A Secure and Scalable Framework for Blockchain based Edge Computation Offloading in Social Internet of Vehicles

Uzair Javaid, Student Member, IEEE and Biplab Sikdar, Senior Member, IEEE

Abstract—The integration of Internet of Vehicles (IoV) with social networks has introduced Social IoV (SIoV) that will offer new applications in vehicular networks, e.g., personalized recommendations and route planning. This will be facilitated by heterogeneous access technologies and edge computing to offload tasks from vehicles via secure resource assignment. Thus, each vehicle in SIoV acts as a social subject that manages its own network. SIoV will lead to an explosive growth in network size, and induce issues like scalability and resource discovery. Blockchain is a potential candidate to address these, however, it is not suitable for SIoV with traditional proof-of-work (PoW) consensus. In this paper, we propose a framework that uses a dynamic PoW (dPoW) consensus with a checkpoint mechanism and a resource assignment policy. The dPoW consensus has different mining difficulty levels that change according to the communication traffic, whereas the checkpoint defines an alternative mechanism to generate the next block hash. The assignment policy manages an access control list to mandate the edge modules to securely distribute resources among vehicles. To study the feasibility of our framework, we present a formal security analysis using the Access Control Logic model. For the performance analysis, we use three metrics, i.e., scalability, latency, and security. With these analyses, we demonstrate that our framework offers enhanced security and can scale with a minimal increase in computation overhead. A case study with a comparative analysis is also discussed that evaluates the network dynamics and attests the superior performance of the framework under a real-life vehicular network scenario.

Index Terms—Social Internet of Vehicles (SIoV); scalability; security; blockchain; edge computing; computation offloading

I. INTRODUCTION

The pattern of recent developments in Internet of Things (IoT) has been towards realizing “always-connected” systems and an “all-connected” world [1]. This has further facilitated advancements in Internet of Vehicles (IoV) that introduces overlay networks built atop Vehicular Social Networks (VSN) [2] and Vehicular Ad hoc NETworks (VANETs) [3] to introduce Social IoV (SIoV) [4]. Thus, IoV is a distributed and connected network of vehicles with smart traffic management and roadside infrastructure that manage the communication traffic of vehicles. In contrast, SIoV will enable each vehicle to build and manage its own social network with other vehicles. This is because SIoV induces the fundamental characteristics of social networking within the avenues of vehicular networks.

This number of vehicles is expected to cross two billion within the next 10–20 years [5]. Therefore, SIoV is indispensable for handling this sheer number of vehicles and facilitate the next-generation of intelligent transportation systems (ITS). Thus, a primary objective of SIoV is to improve the safety of vehicles as well as its passengers and the pedestrians alike, by using ubiquitous communication. Moreover, similar to social networks, SIoV will also introduce a wider array of applications in vehicular networks such as localized recommendations and profile based personalized services, vehicle navigation and monitoring, managing network access with edge modules, targeted advertising, and driver behavior modeling [6]. Similar to IoT technologies enhancing IoV, SIoV will be enhanced by distributed edge computing with secure resource assignment.

To integrate social networking in IoTs, edge computing provides a distributed paradigm for offloading tasks from vehicles to edge modules. This partially addresses the resource constraint problem of vehicles by bringing data, storage, and computation closer to them to save bandwidth and drastically improve response times. However, in vehicular networks, it is not easy to guarantee the reliable performance of edge modules due to uneven and insecure resource distribution [7]. Therefore, secure assignment of resources from edge modules to vehicles for computation offloading remains an open issue. Moreover, such an integration within the IoV architecture could also lead to important security problems [8].

Vehicular networks usually rely on decentralized resources, i.e., vehicles communicate with the roadside infrastructure via roadside units (RSUs) that are distributed over a local region. Such distribution limits the centralized models from being suitable for vehicular networks [9]. However, many SIoV applications still rely on centralized architectures. Blockchain could offer promising solutions to decentralize them, which is a globally distributed and online ledger that can be public or private. It provides a transparent and privacy preserving environment with no trusted central authority [10]. Therefore, many applications have adopted it to exploit its different properties, i.e., decentralization for resource sharing, immutability to preserve data integrity, and fault-tolerance to offer resistance against cyber attacks [11]. However, operating a blockchain and its mining processes (e.g., proof-of-work (PoW) based algorithms) while also supporting increasingly complex applications, requires huge computing resources. Therefore, different research avenues have been identified (e.g., real-time processing, resource-intensive applications, consensus mechanisms, and mining) to address the scalability issue of blockchain [11].
A. Motivation and contribution

Conventional PoW based blockchain solutions for SIoV suffer from different challenges such as scalability, poor resource pool, and computational complexity. Scalability represents block throughput rate, i.e., how many transactions a blockchain can process per unit time. Resource pool is the set of resources made available to vehicles by edge modules for task offloading. Computational complexity represents the difficulty level required to mine a block. Thus, blockchain in its traditional structure is not suitable for SIoV as it suffers from poor scalability and high latency.

To mitigate these issues, we propose a two-dimensional blockchain based framework that comprises a dynamic PoW consensus, a checkpoint mechanism, and a resource assignment policy. The blockchain is used for vehicle registrations and to maintain a list of trusted vehicles for access control. The consensus mechanism offers varying mining difficulty levels that enable the blockchain to scale. The checkpoint mechanism defines how to generate a hash of the current block, while the resource policy enforces secure distribution of resources among vehicles via edge modules. Thus, the contributions made by this paper can be summarized as follows:

(i) A two-dimensional blockchain based framework to characterize scalability and security in SIoVs.
(ii) A mechanism to manage vehicle registrations with the public key infrastructure (PKI) features of blockchain.
(iii) A dynamic consensus mechanism that can scale according to the communication traffic rate of vehicles, and a checkpoint based mechanism for hashing a block.
(iv) An access control policy for providing rich computing resources to vehicles via edge modules.
(v) A rigorous evaluations of the framework together with a case study and a comparative analysis.

B. Paper organization

The remainder of the paper is structured as follows. Section II discusses the related work. Section III defines the SIoV-blockchain model while Section IV describes the proposed framework. Sections V and VI present an evaluation. Section VII formulates a case study with a comparative analysis. Finally, the paper is concluded in Section VIII.

II. Literature Review

Traditional vehicular networks rely on centralized servers and therefore, find it difficult to accommodate SIoV applications with increasing complexity. Moreover, the introduction of IoV in social computing presents significant challenges for scalability [12]. Blockchain technology shows promising potential to address these issues such that much of the existing blockchain literature on vehicular networks directly builds atop the existing blockchain paradigms, especially Ethereum (with smart contracts) and Bitcoin blockchains.

Yang et al. [13] formulate a framework for trust management in vehicular networks with blockchain and Bayesian Inference model. They use this combination to generate ratings for messages transmitted by vehicles. Another model to enforce trust among vehicles in VANETs is discussed in [9], where the model enables monitoring of vehicles and their behavior to filter out selfish, rogue, and malicious vehicles. Similarly, a secure vehicle management scheme for VANETs is provided in [14], where credits are assigned to vehicles and securely managed by the scheme. A lightweight directed acyclic graph (DAG) based blockchain for VSNs is evaluated in [15], which promotes a data reduction approach in VSNs to reduce storage costs for vehicular nodes. In a similar approach, the authors of [16] provide an efficient blockchain consensus for VSNs. They claim that their consensus is lightweight and consider a private ledger together with a public one. They discuss communication reliability issues in the private chain and design a scheme to transfer data onto the public chain.

Lu et al. [17], [18] propose a trust management system for vehicular networks using blockchains, which uses the notion of credibility scores based on the historical data of vehicles. Nisha et al. [19] address authentication in vehicular networks and design a framework using blockchain. A trusted crypto point is proposed by Singh et al. [20] that uses blockchain and enables vehicles to securely share data. Similarly, a privacy preserving authentication protocol using Physical Unclonable Functions (PUFs) is also discussed in [21]. Moreover, a message dissemination service for IoVs with blockchain is formulated by Shrestha et al. [22]. A data sharing framework for VSNs is also discussed in [23] that uses a combination of blockchain with secure key-aggregate search encryption scheme. Zhang et al. [24] discuss the significance of blockchain based mobile edge computing in IoV to offset resource consumption. Their solution reduces computational overhead in a blockchain but they have no policy to safeguard resources in edge modules. The authors in [25] use blockchain to present a reward-based communication mechanism for VANETs. They propose Trust Bit that offers unique crypto identities for vehicles that are issued by the vehicle owners.

Furthermore, [26] introduces a data sharing scheme for VANETs with a consortium blockchain. It is worth noting here that consortium blockchains suffer from poor scalability. In [27], the authors discuss a blockchain based software-defined network for making VSNs more secure, which makes a vehicular network programmable, partitionable, and virtualized. A verifiable one-to-many data sharing scheme in VSNs using blockchain is implemented in [28]. This scheme records the access policy of data, offers cloud non-repudiation, and realizes user self-certification. It also takes into account the computing capabilities of a social vehicle and proposes a mechanism for certification. In a similar approach, [29] presents a certification mechanism using blockchain for efficient privacy preserving and location based services in VSNs. For miner computation offloading, PoolCoin is discussed in [30], which offers reputation management for miners in a blockchain. Khelifi et al. [31] report an interesting case of blockchain for secure caching in IoV with named data networking. Lastly, Lei et al. [32] evaluate heterogeneous vehicular systems with dynamic key management through blockchain.

Although the aforementioned frameworks offer robust security and privacy preserving features, they fail to analyze their scalability. Therefore, PoW consensus based blockchain
use a permissionless or permissioned ledger with a consensus protocol that enables interactions between the network constituents, i.e., data collection, data processing, and writing data to storage devices. Similarly, miners here represent the computing power provided by the RSUs.

4) Edge modules: These maintain resource pools of the network and govern how resources are assigned to vehicles. For instance, when a vehicle requests to use an application-specific resource (e.g., data processing) to offload its task, the edge module checks if the vehicle is registered and then, assigns it with a specific resource limit. Moreover, the edge modules are responsible for preventing the network from overflowing, thereby acting as ‘reference monitors’. Note that when assigning resources, we refer to edge modules as objects.

5) Blockchain: This represents a regional blockchain ledger. The use of such a ledger is considered here as it is widely established that blockchain solutions have latency and block propagation issues. Therefore, we consider a local and permissioned blockchain instance hosted by a geographically bounded vehicular network for simplification.

III. THE SIoV-BLOCKCHAIN MODEL

Figure 1 depicts a holistic overview of the proposed network model. The road lanes are represented in yellow with vehicles that are securely communicating with the infrastructure via RSUs, and requesting resources from the edge modules. We now explain the core components of our framework.

1) Vehicles: These represent the key users of the SIoV-blockchain model that communicate with the blockchain via RSUs and request resources from the edge modules for offloading application-specific computing tasks. Note that when requesting resources, we refer to vehicles as subjects. Moreover, every vehicle also has a blockchain account and a public-private key pair for encrypted communication.

2) RSUs: These are part of the roadside infrastructure that provides communication services to vehicles. RSUs operate on a 5.9GHz DSRC band that provides low latency and is compatible with vehicular networks. This is required for high speed events and nodes with high mobility. We consider RSUs as miners in this paper, that also manage vehicle registrations.

3) Server/miner: These are systems that can interact with vehicles and the roadside infrastructure to provide different services, e.g., deploying a blockchain. A server is, therefore, inherently trusted because it instantiates a blockchain. It may

TABLE I: A summary of existing literature

<table>
<thead>
<tr>
<th>Feature</th>
<th>Proposed</th>
<th>[9], [14], [27]</th>
<th>[15], [16], [21], [24], [22], [20], [25], [18], [30], [28], [21], [26], [31], [19], [13], [29], [17], [20], [25], [18], [32], [23]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scalability</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Security</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Distinctive ID</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Privacy</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Registration</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Resource handling</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Latency</td>
<td>✔</td>
<td>✔</td>
<td>✔</td>
</tr>
</tbody>
</table>

- scalability: does it discuss how it scale with traffic?
- security: does it argue about its security features?
- distinctive ID: does it assign unique IDs to vehicles?
- privacy: does it preserve the privacy of vehicles?
- registration: does it provide a mechanism for vehicle registration?
- resource handling: does it provide resource management?
- latency: does it argue how it affects message latency?

IV. A BLOCKCHAIN FRAMEWORK WITH CHECKPOINTS

Similar to a sliding window algorithm, the checkpoint based blockchain framework operates by sliding through the ledger to mine new blocks. When a blockchain is initialized, the checkpoint size is the same as of the genesis block, i.e., one. However, after adding new blocks, size of the checkpoint increases to \( n \) blocks. Note that these blocks are referred to as ‘checkpoint blocks’ henceforth. Thus, when a new block is mined, the blocks included in the checkpoint are hashed together with the current block, i.e., a miner needs to hash the current block together with the checkpoint blocks.

The checkpoint framework has a computational complexity of \( O(n) \) for a constant mining difficulty, where \( n \) represents the checkpoint size. As data is omnipresent in SIoV networks, it is of paramount importance that its integrity be preserved. This is addressed with checkpoint blocks that enhance the immutability of transactions in a ledger. To mine a new block with checkpoints, an adversary or a dishonest miner requires knowledge of the checkpoint size \( n \), as compared with a checkpoint free blockchain where no such parameter is involved. In
this paper, we assume size of the checkpoint is kept secret such that only honest miners (RSUs) in the network have its knowledge. Moreover, vehicles in the proposed framework are the primary users that have their own blockchain accounts [33] and a 20-byte address (i.e., their ID) similar to the address sizes of Bitcoin [34] and Ethereum [35], [36]. However, before interacting with the infrastructure, the vehicles first need to register themselves. We assume that RSUs host the blockchain network and are therefore, responsible for the registration of vehicles. When a new vehicle wants to register, RSUs assign it a public-private key pair and add it to a list of trusted devices, which we term as an Access Control List (ACL). Note that the ACL is shared with the edge modules for secure distribution of resources.

### Listing 1: Structure of a block and its contents

```json
{  "block":  {    "index": "number of the block in blockchain",    "target": "difficulty with which the block was mined",    "nonce": "value to produce the required target hash",    "current_hash": "hash of the current block",    "previous_hash": "hash of the previous block",    "timestamp": "creation time (seconds from Unix Epoch)"  }}
```

### A. Dynamic proof-of-work

The checkpoint mechanism incurs additional mining time cost, therefore, a dynamic PoW (dPoW) consensus is proposed that uses four different difficulty levels. Note that each difficulty is set for a certain rate of incoming communication traffic from the vehicles. Thus, we categorize the difficulty levels according to arrival rates and define them as follows:

- **Low**: It has the highest difficulty and corresponds to the lowest traffic arrival rates. The difficulty is defined by the first four significant zeros of the target hash string. Note that each zero represents a nibble (4 bits). Thus, the total difficulty of this level is 16 bits (four nibbles).

- **Medium low**: This level corresponds to traffic arrival rates higher than Low level and has higher difficulty. The difficulty here is defined by the first three significant zeros of the required target hash. Thus, the total difficulty of this level is 12 bits (three nibbles).

- **Medium high**: This level corresponds to traffic arrival rates higher than Medium Low level and has low difficulty defined by the first two significant zeros. Thus, the total difficulty decreases to 8 bits (two nibbles).

- **High**: It has the lowest difficulty of all levels and corresponds to the highest arrival rates. The difficulty is defined by the first significant zero only. Thus, the total difficulty here is 4 bits (one nibble).

Thus, similar to PoW, to mine a new block with checkpoints regardless of the difficulty level, a miner needs to concatenate the checkpoint blocks with the current block and then, start hashing them together with an initial nonce value of 0. A nonce represents the value (or in other words, proof) of a block that produces the required target hash string. Thus, on each hash iteration, a miner increments the value of nonce by one until the target value is produced.

### B. Parameters of proposed framework

The size of a block in Bitcoin is limited to 1 megabytes (MB) [37] such that every 10 minutes, a new block is mined [38], [34]. In this paper, we consider a variable block size with an upper bound of 1MB. This is because of the dPoW consensus with different difficulty levels. Thus, if a block requires more size than the upper bound, its data is partitioned to fit in more blocks via sampling. Moreover, the block structure of our framework is defined in Listing 1, which shows the content that is stored in a block. The block size in the proposed framework can be calculated as follows:

\[
\text{block size} = \text{block overhead} + [(\text{data + encryption overhead}) \times \text{samples}],
\]

where ‘encryption overhead’ is the added cost of encryption (in number of bytes) for the data encrypted in the blockchain, ‘samples’ denote the number of data samples required for data fitting, and ‘block overhead’ is the number of bytes used to represent a block header. Note that the proposed framework uses Elliptic Curve Cryptography (ECC) together with Elliptic Curve Digital Signature Algorithm (ECDSA) for encrypting data and recall that block size is limited to 1MB only. Thus, we can now define the difficulty in computing a target hash value in the following way:

\[
\text{difficulty} = \frac{(\text{difficulty level}^{dPoW})}{\text{target hash}},
\]

where ‘difficulty level’\(^{dPoW}\) represents the four levels discussed above, while ‘target hash’ is the value required for mining a block. For instance, the target hash defined by Low level can be represented in hexadecimal base as: 0000 aaaabcdccccdddeeefff 0000 1111 2222 3333 4444 5555 6666 7777 8888.

We now calculate the block interval time, which is the average time taken to mine a new block, as follows:

\[
\text{block interval} = \frac{\text{difficulty} \times 2^{(2^\text{hashrate})}}{\text{hashrate}}
\]

where ‘2\(^{\text{hashrate}}\)’ signifies the difficulty level, i.e., it is 16 bits for Low level, 4 bits for High level and vice versa, whereas ‘hashrate’ represents the speed with which a miner computes a hash, i.e., the number of hashes generated per second. Furthermore, the mining rewards are distributed to the miners on successfully mining a block as follows:

\[
\text{block reward} = \frac{\text{hashrate} \times \text{cost/hash}}{\text{token price}}
\]

where ‘cost/hash’ is the computation cost required per hash, ‘token price’ is the price of the token with which the framework can be designed (e.g., it is Ether for Ethereum, Bitcoin for Bitcoin blockchain, etc.), and ‘block reward’ is the amount of tokens awarded to a miner. Note that the ‘block reward’ is at the discretion of the designer and can be chosen as required by different applications.
C. Edge computation offloading

In typical SIoV applications, an edge module may have a resource (service, data etc.) that is needed by vehicles for offloading their computing tasks. However, to prevent unauthorized access, resources must be safeguarded and their wastage mitigated via an access control policy. In this paper, we formulate this by adopting the security model of an access control matrix (ACMx) [39], [40], where ACMx is only an abstract representation of an access control problem. Moreover, as RSUs register vehicles and store their IDs in the ACL, we define the ACMx for resource assignment with ACL where each entry specifies a subject, an object, and a resource. Thus, we derive the following definitions to formulate resource assignment in the proposed framework.

- A set of subjects \( S \) that includes vehicles \( v_s \) which need to access a specific resource \( r_i \), where \( v_s \in S \).
- A set of objects \( O \) that includes edge modules \( e_o \) which host a set of resources \( R_O \), where \( e_o \in O \).
- Each object device \( e_o \in O \) has a set \( \rho_o \) of available resources, where \( \rho_o \subseteq R_O \).
- Each resource \( r_i \in R_O \) has a defined static limit and is associated with a set of access control rights \( A_{ri} \).
- For each subject device \( v_s \in S \) and resource \( r_i \), a mapping \( g : r_i \rightarrow v_s \), \( g \in (v_s,r_i) \subseteq A_{ri} \) is defined to specify access rights on \( r_i \) granted to \( v_s \) by \( e_o \).

Here, \( r_i \) represents information, a file, or other application relevant data, while the set of access control rights \( A_{ri} \) uses propositional logic with access control operators to relate principals with the statements they make. To formally define our framework, we adopt the “access control logic” model [42] for reasoning about the operation of the proposed framework and its constituents. This model uses propositional logic with access control operators to relate principals with the statements they make.

A. Formal security analysis

To formally define our framework, we adopt the “access control logic” model [42] for reasoning about the operation of the proposed framework and its constituents. This model uses propositional logic with access control operators to relate principals with the statements they make. The formal analysis of our framework is based on developing a series of inference rules for rigorous reasoning of the access control statements considered in this paper, including access requests, authority, and jurisdiction. Thus, using this model, we first define the following assumptions for edge modules (reference monitors):

1. Completeness: Edge modules cannot be bypassed, i.e., a resource cannot be accessed without them.
2. Isolation: Edge modules are tamper proof, i.e., the access control policy details cannot be changed by an adversary.
3. Verifiable: Edge modules are correctly implemented, i.e., their functions have been specified precisely.

We now define the resource assignment on a subject as a “ticket- mechanism” and the trusted list of subjects as a “ACL-mechanism” via access control statements and access control operators.

Definition 2. Access control statements: These statements determine with precision and accuracy which access requests from which subjects should be granted, and which should be denied. We formulate these statements by using propositional variables, i.e., access control operators, to express our assumptions and expectations, such as which authorities we trust and which subjects should be granted access to which objects.

Definition 3. Access control operators: These operators are used to express the access control statements. We use three such operators (i.e., “says”, “controls”, and “speaks for”) with the following semantics: Says: a formula \( v_s \ says \ \Phi \) that is meant to denote a situation in which subject \( v_s \) makes the statement \( \Phi \); Controls: a formula \( v_s \ controls \ \Phi \) that expresses a subject \( v_s \’s \) jurisdiction or authority regarding the statement \( \Phi \); Speaks for: a formula \( v_{s1} \Longrightarrow v_{s2} \) that represents a
proxy relationship between $v_{s1}$ and $v_{s2}$ that allows us to safely attribute $v_{s1}$’s statements to $v_{s2}$ as well.

**Definition 4. Ticket-oriented mechanism:** A ticket represents the capability granted to a subject, i.e., a constant resource assignment which it can use to perform its tasks. In other words, it is the authority of a subject that allows it to use the resources which are assigned to it. The resource control mechanism proposed in this paper can be written as a ticket-oriented access control mechanism consisting of the following four components, which we express as logic statements:

**Access policy:** This statement asserts that edge modules have jurisdiction over which subjects (say $v_s$) can exercise an access right $a_r$ on a resource $r$.

\[
\text{EDGE controls } (v_s \text{ controls } (a_r, r))
\]

**Ticket:** This statement asserts that a ticket is a credential stating that a subject has the right to access a resource.

\[
\text{TICKET says } (v_s \text{ controls } (a_r, r))
\]

**Trust:** Tickets represent unforgeable tokens of authority issued by edge modules, i.e., the static resource assignment.

\[
\text{TICKET } \Rightarrow \text{ authority}
\]

**Offloading request:** This occurs when a subject requests a resource and presents its ticket, i.e., the resource it can consume.

\[
v_s \text{ says } (a_r, r)
\]

These components provide sufficient information to contracts to determine if a requested action $(a_r, r)$ by $d_s$ should be permitted or not. A ticket-oriented access control policy for our scheme can then be expressed as the following inference rule:

\[
\text{T-RULE } \quad v_s \text{ says } (a_r, r) \quad \text{EDGE controls } (v_s \text{ controls } (a_r, r))
\]

\[
\text{TICKET } \Rightarrow \text{ authority } \quad \text{TICKET says } (v_s \text{ controls } (a_r, r))
\]

\[
\text{TICKET } \Rightarrow \text{ authority }\]

\[
\langle a_r, r \rangle
\]

**Theorem 1.** The edge computation offloading with static resource limit assignment successfully preserves the resources of the proposed framework.

\[
\text{ACL says } (v_s \text{ controls } (a_r, r))
\]

Thus, for any $i$, we can extract the following proposition:

\[
\text{ACL says } (v_{s1} \text{ controls } (a_{r1}, r_1)) \land \text{ACL says } (v_{s2} \text{ controls } (a_{r2}, r_2)) \land \ldots \land \text{ACL says } (v_{sn} \text{ controls } (a_{rn}, r_n)),
\]

**Proof.** Based on the adversary model, an adversary $A$ may compromise a subject to send communication traffic to an object and/or other subjects. If $A$ can successfully consume all available resources, we then say that $A$ compromises the whole system.

Assume that $A$ compromises a subject $v_{s1}$ and gains unauthorized access to its access rights, i.e., $(a_{r1}, r_1)$. $A$ may now send a request to an object $e_{o1}$ to exercise its right on resource
r_1 and consume it. Thus, \( A \) (using compromised \( v_{s1} \)) is able to successfully consume \( r_1 \) which can be formally written as:

i. \( v_{s1} \) says \( r_1 \): access request generated by \( A \).

ii. EDGE controls \((v_{s1} \text{ controls } r_1)\): access policy.

iii. TICKET \( \Rightarrow \) authority: trust policy.

iv. TICKET says \((v_{s1} \text{ controls } r_1)\) by Definition 4.

v. EDGE says \((v_{s1} \text{ controls } r_1)\): (iii), (iv) speaks for.

vi. \( v_{s1} \) controls \( r_1 \): (ii), (v) controls.

vii. \( r_1 \): (vi), (i) controls.

Now, \( A \) may try to consume more resources than what is allocated to \( v_{s1} \) in which case his/her atomic proposition translates to \( f_{a1}, (r_1, r_2) \). \( A \) sends the request again to \( e_{o1} \) but is rejected by Definition 5, where \((v_{s1} \text{ controls } r_1) \land (v_{s1} \text{ controls } r_2) \neq v_{s1} \text{ controls } (r_1 \land r_2)\). This limits the impact of \( A \) to only one subject and its resources only, thereby preserving the resources of the system. Extending this argument, if \( A \) tries to consume more resources than \( r_1 \) such that \((v_{s1} \text{ controls } (r_1 \land r_2 \land \cdots \land r_n)) \), he/she will be rejected again by Definition 5. Furthermore, even if \( A \) tries to launch a sinkhole attack on another subject \( v_{s2} \) by compromising many devices, the impact of the attack will still be limited to one subject device by Definitions 1 and 4.

**Theorem 2.** The proposed framework successfully protects the system from unauthorized subject requests.

**Proof.** Consider a scenario where an adversary \( A \) selects a rogue subject \( v_{sr} \) and uses it to access a resource for which it does not have access rights. If \( A \) can successfully identify himself/herself as the rightful owner of this resource, then we say that \( A \) successfully compromises the system.

Assume that \( A \) tries to access a resource \( r_o \) by sending a message to an object \( e_{o1} \). \( A \) generates its fake credentials for \( (a_{ro}, r_o) \) to exercise the attack. For a successful execution, \( A \) has to bypass the edge modules. By access logic assumptions, they cannot be bypassed. Therefore, \( A \)'s request is denied by the edge modules. This is because these modules maintain the ACL and by Definition 5, ACL has the authority to authorize subjects. Thus, the attack by \( A \) is formally nullified in the following way:

i. \( A \) says \((a_{ro}, r_o)\): access request generated by \( A \).

ii. EDGE controls \((v_{sr} \text{ controls } (a_{ro}, r_o))\): access policy.

iii. ACL \( \Rightarrow \) authority: trust policy.

iv. ACL says \( \cdots \land \cdots \land \cdots \): by Definition 5.

v. ACL says \((v_{so} \text{ controls } r_o)\).

vi. \( A \)'s request is dropped and the attack is invalidated.

VI. Performance Analysis

This section presents an evaluation of the proposed framework with the following performance indicators:

A. Scalability

As a blockchain is essentially a chronological chain or list of blocks, each block in it contains both data (tuples of transactions) and meta-data (previous block hash). Note that in our framework, it is previous block together with the checkpoint blocks. As this chain of blocks constantly grows in length over time, we evaluate scalability in these terms: communication overhead, dynamic proof-of-work consensus, and mining with checkpoints. We also compare our framework with the state-of-the-art one proposed by Yang et al. [13].

1) Communication overhead: To study the effect of overhead generated by the communication traffic of vehicles, we show how an increase in the number of vehicles contributes to the communication overhead. To measure this, we factored in the number of vehicles with time and the size of their transmitted packets. As the blockchain packet that is considered in this paper has a 256-byte length (or 0.25MB), we can see in Figure 2 that the communication overhead of our framework increases in a linear fashion with increase in the number of vehicles. Note that the y-axis represents the overhead in megabytes. We can also see that the overhead induced by [13] is almost three times greater, which attests that our framework operates with three times lower overhead cost.

![Fig. 2: Communication overhead in the proposed framework.](image)

**TABLE II: Different difficulty levels of dPoW consensus**

<table>
<thead>
<tr>
<th>( \delta )</th>
<th>Arrival rate</th>
<th>Difficulty level</th>
<th>PoW target</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>low</td>
<td>high++</td>
<td>SHA-256[0:4]</td>
</tr>
<tr>
<td>2</td>
<td>medium-low</td>
<td>high</td>
<td>SHA-256[0:3]</td>
</tr>
<tr>
<td>3</td>
<td>medium-high</td>
<td>medium</td>
<td>SHA-256[0:2]</td>
</tr>
<tr>
<td>4</td>
<td>high</td>
<td>low</td>
<td>SHA-256[0:1]</td>
</tr>
</tbody>
</table>

2) Dynamic proof-of-work: We first explain and discuss the dPoW consensus without the checkpoint mechanism. This is to demonstrate the consensus operation and to facilitate better understanding of checkpoints. For dPoW, we considered four different difficulty levels as listed in Table II. An illustration of how much time different difficulty levels take to mine a block is also presented in Figure 3. From the table, it can be observed that mining difficulty is the highest with low arrival rate, which has a target hash value with four significant zeroes. Note that the hashing algorithm used in the proposed framework
is SHA\textsuperscript{−}256. Moreover, mining difficulty is the lowest with high arrival rate, which has a target hash value with one significant zero. This dynamic property of changing difficulty levels allows the framework to scale effectively such that high traffic rate associated with low mining difficulty enables a high block throughput, while low traffic rate associated with high mining difficulty results in enhanced security fidelity and robust fault tolerance. It is worth noting here that a trade-off between security and scalability is induced by using dPoW, i.e., low difficulty levels ensure high throughput while high difficulty levels guarantee high security. For low levels, the raw throughput is achieved by losing a certain degree of security, i.e., at low difficulty levels, it is easier to solve the PoW puzzle that takes less time and lower number of hash attempts to mine a block. Therefore, low difficulty levels are more vulnerable than the high levels, and as a result, we use a combination of both to make an acceptable trade-off between the overall security and scalability features of our framework. Thus, for peak communication rates, the framework can scale, while for off-peak rates, the framework prioritizes security.

For our analysis, we simulated the difficulty levels listed in Table II for a total of 1,000 blocks such that each block contained 10 dummy transactions. A summary of the results is compiled in Figure 4. Thus, it can be seen that the total time taken to mine the blocks requires the following time costs:

i. For $\delta = 1$, the total time consumed is around 10,000 seconds (i.e., a throughput of only 1 transaction per second \(tx/s\)), as shown in Figure 4(a).

ii. For $\delta = 2$, it is approximately 660 seconds with a throughput of 15\(tx/s\), as shown in Figure 4(c).

iii. For $\delta = 3$, it is almost 45 seconds with a throughput of roughly 220\(tx/s\), as shown in Figure 4(e).

iv. For $\delta = 4$, it is 4.5 seconds with a throughput of roughly 2,200\(tx/s\), as shown in Figure 4(g).

Thus, the throughput is lower for low traffic rates while for peak rates, it is accordingly higher. This demonstrates scalability in the dPoW consensus.

3) Mining with checkpoints: The PoW target hash in our framework is a 256-bit number (64 hexadecimal digits), where one hex value represents four bits, i.e., a nibble. For instance, \texttt{0000 abc0 abcd abcd abcd abcd abcd abcd efef efef efef efef efef efef efef} 12ff is a hexadecimal string and the four zeros at the start represent a 16-bit difficulty level. Thus, we can deduce that the lower a target value be, the more difficult will be mining a block. Note that the required target value to mine a block can alternatively be represented as difficulty, where a higher difficulty translates to a lower target value. This corresponds with our discussion that low arrival rates have high difficulty, whereas high arrival rates have low difficulty. Moreover, the iteration at which the target hash value is produced, is called the nonce. Therefore, to solve the PoW puzzle and mine a block, a miner starts with a nonce value of zero and iterates it incrementally until the target hash is produced. As we have demonstrated how dPoW functions, we now introduce the notion of checkpoints in it.

The time taken to solve the PoW puzzle with varying difficulty levels and checkpoint sizes is shown in Figure 5. We consider the same difficulty levels as of Table II and five different checkpoint sizes (i.e., 10, 20, 30, 40, 50). Recall that a checkpoint of size 50 means that a miner needs to hash the current block with 49 preceding blocks and then, find the nonce value that produces the target hash. For checkpoint enabled dPoW, we considered 100 blocks with 100 dummy transactions in each. Thus, we now report the median values of mining time under different difficulty levels as follows:

i. For $\delta = 1$ with high++ difficulty, the average time cost is around 900 seconds, as shown in Figure 5(a).

ii. For $\delta = 2$ with high difficulty, it is almost 75 seconds, as shown in Figure 5(b).

iii. For $\delta = 3$ with medium difficulty, it is approximately 20 seconds, as shown in Figure 5(c).

iv. For $\delta = 4$ with low difficulty, it is 0.3 seconds, as shown in Figure 5(d).

An interesting observation here is that the different mining times in Figure 5 do not follow a linear trend, i.e., they do not linearly increase with increase in checkpoint sizes; they are consistent for some checkpoint sizes while inconsistent for others. This is because mining depends on a number of factors, i.e., difficulty level, time cost to produce a target hash, checkpoint length, and block data size. Note that for a particular difficulty level in our simulation, we assumed constant size of block data, whereas the checkpoint size and mining time were varied for each block. Moreover, the nature of the dPoW puzzle requires producing a target hash string such that with low difficulty, the number of solutions is bigger, whereas for high difficulty, the number of solutions is smaller. Therefore, the probability of guessing is higher for low difficulty levels and lower for high difficulty levels, respectively. Thus, as we increase the difficulty, the number of combinations that fulfill the target hash requirements decreases in tandem. Additionally, as dPoW comprises a totally random process of generating hashes, its behavior can be construed as unpredictable, i.e., a miner may be able to produce a target hash with 100 guesses while another miner may generate it with 1,000 guesses. Due to these reasons, the trend in Figure 5 is non-linear and occasionally inconsistent.

Let us assume that the time to reach a target $t_r$ with a hash length of $n$ bits can be expressed for the worst-case as: $t_r = O(2^n)$. Therefore, the time spent by a miner $t_m$ to produce the

![Fig. 3: A depiction of difficulty levels.](image-url)
target hash value can be expressed as $t_m = O(m \times t_r)$, where $t_r \gg m$. This way, we can conclude that such an uncertain mining time is one of the reasons behind the non-linearity for difficulty levels low, medium, and high++ in Figure 5.

1) **Authentication delay at RSUs**: Figure 6 shows the delay incurred by the proposed framework and [13] for vehicles to authorize themselves with the RSUs, i.e., the time taken by an RSU to authorize a vehicle. The simulation results were obtained in Python using a custom discrete-event simulator, where we measured the delay by taking the communication cost of vehicles and sampling it over their data transmission rate (2.048Kbps in this case). We can observe that the delay increases in a linear fashion with increase in number of vehicles. Moreover, the delay incurred by [13] is approximately four times greater than the proposed one, which confirms that our framework has lower turnaround time for authentication.

2) **Time-to-Finality**: To measure TiF for dPoW, we consider the difficulty levels listed in Table II. Note that TiF for checkpoint enabled dPoW is already discussed in Section VI-A3. As we know that transaction processing in a blockchain is a two-fold process that includes gathering transactions to form a block and mine it, and subsequently, reach a consensus with other miners on the mined block. Therefore, we formulate TiF for transactions generated by vehicles, which includes block mining time (or block interval) and block verification time as following: $t_{f,\delta} = \{t_{i,\delta} + t_{c,\delta} : \delta = 1, 2, 3, 4\}$, where $t_{i,\delta}$ is the block interval time while $t_{c,\delta}$ is the consensus latency that is the time taken by a miner to verify a block. Note that $t_{c,\delta}$ is dependent on the choice of difficulty levels $\delta$. Moreover, for simplification and without loss of generality, we assume that $t_{c,\delta} = 1$. This is because verifying a block is an easy task and only involves inputting the nonce value

---

**Fig. 4**: Comparison of mining dynamics at RSUs with dPoW consensus for different traffic arrival rates generated by vehicles.

**Fig. 5**: Mining 100 blocks with checkpoint enabled dPoW consensus with different difficulty levels and checkpoint sizes.

**Fig. 6**: Authentication delay at RSUs.

---

**B. Latency**

Latency is a critical performance metric for SiOVs because it tells how much time a vehicle requires to successfully authenticate itself and thereafter, its request with the infrastructure. Therefore, we evaluate the latency of our framework in terms of authentication delay at RSUs and Time-to-Finality (TiF).
that generates the target hash. Therefore, we consider $t_{i,5}$ to study the mining dynamics of the proposed framework with different communication traffic arrival rates.

For our evaluation, we simulated the difficulty levels listed in Table II for a total of 1,000 blocks such that each block contained 100 dummy transactions, which are shown in Figure 4. Thus, we observe that the hash iterations committed to mine all the blocks require the following number:

i. For $\delta = 1$, the hash iterations required for each block (i.e., TiF for its set of 100 transactions) has a peak value of 400,000 hash iterations per block $(h/b)$, as can be seen in Figure 4(b).

ii. For $\delta = 2$, it is 35000$h/b$, as can be seen in Figure 4(d).

iii. For $\delta = 3$, the peak value is 2500$h/b$, as can be seen in Figure 4(f).

iv. For $\delta = 4$, it is approximately 120$h/b$, as can be seen in Figure 4(h).

Thus, we can deduce from these observations that TiF is high for low arrival rates, whereas it is low for high arrival rates. This translates into low latency operation of the proposed framework.

### Algorithm 1: Block concatenation

**Function:** append  
**Input:** block\_digest  
**Output:** target\_hash\_string

```plaintext
1 set target : $T = 0|0|0|00|000'$
2 set nonce : $N_{curr} = 0$
3 while mine do
4 for block\_index $\in [0, 100]$ do ▷ current block
5 $B_{curr} = block(\cdot)$ ▷ current block
6 $B_{chk} = \sum_{i=1}^{n-1} B_{t_i}$ ▷ checkpoint blocks
7 $T_{curr} = string(tx).encode(\cdot)$ ▷ transactions set
8 $H_{curr} = h(B_{chk}, B_{curr}, T_{curr}, N_{curr})$
9 if $H_{curr} == SHA-256[0 : 4] \leq T$ then
10 | break
11 else
12 | $N_{curr} ++$
```

### C. Security

We formulate the degree of security offered by the proposed framework using the following steps:

i. Let us assume that a security password $P$ is input to a classical symmetric encryption algorithm $E$ such that at a stage $S_1$, it randomly generates a key $K_P$.

ii. $E$ now uses $K_p$ and vehicle data $D$ to generate a packet of encrypted data $\{D\}_K_P$, where $K_P$ is stored in the blockchain and each $K_P$ uniquely corresponds to a vehicle as its pseudonym ID.

iii. The block concatenation process, as shown in Algorithm 1, takes the parameters of the current block as well as of the checkpoint blocks $n-1$ as an input, where $n$ denotes the checkpoint size. It can be observed that this algorithm will fail if the current block to be mined already exists in the blockchain. Otherwise, the miners in the network will begin to compete and solve the PoW puzzle for the current block. This way, the algorithm will succeed if the puzzle is solved and is verified by the other miners in the network.

iv. Let us represent the $i^{th}$ block in the blockchain ledger by $B_i$, current block to be mined by $B_{curr}$, the set of transactions in it by $T_{curr}$, and its nonce value by $N_{curr}$. Using these notations, we can now formulate the hash of the current block $H_{curr}$ in the following way:

$v_i = h(\sum_{i=1}^{n-1} B_{t_i} + [B_{curr} + T_{curr} + N_{curr}])$, where $\sum_{i=1}^{n-1} B_{t_i}$ represents the previous blocks, i.e., $n-1$ checkpoint blocks and $l$ is the length of the blockchain inclusive of the current block.

#### 1) Computational complexity:
When we have a checkpoint of size $m$ such that its value is shared with an RSU (miner) beforehand, the computational complexity in solving the PoW puzzle can be expressed as: $t_m = O(m \times t_r)$, where $t_r$ is the time required to produce the target hash string of the current block in a search space of size $2^n$. Note that $n$ represents the length of the output hash in bits. Therefore, we can write $t_m$ as: $t_m = O(m \times 2^n)$. Given our assumption that an adversary $A$ has no knowledge about the size of the checkpoint, $A$ needs to compute the hashes for all possible checkpoint sizes to launch an attack. Therefore, the computational complexity to launch a successful attack increases to: $t_m = O(m \times (m \cdot t_r)) = O(m^2 \times 2^n)$.

#### 2) Ledger guarantee:
The proposed framework uses a modified version of the PoW consensus algorithm, i.e., checkpoint enabled dPoW. It can be argued that PoW, albeit slow, offers a very high degree of security by hashing and chronologically linking all the blocks with the genesis block in a blockchain. However, this is only true if the number of honest miners that share a computing resource pool, is greater than the dishonest ones. Therefore, to guarantee the security of the ledger with consensus algorithm $\delta$, the number of malicious miners $n_m$ (out of a total of $n_h$ miners) needs to be strictly restricted by the following constraint: $n_m \leq v_3$, $\delta = dPoW$, where the maximum tolerable number of malicious miners can be represented by $v_3 = \lfloor \frac{2n_h-1}{2} \rfloor$, i.e., $\leq 50\%$.

Furthermore, to prevent forking and pruning issues in the proposed framework, we use the ‘longest chain’ rule [34]. It is possible that a miner may receive multiple blocks with the same set of transactions that may introduce a conflict of chains and therefore, effect a forking event. However, in our framework, upon receiving new block proposals, a miner is required to compare the length of the chain associated with each received block. Afterwards, the miner chooses the block which has the longest chain length and discards the other. If a miner receives blocks with multiple longest chains, he/she chooses one randomly. Note that the longest chain rule is widely used; including in ZCash, Bitcoin, and Monero.

#### 3) The 51% attack:
Let us demonstrate the validity of the aforementioned constraint and consider a case where an adversary $A$ plans to attack a blockchain by mining an alternative and dishonest chain faster than the honest chain that is mined by honest miners $M$. We can characterize this
attack, or in other words a competition, between $A$ and $M$ as a Binomial Random Walk (BRW) with the following events:

a. SUCCESS: If $M$ mines and extends the honest chain by one block, thereby taking a lead of $+1$.

b. FAILURE: If $A$ mines and extends the dishonest chain by one block, reducing the gap by $-1$.

Using these events, we now define $A$’s probability to catch up from $n + c$ blocks in terms of a Gambler’s Ruin problem, where $n$ is the number of blocks and $c$ is the associated checkpoint size. To formulate this, we assume that a player starts off with unlimited credit at a given deficit. He/she then potentially plays an infinite number of trials to reach his/her break even point and finally recover the deficit. Similarly, $A$’s probability to ever reach break even or alternatively, that $A$ ever catches up with the honest chain can be given as [10]:

$$Q_n = \begin{cases} \frac{1}{2} & \text{if } p \leq q \\ \left(\frac{q}{p}\right)^{n+c} & \text{if } p > q \end{cases}.$$  \hspace{1cm} (5)

where $p$ and $q$ are $M$’s and $A$’s probability to mine the next block, respectively, and $Q_n$ represents the probability that $A$ will catch up from $n + c$ blocks behind the honest chain or, in other words, longest chain. Note that the quotient $n + c$ here represent the block deficit and checkpoint size, respectively.

Figure 7 simulates this attack and demonstrates its ineffectiveness such that when $p = 1, 0.9, 0.8, 0.7$ and 0.6, the probability of $A$ to catch up with the honest chain $Q_n$ decreases exponentially with the increasing number of block deficit and checkpoint size, i.e., after just 10 blocks (for any combination of $n$ and $c$), $Q_n$ reduces to 0. When $p = 0.5$, it increases to 1, which confirms that whoever controls more than 50% of the computing power in the proposed framework, controls the blockchain. However, given our earlier assumption: $p > q$, as the number of block deficit and checkpoint size $A$ has to catch up with increases, $Q_n$ quickly drops exponentially. This justifies that by using a checkpoint of only size 2, the chances of $A$ to break an honest chain dwindle immediately.

A. Simulation design

ns-3\(^3\) was used to study and evaluate our framework, which is a discrete-event network simulator for large scale Internet systems. Using the mobility trace file, we conducted a custom SIoV simulation using the wireless access for vehicular environments (WAVE) protocol with IEEE 802.11p (@5.9GHz) standard (current state-of-the-art for vehicular networks), and continuous access to a 10 MHz control channel. Note that IEEE 802.11p has been developed to be technically compatible with DSRC standards, which is a suite of standards defined for communication of vehicular messages. It supports two types of devices: RSUs which are stationary and usually mounted along the side of the road, and on-Board Unit (OBU) that is mounted on a vehicle. The parameters used in this simulation are listed in Table III.

We considered both blockchain packets and application packets with Optimized Link State Routing (OLSR) routing protocol. Note that the application packets are application specific and therefore, can contain application relevant data. The simulation was run with 10 RSU and 100 vehicle nodes for 100 seconds. The vehicles included cars, buses, and trucks. Moreover, the movement of the vehicle nodes was captured with the mobility trace file within the $735 \times 735m^2$ sized map segment.

All vehicles send a 256-byte blockchain packet 100 times/second to the RSUs at a rate of 6Mbps, i.e., 100,000 blockchain packets per second. We consider such a high 1https://sumo.dlr.de/docs/index.html 2https://sumo.dlr.de/docs/Tutorials/OSMWebWizard.html 3https://www.nsnam.org/
broadcast rate to realize the very high traffic rate required for SIoVs. Note that RSUs here act as both miners and sink nodes. Additionally, the vehicles also continuously send 64-byte application packets to other nodes at a rate of 2.048Kbps for application specific purposes.

The routing protocol that we used in the simulation was OLSR with two-ray ground loss model coupled with Nakagami fading. The transmitting power of a vehicle was limited to 20 dBm and the RSUs were placed 100 meters apart. Note that the delivery of blockchain packets was calculated in terms of packet delivery ratio (PDR), which is the number of packets sent to an RSU divided by the number of packets it actually receives. We did this for all the 10 RSUs in the simulation to see the effect of channel fading over distance. Moreover, every vehicle also has physical layer (PHY) callback for tracing. This is used to determine the total amount of data transmitted, and then to calculate the MAC/PHY layers overhead beyond the application data.

### B. MAC/PHY overhead

The MAC and physical layer (MAC/PHY) overhead is an important performance metric in vehicular networks. Typically, the broadcast channel is busy in authenticating vehicles which affects the application throughput and may potentially overload the infrastructure, resulting in longer delays to authenticate. Therefore, MAC/PHY overhead is useful in performance evaluation of SIoVs and is given by:

\[
MAC/PHY\text{overhead} = \frac{(total\ PHY\ Bytes - total\ App\ Bytes)}{total\ PHY\ Bytes},
\]

where PHYBytes represent traffic at Layer 1 (physical) while AppBytes represent traffic at Layer 2 (data link). Its value lies in the range \([0, 1]\) and note that Layer 2 is responsible
for forward error correction and channel management (e.g., flow control and collision prevention). These functions are necessary for the network to operate properly, but they add processing overhead which results in lower throughput than at Layer 1. The overhead can be thought of as a moving average of the overhead of a system over time. Therefore, in well-designed security protocols for IoVs, the MAC PHY overhead should reduce with time.

C. Evaluation and discussion

It can be seen in Figures 9(a) and 9(b) how our framework performs in terms of receiving rate of application data and application packet. Note that data rate is bits transmitted per second and also recall that all vehicles are sending a 64-byte application packet at a rate of 2.048 Kbps. We can note from the figures that the average application throughput remains at around 18 Kbps while the application packet receiving rate is an average of 35 packets per second (pps). We note that our framework achieves the throughput upper bound of IEEE 802.11p in this case [43]. Moreover, Figure 9(c) shows the induced MAC PHY overhead and its reduction in value over time. Figure 9(d) shows the blockchain packets broadcast to the RSUs (PDR) by vehicles. As RSU1 is within the close proximity of a vehicle, it has the highest PDR. Similarly, RSU10 being the furthest, has the lowest due to the characteristics of fading. A summary of this discussion is presented in Table IV.

TABLE IV: Comparison of key parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Proposed</th>
<th>Yang et al. [13]</th>
<th>Improvement (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transmitted packet size</td>
<td>256-byte</td>
<td>800-byte</td>
<td>68</td>
</tr>
<tr>
<td>Receiving rate Throughput (pps)</td>
<td>35</td>
<td>30</td>
<td>16.67</td>
</tr>
<tr>
<td>Receiving rate (Kbps)</td>
<td>18.8760</td>
<td>13.6998</td>
<td>37.79</td>
</tr>
<tr>
<td>MacPHYOverhead reduction</td>
<td>56.371%</td>
<td>0.834459</td>
<td>5.83</td>
</tr>
<tr>
<td>RSU1 PDR</td>
<td>0.963481</td>
<td>0.621482</td>
<td>38.41</td>
</tr>
<tr>
<td>RSU2 PDR</td>
<td>0.809024</td>
<td>0.575125</td>
<td>29.78</td>
</tr>
<tr>
<td>RSU4 PDR</td>
<td>0.779408</td>
<td>0.512243</td>
<td>31.25</td>
</tr>
<tr>
<td>RSU5 PDR</td>
<td>0.71202</td>
<td>0.485201</td>
<td>35.76</td>
</tr>
<tr>
<td>RSU6 PDR</td>
<td>0.62116</td>
<td>0.390901</td>
<td>37.88</td>
</tr>
<tr>
<td>RSU7 PDR</td>
<td>0.57234</td>
<td>0.297587</td>
<td>48.93</td>
</tr>
<tr>
<td>RSU8 PDR</td>
<td>0.517475</td>
<td>0.297587</td>
<td>43.81</td>
</tr>
<tr>
<td>RSU9 PDR</td>
<td>0.457475</td>
<td>0.297587</td>
<td>37.88</td>
</tr>
<tr>
<td>RSU10 PDR</td>
<td>0.457475</td>
<td>0.297587</td>
<td>37.88</td>
</tr>
</tbody>
</table>

D. A comparative analysis

We now compare the dynamics of our framework with Yang et al. [13], where we simulate both frameworks using the parameters listed in Table III. Figure 10 presents a graphical depiction of the comparison while a summary of the results is documented in Table IV. We can see that the proposed framework outperforms [13] in most of the comparison fields.

Figure 10(a) shows that the application traffic receiving rate for [13] is 13.69 Kbps as compared to 18.87 Kbps of the proposed framework, i.e., a 37.79% improvement. Figure 10(b) shows that the throughput rate of application packets for [13] is approximately 30 pps as compared to 35 pps of the proposed framework. This is because the size of the transmitted packet in [13] is 800 bytes while in the proposed framework, it is 256 bytes.

Figure 10(c) shows that the MAC PHY overhead of the proposed framework is slightly higher than [13]. We can see that after 100 seconds, the MAC PHY overhead of our framework is at 56.371% while that of [13] is 51.925%. Furthermore, Figures 9(d) and 10(d) show that the proposed framework outperforms [13] in terms of RSU PDR at different locations. Thus, given a higher throughput rate with lower overhead and packet size, the proposed framework is able to accommodate more vehicles in lesser time, which can be verified by the PDR values in Table IV. Note that this also results in good scalability of the proposed framework.

VIII. Conclusion

Social Internet of Vehicles is currently limited by different constraints on scalability, security, and computing capacity. Therefore, traditional proof-of-work based blockchain frameworks are not suitable to address them. This paper proposed a two-dimensional blockchain based framework that uses a dynamic consensus together with a block checkpoint mechanism and a resource assignment policy. The blockchain is responsible for vehicle registrations and their access control. The dynamic consensus allows it to scale efficiently with increase in communication traffic of vehicles, whereas the checkpoint defines a mechanism to generate the next block hash. Moreover, the resource policy enforces secure distribution of resources among vehicles by edge modules for offloading tasks. These properties allow the framework to scale and accommodate the ever increasing transaction traffic associated with social vehicular networks, and provides rich computing resources for the vehicles to extend their resource limitations by offloading their tasks to the edge modules.

REFERENCES

Fig. 10: A comparative analysis of the blockchain frameworks.


Uzair Javaid received the B.Sc. degree (magna cum laude) in electrical engineering from the FAST-National University of Computer and Emerging Sciences, Peshawar, Pakistan. He is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering, National University of Singapore, Singapore.

His research interests include blockchain, applied cryptography, cyber physical systems, and network security.

Biplab Sikdar (Senior Member, IEEE) is an Associate Professor in the Department of Electrical and Computer Engineering at the National University of Singapore, where he serves as a Vice Dean in the Faculty of Engineering. He received the B. Tech. degree in electronics and communication engineering from North Eastern Hill University, Shillong, India, in 1996, the M.Tech. degree in electrical engineering from the Indian Institute of Technology, Kanpur, India, in 1998, and the Ph.D. degree in electrical engineering from the Rensselaer Polytechnic Institute, Troy, NY, USA, in 2001.

He was an Assistant Professor from 2001 – 2007 and Associate Professor from 2007 – 2013 in the Department of Electrical, Computer, and Systems Engineering at Rensselaer Polytechnic Institute from 2001 to 2013. His research interests include IoT and cyber-physical system security, network security, and network performance evaluation. Dr. Sikdar is a member of Eta Kappa Nu and Tau Beta Pi.

Dr. Sikdar served as an Associate Editor for the IEEE TRANSACTIONS ON COMMUNICATIONS from 2007 to 2012 and as an Associate Editor for the IEEE TRANSACTIONS ON MOBILE COMPUTING from 2014 – 2017.