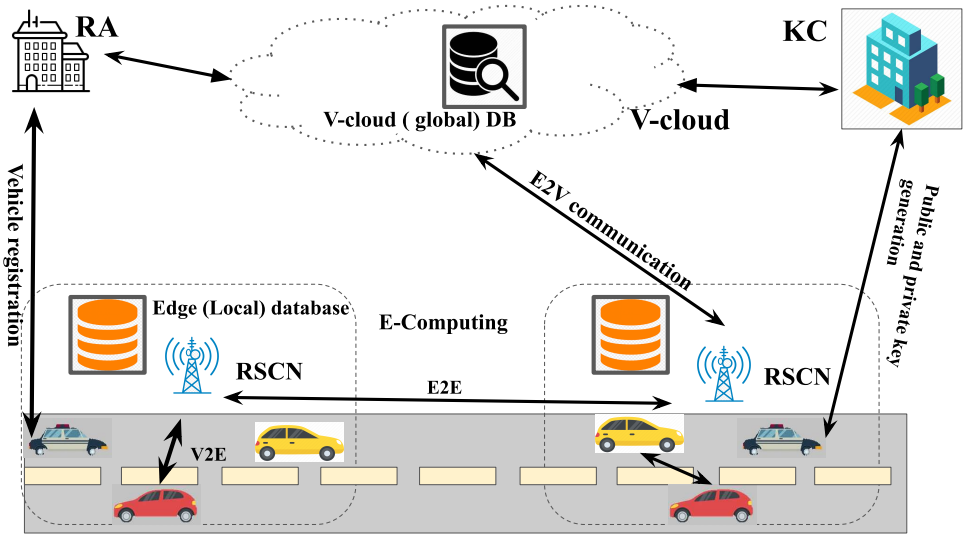

Figure 4.1: The proposed hybrid architecture for IoV leverages RSCN as highly capable nodes, structured into domains, and functioning as edge nodes

Edge nodes: In this hybrid IoV network, RSCNs are segmented into different domains, each acting as an edge node for its specific domain. Vehicles and other domain-specific components (denoted as D_i) communicate directly with their respective edge node for data requests and queries. The edge node first searches its local database for the requested information[63]. If the information is not available locally, the edge node escalates the query to the V-cloud, retrieves the necessary data, and subsequently responds to the initial request. This distributed configuration of edge nodes interconnected across various domains fosters a decentralized yet coordinated network environment, mitigating the bottleneck issues inherent in purely centralized systems.

On-board units: OBUs are integral to the data collection and processing functions within vehicles. Each OBU is equipped with the TPD to ensure data confidentiality and integrity. These devices utilize biometric parameters to generate message integrity checks and authentication tokens, thus enhancing the security of communications and data exchanges within the IoV network. The OBUs ensure that sensitive data remains protected from unauthorized access and manipulation during transit and processing.

4.2 PROPOSED SCHEME: *SECEDGE*

Along with the above architecture, algorithms for secure V2V, Vehicle-to-Edge (V2E), and Edge-to-Cloud (E2C) communications, secure data storage at the edge (locally)/cloud (globally), and secure searching/ retrieval of data from edge/cloud have been proposed in this thesis[45]. The scheme is designed using the ECC and SSE based encryption technique to preserve security as they are comparatively efficient on-board computation. The widely accepted CK-adversary model[181], described in subsection 3.1.1, is considered. This section explains the following phases of *SecEdge*. For better understanding, Table 4.1 defines all the variables used in *SecEdge*.

Table 4.1: Required symbols and their explanation

Symbols	Explanation
$Mkey_{kc}$	Master key for KC
$Mkey_{RA}$	Master key for RA
$Mkey_E$	Master key for edge Nodes
$Mkey_{V_i}$	Master key for vehicle V_i
$Mkey_{V_i}^{kc}$	Master key for V_i chosen by KC.
$Pubkey_{kc}$	Public key for KC
$Pubkey_E$	Public key for edge nodes
$Pubkey_{V_i}$	Public key for vehicle V_i
$Pubkey_{V_i}^{kc}$	Public key for V_i calculated by KC.
g	Generator of additive group G
$\{w_i, u_i\}$	Secure parameters of TPD
$Prikey_{V_i}$	Private key for vehicle V_i
$\{Prikey_{V_{i1}}^{kc}, Prikey_{V_{i2}}^{kc}\}$	Set of private keys for vehicle V_i calculated by KC.
L_i, η_i, δ_i, Q_i, λ_i, μ_i, ρ_i	Intermediate variables used for the required calculations.
τ_i, τ_j	Timestamp variables
VID	Vehicle identity
$PVID$	Pseudo identity of vehicle
M	Generated message

System initialization and key generation: This stage is carried out by the KC to generate its public and private key combination along with the initial parameters of the system. Initially, KC chooses an additive group $\mathbb{G}$ along with two adequately large prime numbers p and q such that the order of $\mathbb{G}$ is q. The following equation determines points on the elliptic curve: $Y^2 = X^3 + AX + B(\mod p)$. KC performs the following steps to initialize the whole system:

- KC selects its secure master key $Mkey_{kc}$ randomly such that $Mkey_{kc} \in Z_q^*$ and calculates public key $Pubkey_{kc}$ respectively: $Pubkey_{kc} = Mkey_{kc}.g$, Where g is generator of an additive group $\mathbb{G}$.
- In the proposed IoV network, RSCNs are deployed domain-wise and considered as edges for data stored locally. Hence, for each edge, KC chooses $Mkey_E^i$ as their master key and calculates their respective public keys $Pubkey_E^i$: $Pubkey_E^i = Mkey_E^i.g$
- KC publishes $\{Pubkey_{kc}, Pubkey_E^i, \mathbb{G}, g\}$ as the public parameters. KC also sends all parameters to the edges and V-cloud through a secure channel to store these values in the database.
- The vehicle registration is done offline by the RA. RA calculates $w_i = VID_i \oplus h(Mkey_{RA}.Pubkey_{kc})$ and $u_i = h(VID_i)$. RA store these values $\{w_i, u_i\}$ in TPD, (TPD_i) of V_i. Where, VID_i is the real identity of V_i, $h() : \{0,1\}^* \rightarrow Z_q^*$, $Mkey_{RA}$ is the master key of RA that is selected randomly by RA. Thus the registration process of vehicles gets done by RA.
- Before sending a message to the edge, vehicles need their partial keys for the generation of their private and public keys. For partial key generation, vehicle V_i sends u_i to KC through a secure medium. After obtaining u_i, KC verifies it in the list of blacklisted vehicles received from RA through a secure channel. When a corresponding entry exists in the list, KC revokes it for further communication. On the contrary, KC initiates generating the partial key for V_i.

- Now, KC chooses $Mkey_{V_i}^{kc} \in Z_q^*$ as the master key for vehicle V_i and calculates the public key for the same as $Pubkey_{V_i}^{kc} = Mkey_{V_i}^{kc}.g$. Further, KC generates a set of private partial keys as:

$$Prikey_{V_{i1}}^{kc} = Mkey_{kc}.u_i(\quad \mod q), \tag{4.1}$$

$$Prikey_{V_{i2}}^{kc} = h(u_i.Pubkey_{kc}) \oplus (Mkey_{V_i}^{kc} + Mkey_{kc}). \tag{4.2}$$

  KC sends $\{Prikey_{V_{i1}}^{kc}, Prikey_{V_{i2}}^{kc}, Pubkey_{V_i}^{kc}\}$ to corresponding vehicle V_i securely.

- Now, vehicle V_i initiates the private key generation process using the private partial keys received from KC. V_i picks its master key as $Mkey_{V_i} \in Z_q^*$ randomly to generate its public and private keys. Now, V_i calculates:

$$Pubkey_{V_i} = Mkey_{V_i}.g, \tag{4.3}$$

$$\begin{aligned} Prikey_{V_i} &= h(Pubkey_{kc} \oplus Pubkey_{V_i}).(Prikey_{V_{i2}}^{kc} \oplus h(u_i.Pubkey_{kc}))modq, \\ &= h(Pubkey_{kc} \oplus Pubkey_{V_i}).(Mkey_{V_i}^{kc} + Mkey_{kc})(\quad \mod q). \end{aligned} \tag{4.4}$$

  Vehicle V_i saves $\{Prikey_{V_{i1}}^{kc}, Prikey_{V_i}\}$ as its private keys into the corresponding TPD and declares its public keys as $\{Pubkey_{V_i}^{kc}, Pubkey_{V_i}\}$ to the network.

Secure Communication: In *SecEdge*, edge nodes are divided into the domain. When V_i comes in the range of edge E_j in the domain D_j. E_j generates V_i's pseudo identity $PVID_i = \{PVID_i^1, PVID_i^2\}$, that is needed for further communication with other vehicles and the edge of the corresponding domain. After receiving the $PVID_i$ by E_j, V_i tries to broadcast some traffic or safety-related messages M_i. For this, V_i needs to sign M_i to

Algorithm 8 Secure vehicle-to-edge communication

Input: $\{w_i, u_i\}, g, M_i, \tau_i$
Output: set of pseudo identity $\{PVID_i^1, PVID_i^2\}$, signature $sign_i$

Pseudo identity generation by E_j:
  E_j of domain D_j randomly selects $\beta_{ij} \in Z_q^*$.
  E_j calculates a set of pseudo identities for V_i as:
  $PVID_i^1 = \beta_{ij}.g$,
  $PVID_i^2 = w_i \oplus h(u_i||\beta_{ij}||\tau_i)$.
  E_j sends this set of $PVID_i$s to the corresponding V_i.
Signature generation by V_i:
  V_i selects $l_i \in Z_q^*$ randomly and calculates:
  $L_i = l_i.g$,
  $\eta_i = (l_i^{-1} \cdot (Prikey_{V_{i1}}^{kc} + Prikey_{V_i}.h(M_i, Pubkey_{V_i}^{kc}, \tau_i, PVID_i))modq)$,
  $\delta_i = (Mkey_{V_i}^{-1}.Prikey_{V_{i1}}^{kc})(\mod q)$.
  V_i broadcasts $\{M_i, sign_i = \{\eta_i, \delta_i\}, L_i, Pubkey_{V_i}^{kc}, Pubkey_{V_i}, PVID_i, \tau_i\}$ within the domain of E_j.
Timestamp verification to avoid replay attack:
if $\tau_{receiver} - \Delta t > \tau_i > \tau_{receiver} + \Delta t$ **then**
    E_j and other vehicles discard the message.
else
    Calculates $Q_i = h(Pubkey_{kc} \oplus Pubkey_{V_i}) \cdot h(M_i, Pubkey_{V_i}^{kc}, \tau_i, PVID_i) \cdot (Pubkey_{V_i}^{kc} + Pubkey_{kc})$.
end if
if $\eta_i \cdot L_i - \delta_i \cdot Pubkey_{V_i} = Q_i$ **then**
    Signature is verified successfully.
else
    Message is discarded.
end if

guarantee integrity. V_i generates its signature $sign_i = \{\eta_i, \delta_i\}$ with the help of the private and public keys of V_i. Afterward, V_i broadcasts $\{M_i, sign_i, L_i, Pubkey_{V_i}^{kc}, Pubkey_{V_i}, PVID_i, \tau_i\}$ within the domain of edge E_j. Vehicles and edge E_j of the corresponding domain D_j continuously receive some safety and traffic-related messages broadcast by the other vehicle of the domain D_j to get some information about traffic conditions. E_j and all the vehicles of D_j verify the received message $\{M_i, sign_i = \{\eta_i, \delta_i\}, L_i, Pubkey_{V_i}^{kc}, Pubkey_{V_i}, PVID_i, \tau_i\}$ to guarantee that the message is not modified during its journey. When the message verification is done successfully, the received message is saved along with other information related to that vehicle V_i for future monitoring. The processes of $PVID_i$ generation, signature generation, and signature verification are briefly illustrated in the Algorithm 8.

Secure storage at edge node (locally) and V-cloud (globally): As per the proposed IoV architecture, after receiving the data from multiple vehicles of the domain D_j, the edge E_j saves the meaningful data in its local database based on the index of $PVID_i$ and sends the same to the V-cloud for global storage.

Table 4.2: Edge E_j database

Personal Identity	**Encrypted Data**
$PVID_i$={ $PVID_i^1, PVID_i^2$ }	$E_{Pubkey_{kc}}(V_{i,cord}, V_{i,S}, V_{i,T})$
$PVID_{v_{i+1}}$={ $PVID_{i+1}^1, PVID_{i+1}^2$ }	$E_{Pubkey_{kc}}(V_{i+1,cord}, V_{i+1,S}, V_{i+1,T})$
$PVID_{v_{i+2}}$={ $PVID_{i+2}^1, PVID_{i+2}^2$ }	$E_{Pubkey_{kc}}(V_{i+2,cord}, V_{i+2,S}, V_{i+2,T})$
: .	: .

SecEdge can overcome the bottleneck problem that occurs due to the centralized approach and can improve the efficiency of the whole network. Besides that, by introducing secure storage and secure searching, the data leakage problem that occurs due to weak security parameters and misconfigured internal systems can also be overcome. At V-cloud, the obtained data is stored in the database based on the index of different edges, i.e., $E_1, E_2, E_3, ...E_n$. Using this strategy, the searching and response time at the time of the query can be minimized. Usually, the number of vehicles $(V_1, V_2, V_3, \ldots, V_m)$ in the IoV environment is very large

Table 4.3: V-cloud database

E_ID	**D_ID**	**Personal Identity**	**Encrypted Data**
ID_{E_j}	D_j	{ $PVID_i^1, PVID_i^2$ } { $PVID_{i+1}^1, PVID_{i+1}^2$ } { $PVID_{i+2}^1, PVID_{i+2}^2$ } : .	$E_{Pubkey_{kc}}(V_{i,cord}, V_{i,S}, V_{i,T})$ $E_{Pubkey_{kc}}(V_{i+1,cord}, V_{i+1,S}, V_{i+1,T})$ $E_{Pubkey_{kc}}(V_{i+2,cord}, V_{i+2,S}, V_{i+2,T})$: .
$ID_{E_{j+1}}$	D_{j+1}	{ $PVID_i^1, PVID_i^2$ } { $PVID_{i+1}^1, PVID_{i+1}^2$ } { $PVID_{i+2}^1, PVID_{i+2}^2$ } : .	$E_{Pubkey_{kc}}(V_{i,cord}, V_{i,S}, V_{i,T})$ $E_{Pubkey_{kc}}(V_{i+1,cord}, V_{i+1,S}, V_{i+1,T})$ $E_{Pubkey_{kc}}(V_{i+2,cord}, V_{i+2,S}, V_{i+2,T})$: .

compared to the number of edges $(E_1, E_2, E_3, \ldots, E_n)$. Hence, by using this edge indexing approach, the search time of a query can be minimized. Vehicle-related information is saved in encrypted form in the database of the edge (local) and V-cloud (global) using the SSE encryption technique. By doing this, it is ensured that an adversary $\mathbb{A}$ cannot access the data from the database. Tables 4.3 and 4.2 show how the data is saved in the edge and V-cloud databases, respectively. guaranteeHere, $ID_{E_j}, D_j, PVID_i, Encrypt(VM_i)$

are saved based on edge indexing. Algorithm 9 illustrates the process of storing the information at the edge and V-cloud database securely.

Algorithm 9 Secure data storage at edge E_i and V-cloud

Input: ID_{E_j}, domain D_j, $PVID_i$
Output: Secure storage at E_j and V-cloud.
Start
Data storage at Edge E_j (Locally):
    After receiving the message by V_i, E_j of domain D_j saves the data of V_i based on the index of $PVID_i$.
    $V_{i,cord}, V_{i,S}, V_{i,T}$ = Vehicle's geographic location, speed, traffic domain availability.
    Calculates $Encrypt(VM_i) = E_{Pubkey_{kc}}(V_{i,cord}||V_{i,S}||V_{i,T})$.
    E_j saves $Encrypt(VM_i)$.
Data storage at V-cloud (Globally):
    E_j receives M_i from V_i .
    Calculates $\lambda_i = ID_{E_j} \oplus D_j \oplus M_i$ and $\mu_i = h(\lambda_i||M||\tau_i)$.
    E_j sends $\{ID_{E_j}, \lambda_i, \mu_i, \tau_i\}$ to the V-cloud.
Timestamp verification to avoid replay attack:
if $\tau_{receiver} - \Delta t > \tau_i > \tau_{receiver} + \Delta t$ **then**
        V-cloud discard it.
else
        Calculates $M_i^{'} = ID_{E_j} \oplus \lambda_i \oplus D_j$ and $\mu_i^{'} = h(\lambda_i||M^{'}||\tau_i)$.
end if
if $\mu_i = \mu_i^{'}$ **then**
        V-cloud saves the information in the encrypted form $Encrypt(VM_i)$ based on the index of ID_{E_j}.
else
        V-cloud discard the information.
end if
End

Secure query searching: Whenever V_i of the domain D_j requires some data stored in the database, firstly, it sends the query request to the corresponding edge E_j of that particular domain D_j. If the information does not exist in the edge database, E_j sends the query to the V-cloud. Thus, the communication overhead at V-cloud can be minimized by searching the data locally. The process of sending the queries, searching in the database using binary searching technique, and getting the response is illustrated in the Algorithm 10. The binary search method is used for searching in the V-cloud database.

4.3 SECURITY PROOF AND ANALYSIS

An adversary $\mathbb{A}$ may try to retrieve the secret parameters or access the entire network illegally, so there is a need to analyze the robustness of the security paradigm of the *SecEdge* scheme. Therefore, security proof is provided using ROM based on the adversary model described in Section 3.1.1. A formal security proof based on the software tool: ProVerif is also provided in this section.

Formal security proof using ROM: In *SecEdge*, there are two main entities, i.e., vehicles, and edge node, and their instances are denoted as $\mathbb{I}_v^{a_1}$, and $\mathbb{I}_{en}^{a_2}$. It is assumed that $\mathbb{A}$ can access and monitor all the public channel communication between the three entities. $\mathbb{A}$ can perform different types of attacks to breach the security of the *SecEdge* scheme. These attacks are simulated using the following Oracle queries:

- $Corrupt(\mathbb{I}_v^{a_1})$: $\mathbb{A}$ illegally access the credentials such as user ID, passwords of the participant $\mathbb{I}_v^{a_1}$.

Algorithm 10 Secure searching and query response

Input: $PVID_i, Query, Pubkey_{V_i}, \tau_i$

Query generation by V_i:
V_i calculates $\rho_i = PVID_i^1 \oplus PVID_i^2 \oplus query \oplus Pubkey_{V_i}$ and $\sigma_i = h(\rho_i || query || \tau_i)$.
V_i sends $\{PVID_i, \rho_i, \sigma_i, \tau_i\}$ to the corresponding edge E_j and V-cloud.
if $T(receive) - \tau_i \geq \Delta t$ **then**
    E_j and V-cloud discard it.
else
    E_j and V-cloud Calculate $Query^{'} = \rho_i \oplus PVID_i^1 \oplus PVID_i^2 \oplus Pubkey_{V_i}$ and $\sigma_i^{'} = h(\rho_i || Query^{'} || \tau_i)$.
end if
if $\sigma_i^{'} = \sigma_i$ **then**
    Perform binary searching.
    Decrypt the information using $Pubkey_{kc}$ and gets the *Result* to the *Query*.
else
    Simply discard it.
end if
Response generation by V-cloud:
After getting the Result, V-cloud calculates $G_i = Result \oplus Pubkey_{V_i} \oplus PVID_i^1 \oplus PVID_i^2$ and $H_i = h(Query^{'} || Result || \tau_j)$.
V-cloud sends $\{G_i, H_i, \tau_j\}$ to the corresponding vehicle V_i.
Timestamp verification to avoid replay attack:
if $\tau_{receiver} - \Delta t > \tau_j > \tau_{receiver} + \Delta t$ **then**
    Vehicle V_i discard it.
else
    Calculates $Result^{'} = G_i \oplus Pubkey_{V_i} \oplus PVID_i^1 \oplus PVID_i^2$ and $H_i^{'} = h(Query^{'} || Result^{'} || \tau_j)$.
end if
if $Query^{'} = Query$ **then**
    The *Result* received by the vehicle is genuine.
else
    False result is obtained. V_i discard it.
end if

- $Send(\mathbb{I}_x^{a_n}, M)$: Message M is sent to $\mathbb{I}_x^{a_n}$ and if M is valid, $\mathbb{A}$ is considered to be a legitimate user.
- $Perform(\mathbb{I}_v^{a_1}, \mathbb{I}_{en}^{a_2})$: $\mathbb{A}$ can access the message transmitted via the public channel between the both of the entities and performs the passive and active attacks.
- $Expose(\mathbb{I}_x^{a_n})$: $\mathbb{A}$ exposes the session key established during the communication between the two entities.
- $Flip(\mathbb{I}_x^{a_n})$: $\mathbb{A}$ tosses an unbiased coin and gets the result c_{toss}. Depending on the results, $\mathbb{A}$ determines the following outcomes: $c_{toss} = 1$, the obtained session key is correct. $c_{toss} = 0$, $\mathbb{A}$ obtains random nonce, and if both results are not obtained, $\mathbb{A}$ gets $NULL$.

Theorem 1: The advantage of $\mathbb{A}$, to break the security of the proposed scheme for computing the private keys in polynomial time T_{poly} is formulated as:

$$\Gamma_{\mathbb{A}}^{SecEdge}(T_{poly}) \leq \left(\frac{Query_h^2}{|h()|} + 2max(Z^{'}.Query_S^{S^{'}}, \frac{Query_S}{2^b})\right), \tag{4.5}$$

Where $Query_h$ and $Query_S$ are the numbers of queries for a hash function, and sending/receiving a message respectively. Here, b denotes the number of bits in password. $Z^{'}$ 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. $|h()|$ represents the range space of hash functions.

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

- **Case 1:** The $\mathbb{A}$ performs an attack on the proposed scheme for guessing w bits of the password in polynomial time, T_{poly}. Here, w bits are selected in starting of the first attempt, hence the advantage for $\mathbb{A}$ can be formulated as:
$$\Gamma_{\mathbb{A}}^{SecEdge}(T_{poly}) = |2.Pr[Win_1] - 1|, \tag{4.6}$$
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 transmitted message M that is in the form of $\{M_i, sign_i = \{\eta_i, \delta_i\}, L_i, Pubkey_{V_i}^{kc}, Pubkey_{V_i}, PVID_i, \tau_i\}$. Here, $\mathbb{A}$ needs to find out the secret parameters like L_i, η_i, and δ_i. These parameters are only known to the authenticated users and thus the probability of winning is the same as the case 1 and it can be shown as:
$$P[Win_2] = P[Win_1]. \tag{4.7}$$

- **Case 3:** $\mathbb{A}$ performs an impersonation attack by pretending itself as an authorized 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:
$$|P[Win_3] - P[Win_2]| \leq \frac{Query_h^2}{|h()|}. \tag{4.8}$$

- **Case 4:** It is assumed that the OBU and Edge are the corrupt devices and $\mathbb{A}$ can access the secret parameters stored in these devices. The probability of guessing the password is reduced to $\frac{1}{2^b}$, where b is the number of bits in the password. Here, $\mathbb{A}$ can also try to guess the password having less entropy using Zipf's law, and those parameters can be used as well. The relation can be formulated as:
$$|P[Win_5] - P[Win_4]| \leq \max(Z^{'}.Query_S^{S^{'}}, \frac{Query_S}{2^b}). \tag{4.9}$$
Finally, it is assumed that $\mathbb{A}$ can also perform random guessing of w bits of the user password, and the probability of success can be written as:
$$P[Win_5] = \frac{1}{2}. \tag{4.10}$$

Using Eq. (4.6), (4.7), and (4.9), the result is obtained as:

$$\frac{1}{2}\Gamma_{\mathbb{A}}^{SecEdge}(T_{poly}) = |P[Win_2] - P[Win_5]|. \tag{4.11}$$

Further, using Eq. (4.8) and (4.9), the result is obtained as:

$$|P[Win_2 - Win_5]| \leq \frac{Query_h^2}{|h()|} + 2max(Z^{'}.Query_S^{S^{'}}, \frac{Query_S}{2^b}). \tag{4.12}$$

Finally, using Eq. (4.11) and (4.12), the desired output of Eq. (4.5) is obtained:

$$\Gamma_{\mathbb{A}}^{SecEdge}(T_{poly}) \leq \frac{Query_h^2}{|h()|} + 2max(Z^{'}.Query_S^{S^{'}}, \frac{Query_S}{2^b}). \tag{4.13}$$

Theorem 2: *SecEdge* scheme incorporates timestamp-based measures and is resilient to replay attacks by ensuring the freshness and uniqueness of received messages M.

Proof: *SecEdge* uses a timestamp in each transmission so that $\mathbb{A}$ can not update the timestamp value.

- **Timestamp Generation:** Whenever the vehicular node transmits the data, it includes a timestamp factor of τ_i.
- **Timestamp Validation:** Upon receiving M, receiver compares the received timestamp τ_i with its current timestamp $\tau_{receiver}$. The acceptable time range for the received message is defined as $[\tau_{receiver} - \Delta t, \tau_{receiver} + \Delta t]$, where Δt is the maximum acceptable time difference.
- **Freshness Check:** If the received timestamp τ_i falls within the acceptable time range $[\tau_{receiver} - \Delta t, \tau_{receiver} + \Delta t]$, receiver entity considers M as fresh and continues processing it.

By employing timestamp-based measures, the scheme ensures that messages are considered valid only if they are received within a defined time frame, providing freshness and uniqueness guarantees. This prevents attackers from replaying intercepted messages successfully. Δt depends on the communication environment. In a delay-sensitive dense environment like IoV, the threshold value is considered to be slightly more than the worst-case average packet transmission time.

Theorem 3: **Correctness of pseudo identity generation:** If the pseudo identity generation process in *SecEdge* is correctly implemented, the generated pseudo identities, $PVID_i^1$ and $PVID_i^2$, are valid and unique for each vehicle V_i.

Proof: To prove the correctness of pseudo identity generation, it is necessary to demonstrate that the generated pseudo identities satisfy the desired properties.

- **Validity:** It is shown that the pseudo identities, $PVID_i^1$ and $PVID_i^2$, meet the specified criteria:
  - $PVID_i^1 = \beta_{ij} \cdot g$: This property ensures that $PVID_i^1$ is a valid pseudo identity calculated by multiplying the randomly selected value β_{ij} with the base point g.
  - $PVID_i{}^2 = w_i \oplus h(u_i||\beta_{ij}||\tau_i)$: This property guarantees that $PVID_i^2$ is a valid pseudo identity derived using the bit-wise XOR operation on w_i and the hash function $h()$ applied to the concatenation of u_i, β_{ij}, and τ_i.
- **Uniqueness:** It is shown that the generated pseudo identities are unique for each vehicle V_i. As β_{ij} is randomly selected for each vehicle, the resulting pseudo identities $PVID_i^1$ and $PVID_i^2$ will be distinct for different vehicles. This uniqueness ensures that each vehicle has a unique pseudo identity within the scheme.

Therefore, based on the correct implementation of the pseudo identity generation process, the generated pseudo identities $PVID_i^1$ and $PVID_i^2$ are both valid and unique for each vehicle V_i.

Theorem 4: **Signature verification and message integrity:** If the signature generation and verification process in the *SecEdge* scheme is correctly implemented, the algorithm guarantees the verification of signatures and ensures the integrity of the exchanged messages.

Proof: To prove the signature verification and message integrity, it is necessary to demonstrate that the algorithm correctly verifies the signatures and guarantees message integrity.

1. **Signature verification:** The algorithm uses the variables η_i and δ_i for signature generation by V_i. By comparing $\eta_i \cdot L_i - \delta_i \cdot Pubkey_{V_i}$ with the calculated value Q_i, the algorithm verifies the signature's validity.

2. **Message integrity:** The algorithm employs cryptographic mechanisms such as hashing and encryption to ensure message integrity. The message integrity is maintained through the use of hash functions, and the XOR operation on $Pubkey_{kc}$ and $Pubkey_{V_i}$.

Therefore, based on the correct implementation of the signature generation and verification process, the *SecEdge* scheme successfully verifies the signatures and guarantees the integrity of the exchanged messages.

Theoretical security analysis: Next, it is discussed in detail how an attack is not feasible against the proposed *SecEdge* scheme. Table 4.4 shows the security comparison of *SecEdge* with existing state-of-the-art methods[134, 100, 135, 136, 137, 101, 72, 116, 205, 206]. Here, the attacks and other keywords are symbolized as follows: S1: Replay attack; S2: Modification; S3: Impersonation attack; S4: Repudiation attack; S5: Man-in-the-middle attack; S6: Location privacy preservation; ✓: Secure; ×: Vulnerable.

Table 4.4: Security analysis for different schemes including the proposed *SecEdge* scheme

Schemes	**S1**	**S2**	**S3**	**S4**	**S5**	**S6**
Zhang et al.[134]	✓	✓	×	✓	×	×
Azees et al.[100]	✓	✓	✓	×	×	✓
Gayathri et al.[135]	✓	✓	×	×	×	✓
He et al.[136]	✓	✓	✓	×	✓	×
M. Cui et al.[137]	✓	✓	✓	×	✓	×
Zhang et al.[101]	✓	✓	✓	×	×	✓
J. Cui et al.[72]	✓	✓	✓	×	✓	×
Bagga et al.[116]	✓	✓	✓	×	×	✓
Zhang et al.[205]	✓	✓	✓	✓	✓	×
Yang et al.[206]	✓	✓	✓	×	✓	×
SecEdge	✓	✓	✓	✓	✓	✓

1. **Modification:** If $\mathbb{A}$, accesses the message information that is passed by any vehicular node through the insecure medium of communication and wants to change the information illegally, then this type of attack can be possible. In the *SecEdge* method, this attack is not feasible due to the following reasons:

   - The set of pseudo identities $PVID_i$ of V_i is generated using a w_i and u_i as follows:

     $$PVID_i^1 = \beta_{ij}.g, PVID_i^2 = w_i \oplus h(u_i||\beta_{ij}||\tau_i),$$

     Where β_{ij} is randomly selected number, $w_i = VID_i \oplus h(Mkey_{RA}.Pubkey_{kc})$ and $u_i = h(VID_i)$. These values are secretly stored in the TPD and not known to the public as well as not easy to derive.

- The method to generate private keys for a particular vehicle also has complicated calculations mentioned in the "system initialization and key generation" phase. These private keys are unknown to the public and can not be easily guessed.

2. **Impersonation attack:** In $SecEdge$, TPD_i has its master key $Mkey_{V_i}$ and $Pubkey_{V_i}$ is its public key. If it is assumed that $\mathbb{A}$ obtains these parameters by any means, then $\mathbb{A}$ should find $PVIDs$, where

$$PVID_i^1 = \beta_{ij} \cdot g, \quad PVID_i^2 = w_i \oplus h(u_i \| \beta_{ij} \| \tau_i),$$

Here, the β_{ij}, w_i, and u_i are unknown parameters, and guessing these values is not easy.

3. **Repudiation attack:** The meaning of repudiation is to deny something. Non-repudiation represents the capability not to deny an activity, such as transferring or getting information. $SecEdge$ uses a vector v in the verification method to achieve non-repudiation. For example, if $\mathbb{A}$ denies the signature by exchanging the message M_i. Signature is generated as $sign_i = \{\eta_i, \delta_i\}$, where, $\eta_i = l_i^{-1} \cdot (Prikey_{V_{i1}}^{kc} + Prikey_{V_i}.h(M_i, Pubkey_{V_i}^{kc}, \tau_i, PVID_i))(\mod q)$ and $\delta_i = (Mkey_{V_i}^{-1}.Prikey_{V_{i1}}^{kc})(\mod q)$.
These altered messages are stopped after the successful transmission using the public key $Pubkey_{V_i}^{kc}$ of V_i. Consequently, once a vehicle generated the signature and signed the message, it cannot deny generating the signature for the delivered message.

4. **Man-in-the-middle attack:** An authenticate V_i broadcasts M_i within the domain of E_j over a public transmission channel and thus $\mathbb{A}$ can access the message along with the public parameters. However, V_i does not simply send the message, i.e., firstly vehicle generates the corresponding signature for the message. For this, it calculates:

$$\eta_i = l_i^{-1} \cdot (Prikey_{V_{i1}}^{kc} + Prikey_{V_i}.h(M_i, Pubkey_{V_i}^{kc}, \tau_i, PVID_i))(\mod q).$$

$$\delta_i = (Mkey_{V_i}^{-1}.Prikey_{V_{i1}}^{kc})(\mod q).$$

Then V_i broadcasts $\{M_i, sign_i = \{\eta_i, \delta_i\}, L_i, Pubkey_{V_i}^{kc}, Pubkey_{V_i}, PVID_i, \tau_i\}$ within the domain of edge E_j. Hence, $\mathbb{A}$ needs to perform some required calculations to access the message that is not easy to calculate intermediate parameters and secret parameters.

5. **Location privacy preservation:** For each communication, E_j calculates a set of pseudo identities for V_i as:

$$PVID_i^1 = \beta_{ij}.g, PVID_i^2 = w_i \oplus h(u_i || \beta_{ij} || \tau_i).$$

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 number β_{ij}. Hence, it is quite challenging for $\mathbb{A}$ to access the original ID of vehicle VID_i. Thus, vehicle location privacy is preserved using $SecEdge$.

6. **Collision Attack Resistance:** SecEdge uses collision-resistant cryptographic hash functions in several stages of the protocol, including pseudo-identity generation and signature creation. For instance, the pseudo-identity $PID2_{Vi}$ is computed as:

$$PID2_{Vi} = w_i \oplus h_2(u_i \| \beta_{ij} \| T_i)$$

where w_i, u_i, and β_{ij} are secret values and T_i is the timestamp. The use of entropy-rich, unique, and time-sensitive parameters ensures that hash inputs remain highly unpredictable, thereby minimizing

the risk of hash collisions even under adversarial modeling. Furthermore, hash-based signature components such as $H_6(M_i, Pubkc_{Vi}, T_i, PID_{Vi})$ also incorporate similar entropy, enforcing strong collision resistance under the random oracle model.

7. **Brute-Force Attack Protection:** SecEdge employs ECC for public-key operations, which offers high security per bit and raises the computational difficulty of brute-force attempts. In addition, all sensitive private parameters (e.g., w_i, u_i) are securely stored in TPDs, preventing their extraction or exposure during brute-force probing or side-channel exploitation.

8. **Biometric Entropy and Password Guessing Defense:** The protocol models the entropy of biometric passwords to resist dictionary-based brute-force attacks. The probability of success for a brute-force attempt is analytically bounded by (Ref.(4.10)):

$$\mathbb{P}[\text{Win}_5] = \frac{1}{2^b}$$

where b denotes the bit-length of the biometric password. Furthermore, the security analysis explicitly incorporates biased guessing distributions using Zipf's Law to ensure robustness even under skewed attacker assumptions.

Formal security proof using ProVerif: Whenever a security scheme is verified by ProVerif, different security properties in terms of queries are fed as input, and the code is then executed. If the query testing is failed, and still there is an attacker, ProVerif starts tracing that attacker. In *SecEdge*, standard attacks are verified using the ProVerif tool. Since, there are three entities in *SecEdge*, namely: V_i, E_j, and RA, the communication pairs are categorized in two forms in the authentication process, namely: (1) Vehicle to RA, and (2) Edge to RA. Considering these communication pairs, there will be the possibility of eight following events:

- Authentication between vehicle and RA
  - `event BeginV(V, RA,` N_V`,` N_{RA}`)`
  - `event EndV(V, RA,` N_V`,` N_{RA}`)`
  - `event BeginRA(RA, V,` N_{RA}`,` N_V`)`
  - `event EndRA(RA, V,` N_{RA}`,` N_V`)`
- Authentication between an edge node and RA
  - `event BeginEdge(E, RA,` N_E`,` N_{RA}`)`
  - `event EndEdge(E, RA,` N_E`,` N_{RA}`)`
  - `event BeginRA_Edge(RA, E,` N_{RA}`,` N_E`)`
  - `event EndRA_Edge(RA, E,` N_{RA}`,` N_E`)`

After defining these events, the respective assertion queries are run on the ProVerif to verify the scheme. Using these assertion queries and the ProVerif tool, it is proven that *SecEdge* successfully achieves confidentiality, authenticity, privacy-preservation, and integrity. The assertion queries for the mutual authentication of the vehicle can be written as:

event(EndV(V, RA, N_V, N_{RA})) $\Longrightarrow$ event(BeginV(V, RA, N_V, N_{RA}))
event(EndRA(RA, V, N_{RA}, N_V)) $\Longrightarrow$ event(BeginRA(RA, V, N_{RA}, N_V))
Note that, the event BeginH1(H1, H2, N1, N2) denotes that the host H1 initiates the authentication process and generated the session with host H2. For that, the secret parameters used are N1 and N2 (nonce). Further, the following assertion queries can be formed for the mutual authentication process between the edge node and RA:
event(EndEdge(E, RA, N_E, N_{RA})) $\Longrightarrow$ event(BeginEdge(E, RA, N_E, N_{RA}))
event(EndRA_Edge(RA, E, N_{RA}, N_E)) $\Longrightarrow$ event(BeginRA_Edge(RA, E, N_{RA}, N_E))
The authentication queries of *SecEdge* are tested using the ProVerif tool, and the results obtained are "True," indicating that the condition created for the authentication is successfully verified by the ProVerif tool. The results for authentication queries are shown in Table 4.5. Similar to the authentication process, the confidentiality of the secret parameters of *SecEdge* is also verified. If the parameter is safe, the result will be true, indicating that the adversary cannot access that secret parameter. In Table 4.5, the results show the successful verification of secrecy or confidentiality of secret parameters used in the authentication process. Along with the authentication and confidentiality check, the *SecEdge* scheme is verified in terms of anonymity or privacy preservation. If the result produced by the ProVerif tool is true, it denotes that the privacy of a particular entity is not violated and is preserved. The privacy of vehicles and edge nodes is tested. In Table 4.5, it is shown that the anonymity of the vehicle and edge node is successfully verified for *SecEdge*.

Table 4.5: Authentication, confidentiality verification, privacy-preservation results using ProVerif

Query (authentication)	**Results**
$event(EndV()) \Longrightarrow event(BeginV())$	True
$event(EndRA()) \Longrightarrow event(BeginRA())$	True
$event(EndEdge()) \Longrightarrow event(BeginEdge())$	True
$event(EndRA_Edge()) \Longrightarrow event(BeginRA_Edge())$	True
Secret parameters (confidentiality)	**Results**
$Mkey_{kc}, Mkey_{RA}, Mkey_E, Mkey_{V_i}, Mkey_{V_i}^{kc}$	True
$Prikey_{V_i}, Prikey_{V_{i1}}^{kc}, Prikey_{V_{i2}}^{kc}$	True
$\{w_i, u_i\}$	True
Identity (privacy-preservation)	**Results**
VID_i	True
$PVID_i$	True
ID_E	True

4.4 PERFORMANCE ANALYSIS AND SIMULATION RESULTS

In this section, the performance of *SecEdge* is evaluated based on some parameters, i.e., computation cost, communication cost, storage cost, and search time complexity. The performance of *SecEdge* is compared with the state-of-the-art methods[134, 100, 135, 136, 137, 101, 72, 116, 205, 206]. The different calculations in the scheme include point multiplication and addition over an elliptic curve, one-way hash function, M2P function, exponential operations, and pairing operation. The obtained values of the average execution time for these cryptographic operations are denoted as T_m, T_a, T_h, T_{M2P}, T_{exp}, and T_p, respectively. The execution time for each cryptographic operation, while executed on Desktop and Raspberry Pi 4 system, is given in Table 3.8. The followings are the detailed discussion of above-mentioned performance parameters:

Computation cost: *SecEdge* is compared with the state-of-the-art schemes, i.e., [134, 100, 135, 136, 137, 101, 72, 116, 205, 206]. Table 4.6 contains the cryptographic equations by calculating the number of each cryptographic operation in the proposed *SecEdge* scheme. Using these cryptographic equations and the execution time for each cryptographic operation provided in Table 3.8, the computation cost is calculated for *SecEdge* and other existing schemes. It can be observed that the computation delay of *SecEdge* has a lower value in both the Desktop and Raspberry Pi 4 environments.

Table 4.6: Computation cost (ms) comparison of *SecEdge* with state-of-the-art schemes

Schemes	Computation cost	Desktop	Raspberry Pi 4
Zhang et al.[134]	$3T_p + T_{M2P} + T_m$	228.61	421.01
Azees et al.[100]	$2T_p + 5T_{exp} + 2T_a$	139.369	278.492
Gayathri et al.[135]	$7T_m + 3T_a + 6T_h$	83.64	127.08
He et al.[136]	$5T_m + 2T_{M2P} + 5T_h$	84.56	115.98
M. Cui et al.[137]	$8T_h + 7Tm + 5T_a$	95.30	147.43
Zhang et al.[101]	$7T_m + 5T_h$	66.44	97.22
J. Cui et al.[72]	$4T_h + 2T_m + 6T_a$	53.45	87.58
Bagga et al.[116]	$10T_h + 9T_m + 2T_a$	97.34	146.10
Zhang et al.[205]	$3T_m + 2T_p$	149.71	281.75
Yang et al.[206]	$6T_m$	56.29	81.50
SecEdge	$5T_m + 3T_h + T_a$	53.05	78.98

Search time complexity: For the complexity comparison, *SecEdge* scheme is compared with the state-of-the-art schemes[139, 140, 207]. Since the existing schemes used to analyze computation cost and energy consumption do not provide any details for time complexity, different relatable schemes are used for this comparison. Whenever V_i wants to know about the identity, speed, or location of another vehicle, it sends a *query* to the corresponding edge E_j. In the worst situation (i.e., required data is unavailable locally at the edge E_j), V_i sends the query to the V-cloud. Based on the demand, V-cloud performs edge-index-based binary searching ($O(logn)$, where, n is the no. of edges in IoV) and gets the relevant edge E_j. Then, V-cloud uses binary search ($O(logm)$, where, m is the no. of vehicles in the domain D_j) to get the required vehicle information. Thus the total time complexity for searching the required information is $(O(logn) + O(logm)$. Table 4.7 shows the comparison of time complexity for different searching techniques.

Table 4.7: Comparison of search time complexity between *SecEdge* and state-of-the-art schemes

Schemes	Search time complexity
Kamra et al.[138]	$O(f_w)$
Cash et al.[139]	$O((r+d_w)/p)$
Kamra et al.[207]	$O(q^2.M)$
SecEdge	$O(logm) + O(logn)$

Energy consumption: Energy consumption can be explained as the amount of energy that a system requires to perform different computations and operations. It is generally calculated in millijoules (mJ). The energy consumption can be determined using the following formula: $Energy = Total_{CC} \times Power_{max}$, Where, $Total_{CC}$ is the total computational cost and $Power_{max}$ is the maximum processing power. For the wireless transmission, the maximum processing power is considered to be 10.88 W. The computational costs given in Table 4.6 for both Desktop and Raspberry Pi systems are used to calculate the consumed energy. The effect of energy consumption on different schemes can be analyzed through the graph representation in Figure 4.2a.

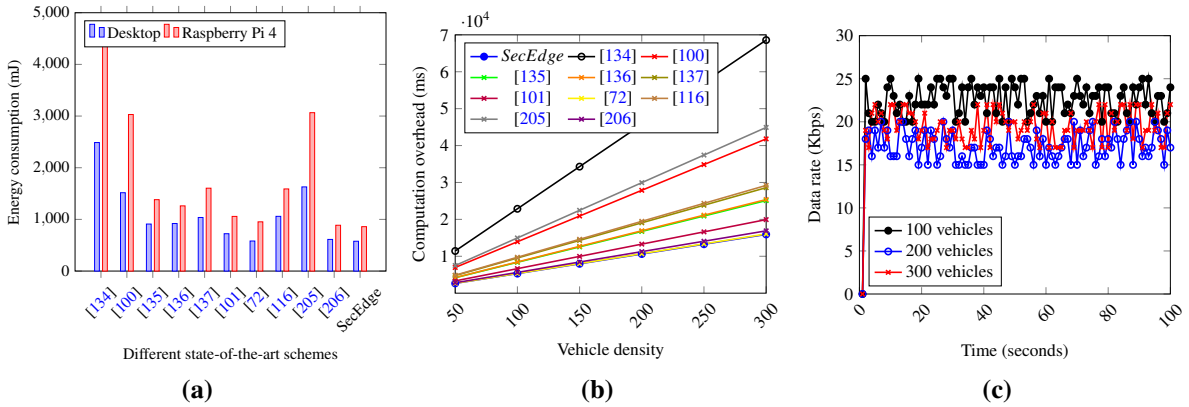

Figure 4.2: Performance evaluation and simulation results of *SecEdge*: (a) Energy consumption comparison of *SecEdge* with state-of-the-art schemes based on execution on Desktop and Raspberry Pi 4 systems, (b) Scalability of *SecEdge* scheme with respect to vehicle density (c) Data-rate of *SecEdge* scheme with respect to the vehicle density

Simulation Environment and Results:

Simulation of *SecEdge* is performed using the SUMO and NS-2 simulator. The .osm file of an urban city in India (Jaipur) is generated from an Open Street Map. With the help of the SUMO simulator, the generated .osm file is converted to an .xml file, and simulation is performed using the NS-2 simulator. The network simulation range is considered to be 1000m $\times$ 1000m and 2500m $\times$ 2500m for highway and city scenarios, respectively. The IEEE 802.11p communication protocol is used for communication. Three scenarios are considered for the simulation: (1) 50 vehicles having a range of 200 m with 3 RSCNs, (2) 100 vehicles having a communication range of 300 m with 7 RSCNs, and (3) 150 vehicles with 10 RSCNs and a communication range of 400 m. Vehicles are allowed different speeds (10 m/s, 15 m/s, and 20 m/s). Five simulations are performed with a duration of 100 seconds each. The communication range for the RSCN is set to 1000 m. For routing, the well-known AODV protocol is used. Scalability and data-rate comparisons are made with respect to the increased number of vehicles. It is claimed that *SecEdge* provides good QoS and low computation overhead in the case of an increased number of vehicles in IoV. Figure 4.2 shows the scalability and data-rate comparison concerning the increased scale of the IoV network. In Figure 4.2b, it is clearly observed that the computation overhead for *SecEdge* is low, even in the case of 300 vehicles. This occurs due to the fewer cryptographic operations, including the one-way hash function used during secure data transmission. This indicates that the performance of *SecEdge* is less affected by the increased number of vehicles. The data rate is considered for three cases: 100 vehicles, 200 vehicles, and 300 vehicles. Figure 4.2c explains that the data rates vary from 15 kbps to 25 kbps, i.e., the data rate is not significantly affected by the increased number of vehicles due to the use of edge nodes. Along with scalability and data rate, it is also claimed that the *SecEdge* scheme is reliable because it avoids the RA crash or any failure and provides a solution for bottleneck issues.

Further, in order to validate the effectiveness of *SecEdge* in terms of enhancing privacy and security, simulations are conducted to evaluate resilience against man-in-the-middle attacks, privacy-preservation, DoS attacks resilience, revocation mechanism, etc. By including experiments related to security performance, *SecEdge* provides empirical evidence to support the claims made about its security capabilities.

Following is a detailed description of these security-related experiments, including the objective, method, and results, as shown in Table 4.8:

Table 4.8: Experiments and results to evaluate security effectiveness

Experiments	**Results**
Man-in-the-middle attack mitigation	Success rate of 95% in detecting and preventing MITM attacks
DoS attack resilience	89% of successful communication sessions maintained despite the attack
Privacy preservation and location tracing prevention	93% prevention rate of location tracing and strong privacy preservation
Security attacks detection and handling	92% accuracy in detecting and responding to security attacks

- **Objective:** Assess the effectiveness of *SecEdge* in mitigating man-in-the-middle attacks.
  Method: A scenario is simulated where $\mathbb{A}$ attempts to intercept and alter communications between the edge and vehicles, and the success rate of detecting and preventing man-in-the-middle attacks with *SecEdge* is measured.

- **Objective:** Evaluation of *SecEdge*'s resilience against DoS attacks that aim to disrupt communication between vehicles.
  Method: A DoS attack scenario is simulated where $\mathbb{A}$ attempts to flood the system with excessive traffic or requests, and the system's ability to maintain normal communication despite the DoS attack is measured.

- **Objective:** Assess *SecEdge*'s effectiveness in preserving privacy and preventing location tracing of vehicles.
  Method: Various scenarios are simulated where $\mathbb{A}$ attempts to track the locations of vehicles or gain access to private information, and the system's ability to protect privacy and prevent location tracing is measured.

- **Objective:** Evaluate *SecEdge*'s ability to detect and handle various security attacks, such as impersonation, replay attacks, and modification of data.
  Method: Different security attack scenarios are simulated, and the detection and response of *SecEdge* to these attacks are measured.

4.5 SUMMARY OF THE WORK

The chapter provides an in-depth analysis of the *SecEdge* scheme, designed to enhance security and efficiency in IoV. In IoV, vast amount of data is generated through various communication channels, necessitating secure data handling solutions. Traditional V-cloud offers scalable storage and computational resources but suffers from bottlenecks and vulnerabilities due to its centralized nature. To mitigate these issues, a hybrid architecture integrating V-cloud with edge computing is proposed, transforming RSCN into robust edge nodes. This decentralized approach enhances data processing and query handling, improving the responsiveness and efficiency of the IoV network. *SecEdge* employs lightweight cryptographic mechanisms,

including one-way hash functions and SSE, to ensure secure data storage and transmission. Formal security proofs using ROM and the ProVerif tool demonstrate SecEdge's resilience against various attacks, as well as its ability to preserve location privacy.

Performance evaluations conducted using SUMO and NS-2 simulators validating the efficiency and scalability of SecEdge. Compared to state-of-the-art schemes, *SecEdge* reduces computation delay by approximately 75% on desktop systems. The scheme also improves search efficiency with a time complexity of $O(\log m) + O(\log n)$ and reduces energy consumption. *SecEdge* handles an increasing number of vehicles with lower computation overhead, maintaining a stable and efficient data rate ranging from 15 kbps to 25 kbps. Empirical results further validate *SecEdge*'s robustness in handling security threats, showing high success rates in mitigating man-in-the-middle attacks, preserving privacy, and preventing DoS attacks. In conclusion, *SecEdge* offers a secure, efficient, and scalable solution for IoV networks, addressing key security and performance challenges and providing a robust framework for the future of IoV.