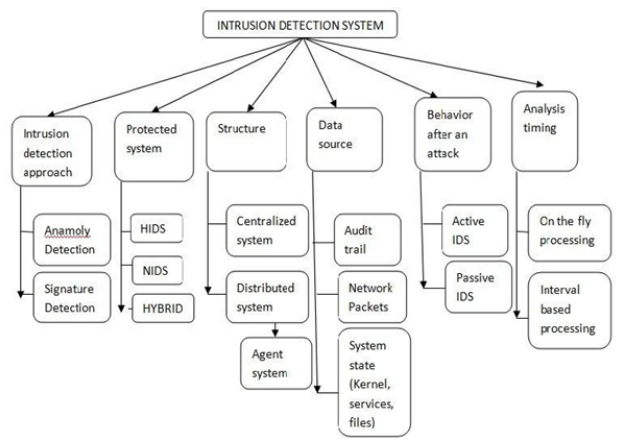
Countermeasures:

1. Reference: CAN-Security-CounterMeasures
   1. In this section, two exemplary countermeasures are being discussed that could help to increase the IT security of future automobiles by addressing these problems like the basic weaknesses exploited in our practical tests.

* 1. Intrusion detection techniques:

An Intrusion Detection System (IDS) is a network-based security system designed to identify and respond to malicious activities within a targeted network. The software architecture of an IDS involves analysing network traffic to detect and thwart intrusion attempts promptly. Upon detecting malicious activities in the network or on a specific node, the IDS responds by blocking the identified malicious node's access to prevent further harm to the network. This response includes preventing the suspected IP address, port, or user associated with the malicious activity from accessing the network.

Intrusion Detection Systems (IDS) can be categorized based on different classifiers. They are classified into three types according to the target of intrusion detection. Network-based Intrusion Detection Systems (NIDS) operate at the network level to identify malicious nodes or intrusion attempts, while Host-based Intrusion Detection Systems (HIDS) function locally on specific nodes to detect intrusions. Wireless-based Intrusion Detection Systems (WIDS) are designed for wireless networks and share similarities with NIDS. IDS can also be classified based on detection approaches, distinguishing between Misuse Detection and Anomaly Detection. Additionally, they can be categorized as Passive IDS or Active IDS based on their behavior. There are two further types based on the systems employed: centralized IDS and distributed IDS. The data collection methods for intrusion detection may involve network-based, host-based, or a combined approach incorporating both host and network-based methods.

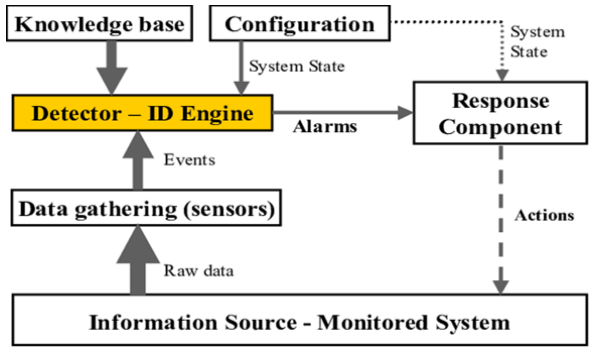


*Figure 1. Classification of Intrusion Detection Systems: This Figure presents the classification of Intrusion*

*Detection Systems on the basis of various factors.*

**IDS Architecture:**

The architectures employed in Intrusion Detection Systems (IDS) can be broadly categorized as centralized or decentralized. A centralized IDS involves data collection and analysis on a single centralized system, making it effective for intrusion detection in a single system where the locations for data analysis remain fixed. On the other hand, decentralized IDS operates by distributing data collection and analysis across multiple hosts, particularly suitable for scenarios involving multiple systems. In decentralized architectures, security analysis is conducted at various network sites, with the number of analysis locations scaling proportionally to the number of host machines under observation. The initial IDS architecture was proposed by Dorothy Denning, and while various models have since been introduced, a fundamental architecture, depicted in Figure 2, is commonly employed across many intrusion detection systems.



*Figure 2. Basic Architecture of an IDS [10]. This figure presents the basic architecture of an Intrusion Detection*

*System, highlighting all its important components.*

**Reference:** Ila Naqvi 1 , Alka Chaudhary 1 , Anil Kumar

* 1. CANet: neural network architecture for detecting intrusions on the controller area network (CAN). This is the first deep learning based intrusion detection system that naturally handles the data structure of the high dimensional CAN bus.

Neural Networks Building blocks:

* + 1. Long Short-Term Memory (LSTM):

The LSTM is a neural network architecture specifically designed to work on time series or natural language processing problems. LSTMs belong to the class of recurrent neural networks (RNNs).

* + 1. Autoencoder:

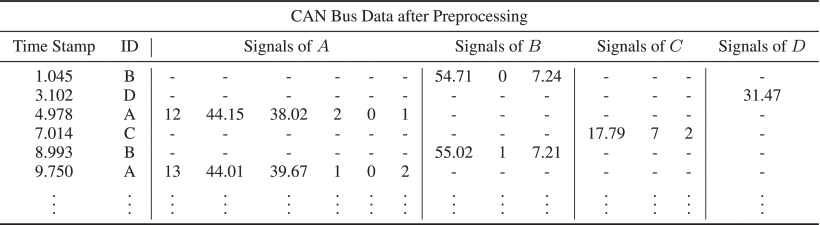
The class of neural network structures that is known as autoencoders has the objective to learn the structure of a training data set, e.g. the normal behavior of a technical system, in an unsupervised manner. This is, no labeled data is needed in the training process. A trained autoencoder can be used to identify data points that deviate from the normal behavior and therefore can function as anomaly detection system. The basic idea behind autoencoders is that high dimensional data that originates from some underlying system often contains correlations. Therefore, a meaningful embedding into a lower dimensional subspace typically exists. An autoencoder consists of two neural network blocks, an encoder and a decoder. Both can be realized by a set of consecutive fully connected layers. The task of the encoder is to map the high dimensional input data into the lower dimensional embedding space, which is called autoencoder bottleneck. The task of the decoder is to reconstruct the input data from its representation in the embedding space. Whereas the deviation between the real input data and its reconstruction should be small on normal data, large deviations are expected for anomalous data that has not been seen by the system in the training process.

* + 1. Exponential Linear Unit (ELU):

ELU is a nonlinear activation function used in deep neural networks. It is defined as

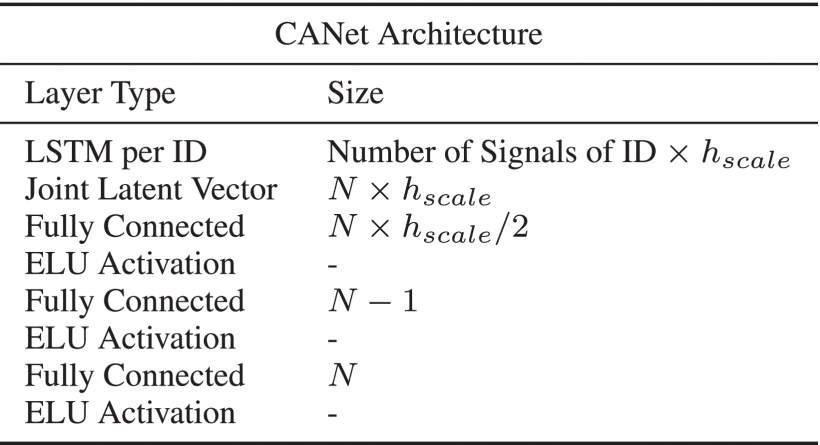


The paper addresses the growing concern of potential remote attacks on electronic control units (ECUs) in connected automobiles, emphasizing the need for effective intrusion detection systems (IDS) to ensure traffic safety. Focusing on the controller area network (CAN) bus, a common vehicle bus standard, the authors note the limitations of rule-based and statistical IDS methods and propose a novel neural network architecture called CANet. The architecture is tailored to handle the multidimensional signal space of CAN data, outperforming existing methods and demonstrating efficacy in detecting unknown attack types. The paper highlights CANet's strength in identifying signal manipulations that may be challenging for classical approaches, leveraging the network's ability to learn physical relationships between signals. The proposed method, trained in an unsupervised manner, shows promise beyond intrusion detection, potentially serving as a tool for anomaly detection and early identification of technical failures. The document concludes with an organization of sections covering background, network architecture, evaluation, and conclusions.

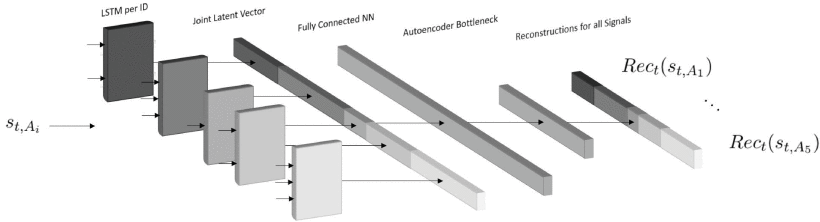


*TABLE 1: Schematic Representation of CAN Bus Data after Preprocessing its Payload Bytes to Signals*

CANet is an unsupervised learning method. The basic idea is to handle the challenging structure of CAN data by introducing an independent LSTM input model for each ID that can capture the temporal dynamics of the corresponding signals. The output of all input models is aggregated and fed into a fully connected subnetwork with an autoencoder structure. This enables the network to take interdependencies of signals of all IDs into account. At each point in time, all potential input signals are reconstructed. The reconstruction error between the true signal values and their reconstruction can be used to calculate the anomaly score. The topology of CANet with the corresponding layer sizes and activation functions are summarized in Table 2. A visualization of the architecture can be found in Figure 1.



*TABLE 2. CANet Architecture for Detecting Intrusions on the CAN Bus*



*FIGURE 1: Schematic representation of the CANet architecture for intrusion detection on CAN bus data. Each ID has its own LSTM input model. When the payload st,Ai of an ID is fed into its input model, only the corresponding memory in the joint latent vector is updated. The entire latent vector is used to reconstruct the signals of all IDs..*

Basic idea: handle challenging structure of CAN data by introducing LSTM input model for each ID – then output of all input models is fed into a fully connected subnetwork with an autoencoder structure – it enables the network to take interdependencies of signals of all IDs

Network Architecture: The described network architecture involves a set of IDs, denoted as A={A1,…,AK}, where each ID A has corresponding signals encoded in the payload. The architecture includes an input LSTM subnetwork for each ID A, with the i-th LSTM associated with ID Ai having nAi inputs and a hidden dimension of nAi⋅hscale. The computational power of the network is represented by hscale. The LSTM subnetwork processes the signals, and the output updates a joint latent vector, concatenating hidden states of all LSTM models. This vector, with a length of N⋅hscale, reflects the current state of the CAN traffic.

Following the joint latent vector, fully connected layers, including an output layer with N neurons, aim to reconstruct input signals from all IDs. During training, the payload of each ID is fed through its LSTM model to update the joint latent vector, facilitating the reconstruction of all payloads at time step t for all IDs.

The reconstruction of the payload at time step t is denoted as



where Rect(st,Aj) represents the reconstruction of the payload associated with ID Aj. The training involves comparing true signal values with their reconstructions, using a quadratic error loss function



Importantly, the temporal dependencies for each ID are stored separately in the corresponding LSTM subnetwork, making the model insensitive to the exact order of consecutive message IDs during training. This feature accommodates the variability in the order of IDs observed in real CAN data.

Anomaly Score: The quadratic error between a signal and its reconstruction is utilized to identify anomalies at time step t∈N. This unsupervised learning approach involves setting a fixed threshold based on normal data for each signal. The threshold is determined as a percentile P of quadratic errors from a small dataset. During evaluation, an anomaly indicator is maintained for each signal. When processing an ID, the indicator is updated to 1 if the reconstruction error exceeds the corresponding percentile P and set to 0 otherwise. The global anomaly score at time step t is set to 1 only if at least one signal has an anomaly indicator of 1; otherwise, it is set to 0. This process allows for anomaly detection in the absence of labeled anomaly data, relying solely on the characteristics of normal data.

Experiments: In this section, the authors evaluate their method on both real and synthetic CAN data.

The real data utilized in the experiments was collected from a test vehicle, considering 13 IDs with a total of 20 signals. The selection of signals ensured the inclusion of physical values, with each signal having at least one other signal showing a physical dependency. The authors divided approximately 13 hours of recorded data into 12.5 hours for training and 0.5 hours for testing. The analysis focused solely on data representing normal driving conditions, excluding events such as engine start and shutdown. All payloads underwent preprocessing into their respective signal value spaces (refer to Table 1).

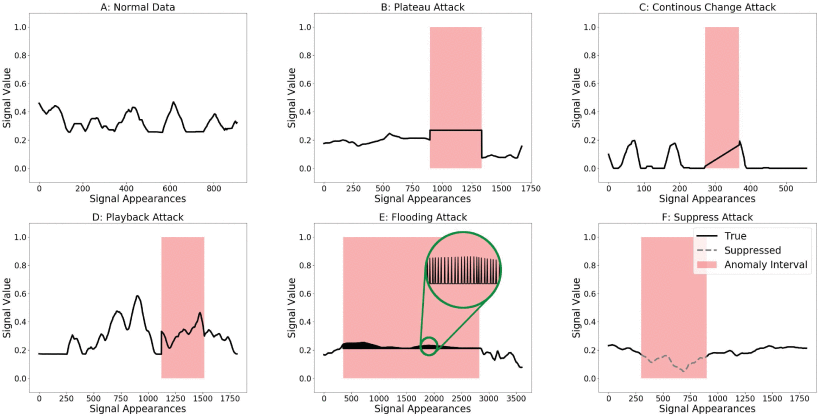
For the synthetic data, the authors worked with a dataset comprising 10 different message IDs, each having varying numbers of signals and noisy time frequencies, totaling 20 signals. The synthetic data was generated to closely resemble real CAN traffic, containing physical values, counters, and signals dependent on one or multiple other signals. The training dataset comprised about 16.5 hours, and the test dataset included approximately 7.5 hours of CAN traffic. The dataset is publicly available at <https://github.com/etas/SynCAN>.

**A. Simulated Attacks:**

In both the real and synthetic datasets, the test data is partitioned into six subsets of equal time length. One subset is reserved for evaluating the model's performance on normal data, while the remaining five subsets are utilized to assess the model's effectiveness against various attack types.

1. Plateau Attack: This type involves overwriting a single signal with a constant value over a period, causing a sudden jump or freezing of the signal.
2. Continuous Change Attack: Signals are gradually overwritten, drifting away from their true values. This tactic aims to deceive the Intrusion Detection System (IDS) by introducing realistic small changes in the signal.
3. Playback Attack: Signal values are replaced over time with a recorded time series, creating a scenario where the attacker attempts to trick the IDS with genuine-looking values from a different traffic situation.
4. Flooding Attack: The attacker floods the CAN bus with messages of a specific ID at a high frequency. This attack is practical as it doesn't require control over an Electronic Control Unit (ECU), only the sending of additional messages to overwrite real message values.
5. Suppress Attack: In this attack, an ECU is prevented from sending messages, such as by turning it off. This results in the absence of messages for a particular ID in the CAN traffic for a certain period.

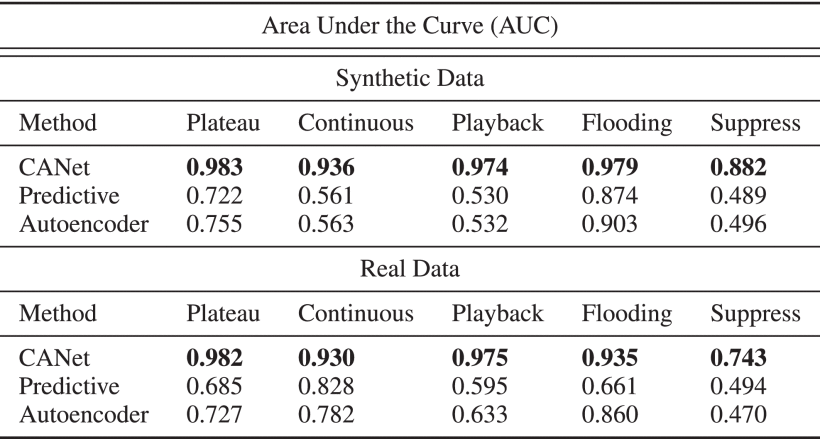
The typical duration of an attack interval in both datasets ranges from 2 to 4 seconds. Visualization of each attack type is provided in Figure 2.



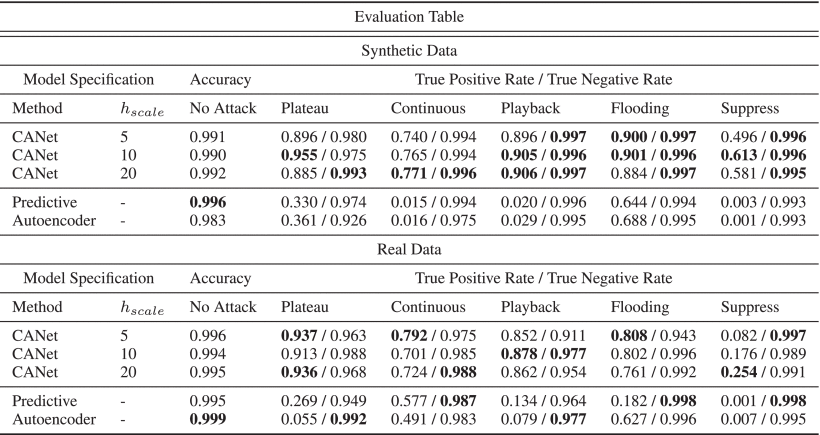
*FIGURE 2: Visualization of CAN data with different attack types on short time intervals. The flooding attack contains high frequency anomalies between its real signal values. In the suppress attack the dotted line represents real signal values that are not transmitted onto the CAN bus.*

**B.** **Network Training Details:**

In this section, the authors present the training details of their method to ensure reproducibility of the results. The code for training and evaluation is implemented in Python 3.5.3 using pyTorch 0.3.0 [30]. The computations are executed on a system with a 3.5 GHz processor, 4 cores, and 32 GB of RAM. The chosen network architecture, as outlined in Table 2 for various hscale values (refer to Table 4), is employed. The optimization is performed using the Adam optimizer with an initial learning rate of 0.01. Input signals to the network undergo a signal-wise normalization to the range of 0-1.



*TABLE 3: AUC of the ROC Curves on Real and Synthetic CAN Data*



*TABLE 4: Summary of Experimental Results on Real and Synthetic CAN Data*

**C. Evaluation:**

Summing up the results, it is observed that CANet demonstrates reliable detection capabilities for attacks on CAN bus data, exhibiting solid performance on normal data. The main findings include:

* The presented architecture stands out as the first method capable of effectively handling the complex data structure arising from signals of multiple CAN IDs within a single model.
* CANet surpasses the baseline CAN Intrusion Detection System (IDS) methods by a significant margin across all selected evaluation criteria.
* The model is trained in an unsupervised manner, having never encountered attacks during training. Consequently, it is anticipated that the model can identify additional unknown attack scenarios beyond those explicitly presented in the research paper.

**Extra**: Risks and Benefits of Neural Network Based IDs Models:

The manuscript highlights the primary advantage of machine learning-based approaches, exemplified in the presented method, as their potential capability to detect unknown intrusions, a task where conventional methods often fall short. Unlike traditional approaches that necessitate selecting a defense mechanism for each possible attack scenario, machine learning, particularly neural networks, significantly reduces development time and the need for extensive domain knowledge in the CAN bus domain. However, challenges arise in the complexity of analyzing machine learning outputs, posing difficulties in automating responses upon intrusion detection. Additionally, neural networks demand substantial training data and are generally more computationally and memory-intensive compared to alternative approaches.

**Conclusion:**

The authors present CANet, a novel neural network architecture that is trained in an unsupervised manner to detect intrusions and anomalies on the CAN bus. It is the first model in the literature capable of working on messages with different IDs simultaneously. CANet models have a high true negative rate, typically over 0.99, which is necessary for real world applications. Additionally, along with the high true negative rate we are able to detect a large amount of the unknown attacks, both on real and synthetic data correctly.

**Reference**: Hanselmann M, Strauss T, Dormann K, Ulmer H (2020) Canet: an unsupervised intrusion detection system for high dimensional can bus data. IEEE Access 8:58194–58205 Link: <https://ieeexplore.ieee.org/document/9044377>

* 1. Blockchain enabled data security in vehicular networks:

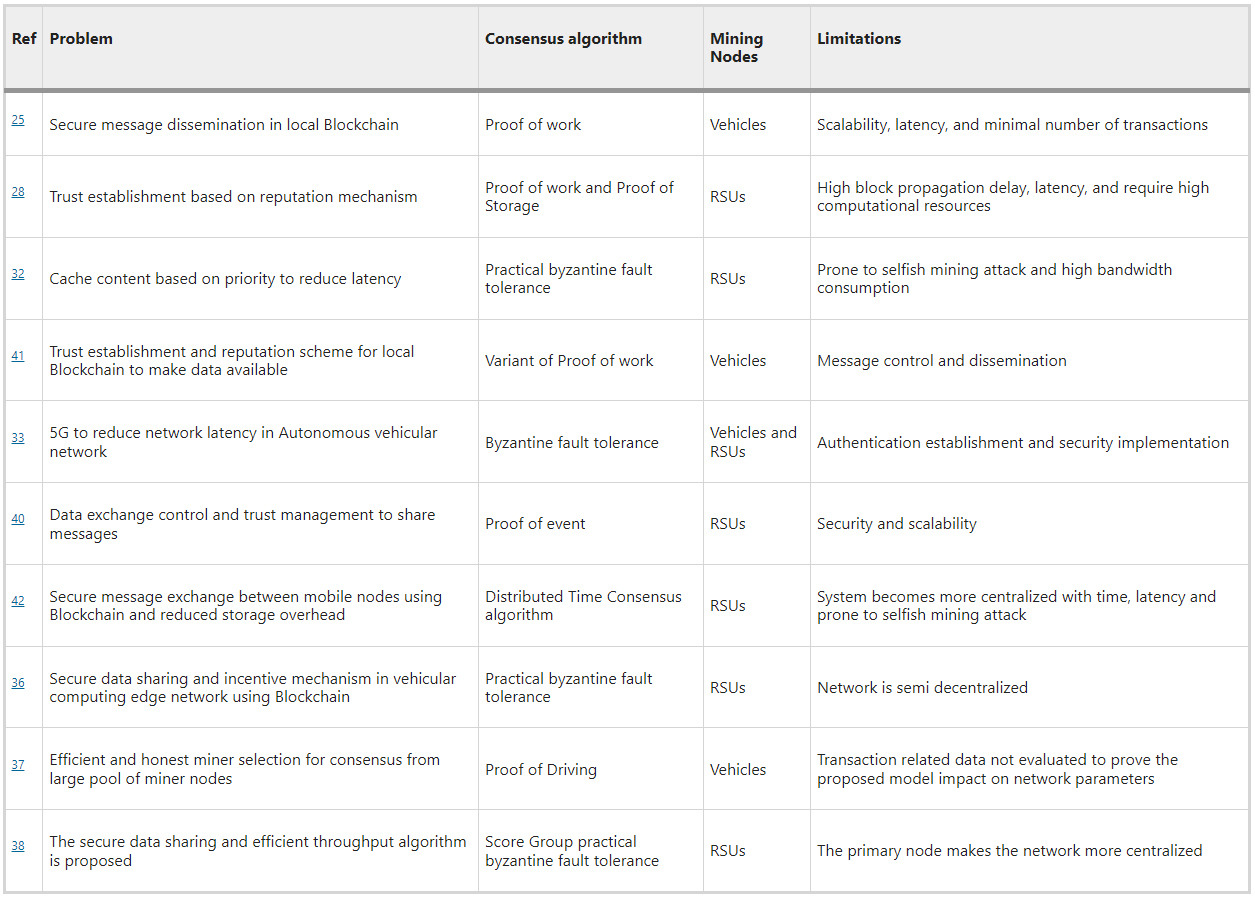
The article discusses the role of Intelligent Transportation Systems (ITS) in facilitating communication between vehicles and infrastructure, with a focus on the challenges faced by vehicular networks. It highlights the emergence of Autonomous Vehicles (AVs) in ITS, equipped with onboard resources for tasks such as object detection and congestion monitoring. The use of 5/6G technologies, Road Side Units (RSUs), and Mobile Edge Computing (MEC) servers for efficient communication and processing in AVs is emphasized. Privacy concerns in data sharing among nodes in ITS are addressed, and the potential of blockchain technology to enhance security and decentralization is explored. The passage introduces a proposed Vehicular network Based Consensus Algorithm (VBCA) for secure data sharing within vehicular networks, drawing inspiration from the Raft consensus mechanism but with customizations. The algorithm aims to improve decentralization, security, and reliability in vehicular networks, with a separation of transaction and block confirmation processes.

The main contributions of this paper are summarized as follows:

* We propose a VBCA algorithm using smart contracts to improve the network throughput in a blockchain-based vehicular network.
* The smart contract mechanism is used to ensure decentralization by letting multiple RSUs append a block in a ledger. A leader selection process is also proposed to ensure the decentralization in block creation. This also addresses the selfish mining problem.
* We have also reduced the transaction latency by proposing the separation of transaction confirmation and block creation process.
* The proposed technique demonstrates improvement in results when compared with the state-of-the-art mechanisms in terms of throughput and the number of blocks created per node in the vehicular network while ensuring decentralization.

Related work:

The paragraph discusses the growing application of blockchain technology in various domains, including product lifecycle management, smart tracking, tracing, and smart transportation. It specifically highlights the relatively new area of research focusing on blockchain in vehicular networks. The lack of a standardized categorization for blockchain-based schemes in this context is noted. The section then presents recent works in the field, organizing them based on the type of blockchain, consensus algorithms, and applications. Towards the end of the section, a comparative summary of the literature is provided in Table 1.



*Table 1: Comparative analysis of Blockchain in Vehicular networks.*

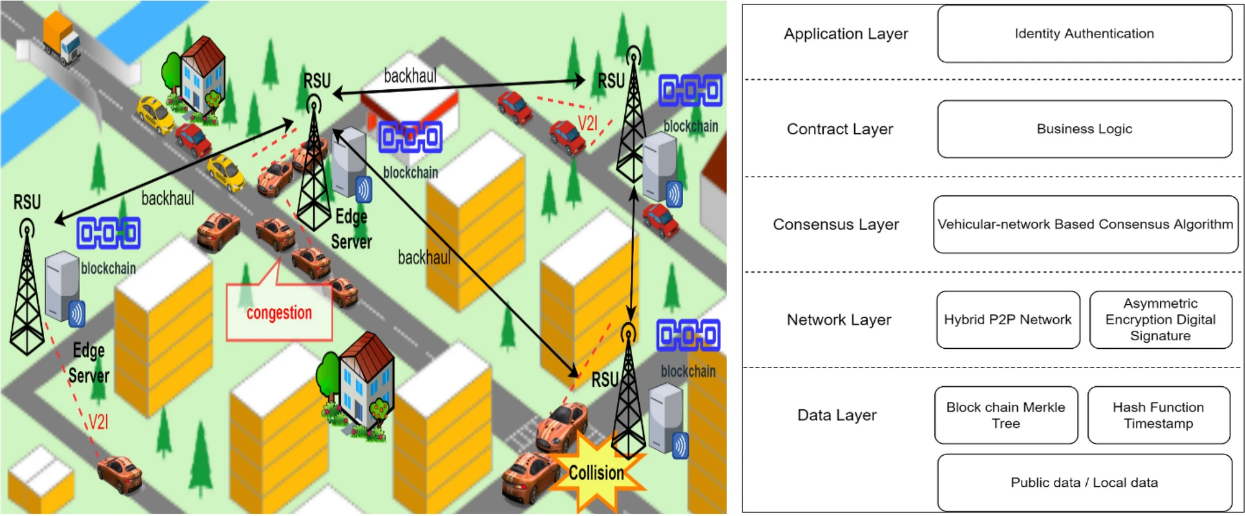
**Public blockchain based solutions in vehicular networks:**

Shrestha et al. proposed a secure message dissemination approach for vehicles using a local blockchain, where the Proof of Work (PoW) consensus algorithm was employed to store trust weights of messages and vehicles as transactions. Cars with high computational power participated in mining, and RSUs handled authentication and provided location certificates. To enhance scalability, the approach suggested maintaining separate blockchains for each region. However, it did not address block propagation delays arising from an increasing number of vehicles in a local blockchain. Additionally, the PoW algorithm's limitation of processing seven transactions per second was noted as insufficient for delay-sensitive vehicular communications.

Yang et al. developed a model for secure communication and trust management in vehicular networks. They aimed to enhance miner selection by limiting the number of nodes, calculating a vehicle's rating for messages sent to another vehicle, and submitting it to RSU. RSUs rank vehicles on the blockchain based on these ratings, and a consensus among RSUs, acting as nodes, is reached using a joint Proof-of-Work and Proof-of-Stake algorithm. While the model ensures integrity and confidentiality through blockchain, privacy concerns persist due to the use of vehicle identification numbers as public key identifiers. Additionally, the model incurs communication overhead as the rating packet size exceeds the message packet size. Another study using the NS3 simulator found that the presence of a selfish miner increased block receive time from 10 to 19 minutes.

**System Architecture:**

The discussed section outlines the key components and layered architecture of the proposed system, distinguishing between two node types: Stationary nodes and Mobile nodes. Stationary nodes, represented by Roadside Units (RSUs) connected to high-power edge servers, offer geographic coverage to specific regions and are interconnected via backhaul links. These nodes handle blockchain operations, with each node appending a block to the blockchain, ensuring redundancy across the network. Mobile nodes, which include vehicles, capture event data through sensors and transmit it to the nearest stationary node. Utilizing the Dedicated Short Range Communication protocol (DSRC) in a peer-to-peer network, vehicles establish reliable connections with nearby RSUs, facilitating efficient communication and reducing latency between peers. Overall, this architecture supports decentralized communication and data handling for improved traffic management and event reporting.

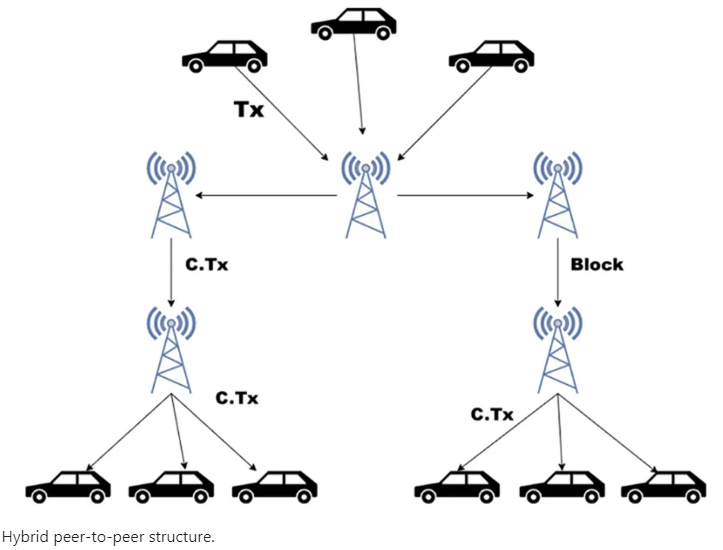


*Figure 1: Vehicular blockchain architecture*

The proposed system operates on edge servers, executing a consensus algorithm for appending validated transactions to the blockchain stored on these servers. Vehicles equipped with diverse sensors contribute substantial data to the edge servers. Machine learning tools and statistics are applied to the collected data to train models, enabling predictions for various applications, such as forecasting traffic load on different transportation network sections. The system comprises layers: Application, Contract, Consensus, Network, and Data. The Application layer facilitates user interactions, with input passed to the Contract layer for authentication and data access control via smart contracts. The Consensus layer employs a custom algorithm to establish trust among nodes, while the Network layer connects nodes in a hybrid P2P manner, managing key protocols and cryptographic algorithms for security. The Data layer ensures the integrity and security of transactions and blocks using hash functions, timestamps, and a Merkle tree structure. The system operates on a consortium blockchain, combining public and private blockchain features, with vehicle registration involving acquiring a public key pair from a certification authority (CA) for integrity and authenticity. Vehicles submit this pair to stationary nodes, which verify it with the global CA and store it on the network.

**Hybrid peer-to-peer network:**

The described network architecture diverges from conventional blockchains through its adoption of a peer-to-peer (P2P) and distributed framework, as illustrated in Figure 2. The proposed system operates as a hybrid P2P, designating a stationary node as a central access point for vehicles within its range. Transactions undergo verification by the leading stationary node, and only those verified transactions are disseminated to other stationary nodes. Subsequently, confirmed transactions are broadcasted by the leader to other stationary nodes, extending across the entire network. This hybrid approach combines P2P principles with a structured hierarchy, enhancing the efficiency and coordination of transaction validation and dissemination within the network.



*­­­­­­­­­Figure 2: Hybrid peer-to-peer structure*

**Proposed Methodology:**

First, the necessary assumptions forming the foundation of the proposed methodology are presented:

(a) The methodology relies on cryptographic primitives, such as encryption schemes and hash functions, to establish secure communication channels between entities, including communication between vehicles and Roadside Units (RSUs) as well as between RSUs.

(b) A critical assumption is made that a malicious node within the network lacks the capability to compromise half of the nodes. This assumption is considered reasonable in practical scenarios.

(c) The hardware equipped in RSUs and Certificate Authorities (CA) is assumed to possess high computational power, ensuring the effectiveness of the proposed methodology.

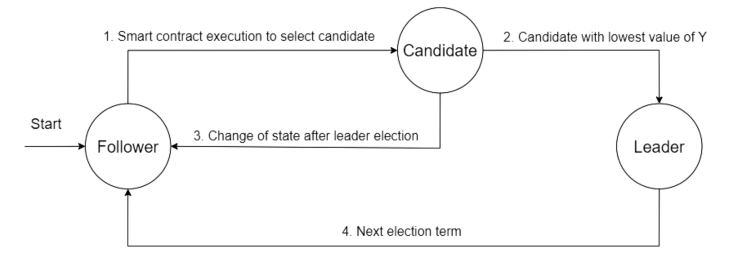
In the system, transactions sent from mobile nodes to stationary nodes are regarded as interactions, initiated by On-Board Units (OBUs) in vehicles. These transactions are digitally signed by the mobile nodes and transmitted to nearby stationary nodes. The blockchain technology employed relies on Public Key Infrastructure (PKI) for transaction verification. Assuming vehicle registration with a Vehicular Authority (VA) and the allocation of a public key pair to each vehicle and Roadside Unit (RSU), a vehicle encrypts a transaction (Tx) containing message data, a timestamp, and the hash of the message data. The encrypted transaction is sent to the relevant stationary node, which uses the vehicle's public key to decrypt it, obtaining the hash value. The stationary node then generates the hash of the original received message and validates the transaction by comparing the generated hash with the one sent by the vehicle. Validated transactions are added to a transaction pool by the leading RSU and simultaneously relayed to other stationary nodes, ensuring rapid information dissemination in emergency situations.

**VBCA consensus algorithm:**

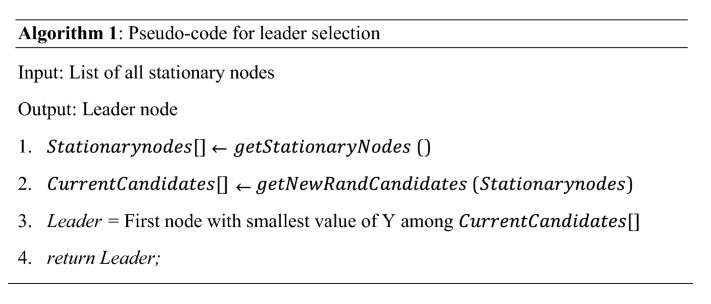
In this section, the proposed VBCA algorithm for blockchain-based vehicular networks is elucidated. The consensus algorithm is exclusively executed at the stationary node level and is subsequently dissected into sub-components, each of which will be expounded upon in the following sections.

**Leader RSU selection:**

In the consensus process, as depicted in Figure 3, a stationary node has the capability to transition among three states: follower, candidate, or leader. The synchronization of stationary nodes is facilitated through a smart contract. The leader selection process is outlined in Algorithm 1. Initially, a list encompassing all stationary nodes is acquired (Line 1). Subsequently, candidate nodes are randomly chosen from the follower nodes through the smart contract, with the condition that the selected candidate node did not remain a candidate in the previous term (Line 2). These candidates then undergo a leader selection process utilizing a Verifiable Random Function (VRF) among themselves, where each candidate possesses a pair of public and private keys. Using a seed string Q known to the entire network, candidates calculate values (Y, p) = VRF(Pr, Q), where p represents the proof of work. The candidate node that computes the lowest value of Y below a specified threshold is ultimately selected as the leader for the current term (Line 3).



*Figure 3: Leader Selection Process*

****

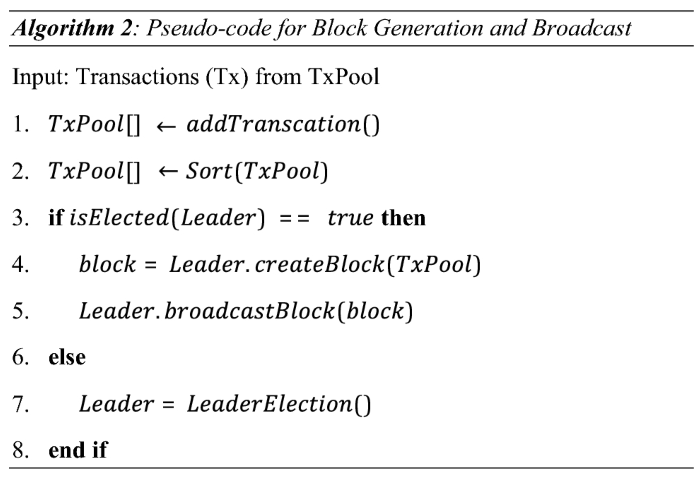
The leader creates a block and propagates it across the network of stationary nodes. Using the proof p and public key of the leader node, any candidate node can verify whether Y is computed correctly or not.

**Block generation:**

The block generation process occurs after the leader selection. Transactions are initially validated and added to the transaction pool (TxPool). These transactions are then sorted based on timestamps.

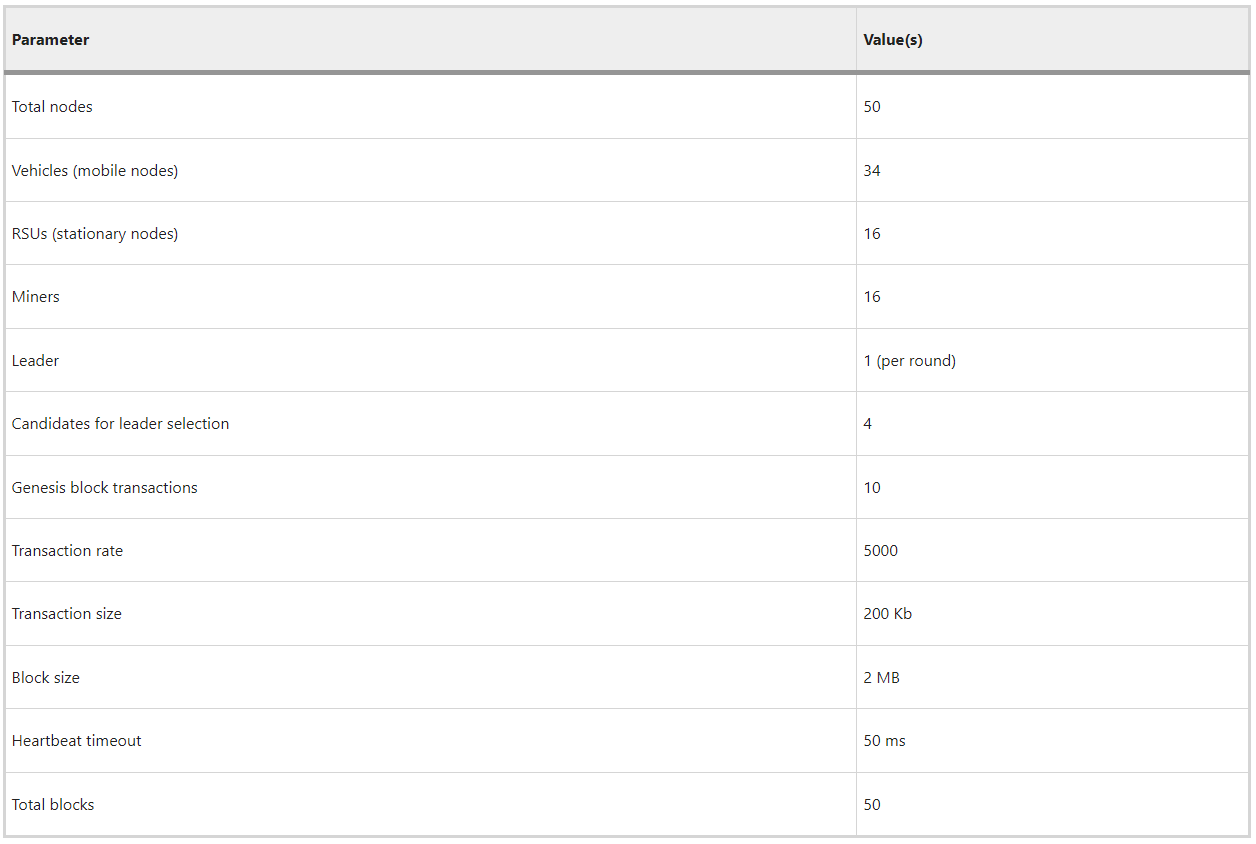
The elected leader, in Line 3–Line 6, selects transactions from the pool, choosing 10 based on a first-in-first-out structure. The leader encrypts the transactions, along with their timestamps, transaction hashes, smallest Y value, and the hash of the previous block, using their private key. The resulting block is broadcasted to other RSU nodes. Follower nodes, upon receiving the block, validate it by decrypting with the leader's public key (known to all nodes) and verifying the Y value. Validated blocks are appended to the ledger, and acknowledgments are sent to the leader. The leader, in turn, waits for acknowledgments based on a heartbeat timeout.

If the 'if statement' in Line 3 is false, indicating that the leader selection process is not successful, Line 7 is executed to handle leader selection.



**Performance evaluation:**

The authors state that, to their knowledge, there is currently no specialized simulator for implementing blockchain on vehicular networks. They highlight the widespread use of Network Simulator 3 (NS-3) for simulating various networks with customized requirements and parameters. Drawing inspiration from a previous study, they consider NS-3 to be a suitable choice for their proposed work, given its support libraries for addressing blockchain's specific needs. The authors conduct simulations of their proposed VBCA algorithm using smart contracts within a vehicular network. They compare their system with other consensus algorithms, namely Distributed Time Consensus (DTC), Practical Byzantine Fault Tolerance (PBFT), and Proof of Work (PoW). The simulation parameters are detailed in Table 2.



*Table 2 Simulation parameters and values.*

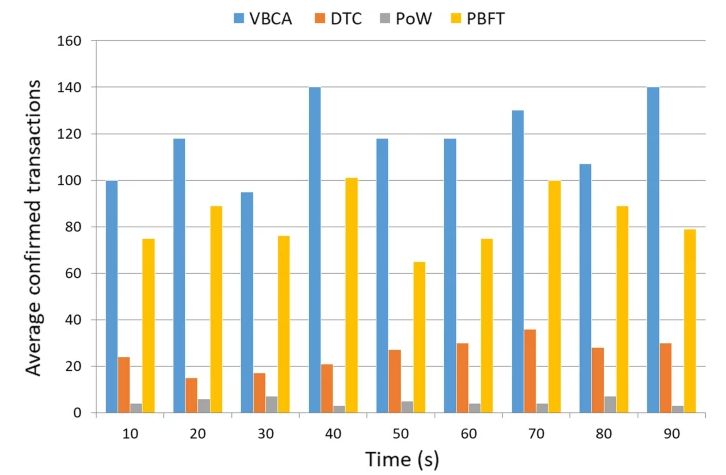
The comparisons involve several performance parameters: (a) block confirmation time, (b) throughput, (c) transaction latency, and (d) confirmed transactions. Block confirmation time is defined as the duration from block creation to its confirmation. Throughput is measured as the number of confirmed transactions per second. While a large number of transactions may be generated, the focus is on the confirmed transactions in the results.

The average transaction latency for T transactions is given as:



**Impact of time on throughput:**

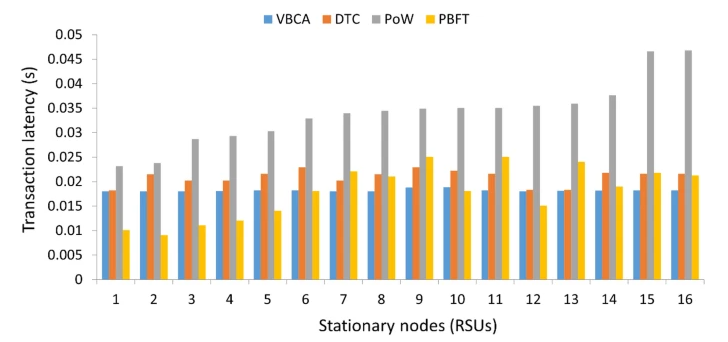
In Figure 4, the evaluation results depict superior performance of the proposed VBCA compared to related schemes, attributed to its leader selection process minimizing transaction confirmation time. When a leader stationary node confirms a transaction, other nodes no longer need to await block creation, enhancing overall throughput. Vehicular networks generate messages per second requiring confirmation within a deadline. In contrast, the PoW scheme's slow mining algorithm limits confirmation to a maximum of 7 transactions per second, adversely affecting throughput.



*Figure 4: Transactions confirmed per second (throughput).*

**Transaction latency at stationary node:**

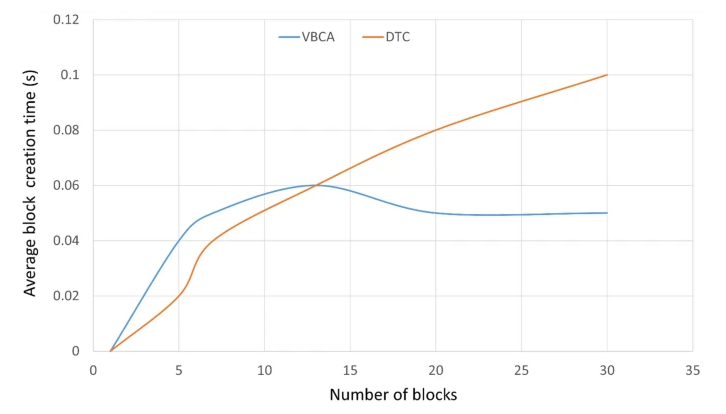
Figure 5 illustrates transaction latency across nodes, revealing that the proposed VBCA outperforms existing schemes, except for PBFT. Notably, PBFT, operating in a fully private network, demonstrates the shortest transaction confirmation time. In contrast, PoW exhibits higher latency as each peer in the network must confirm the transaction. The observed lower transaction latency of VBCA at stationary nodes is attributed to the consortium blockchain, where only the leader stationary node is responsible for confirming transactions into the pool.



*Figure 5: Transaction latency at each RSU.*

**Impact of block creation:**

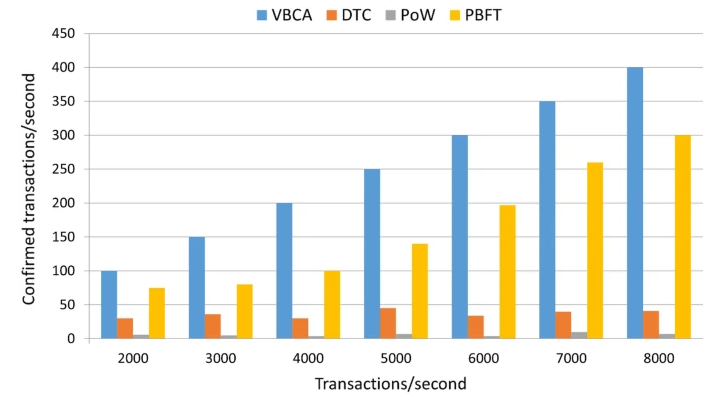
The number of transactions for the genesis block creation is set to ten in the system. The genesis block, which is the first block in the blockchain network, is static. To maintain decentralization, different leaders create varying numbers of blocks. Each subsequent block consists of 10 transactions. In the proposed VBCA, the block creation time is capped at a maximum threshold of 0.05 seconds. In contrast, the mechanism employed by DTC42 initially increases the block creation time until a trusted block creator is selected. Once selected, the block creation time is fixed at 0.1 seconds. The graph in Figure 6 illustrates that the block creation time for VBCA stabilizes and becomes constant after approximately 20 blocks. This stability is attributed to the non-selfish nature of the block creation process in the proposed scheme, where blocks must be created within a fixed threshold to prevent selfish mining attacks. The block creation times for other schemes, which have larger values, are not displayed in the graph.



*Figure 6: Block creation time*

**Impact of transaction rate on confirmed transactions:**

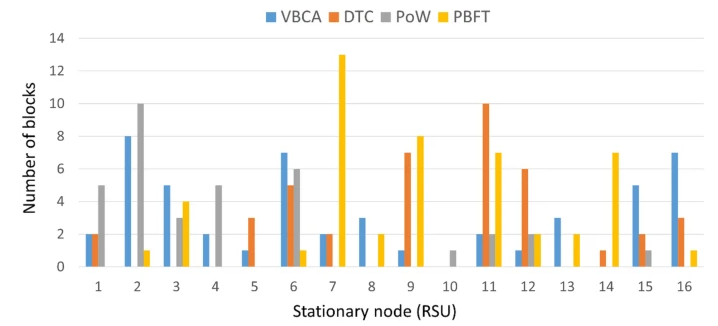
The evaluation of various consensus schemes against transaction rates revealed that VBCA outperforms other approaches, as illustrated in Fig. 7. The higher transaction confirmation rate of VBCA can be attributed to its Hybrid P2P structure. In this scheme, after transaction validation, the stationary node involved forwards the transaction to the leader node, eliminating the need for each individual node in the network to undergo the lengthy validation process. This design choice enhances transaction efficiency and contributes to VBCA's superior performance in comparison to alternative consensus methods.



*Figure 7: Variation in transaction rate*

**Impact of number of blocks per RSU:**

In the context of ledger management, the selection of stationary nodes responsible for appending blocks to the ledger is crucial for ensuring system decentralization. Figure 8 illustrates that VBCA achieves decentralization by distributing block creation across almost every node. In contrast, PoW25 tends to favor nodes with greater resources, resulting in higher network resource consumption, centralization, and an increased occurrence of orphan blocks. On the other hand, the trust table-based DTC scheme (scheme42) is noted for its centralization, making the system more vulnerable to a single point of failure through potential attacks. VBCA enhances decentralization by preventing the consecutive selection of the same leader, thereby allowing other nodes the opportunity for block creation.



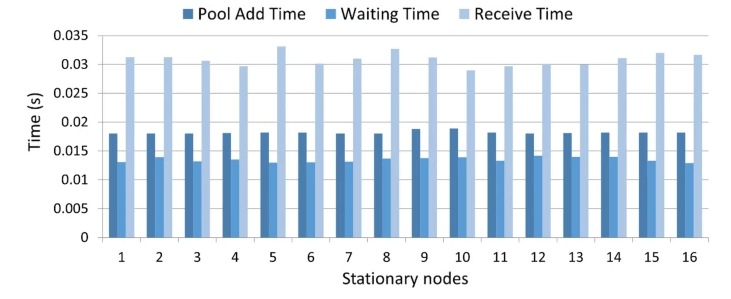
*Figure 8: Block creation per node*

**Transaction confirmation time for block:**

In Figure 9, the transaction confirmation time per block is illustrated. The time at which the leader confirms a transaction by broadcasting it to the network and adding it to the pool is denoted as transaction confirmation time. The transaction confirmation time per block is given as follows:







*Figure 9: Transaction confirmation for block*

In the described system, T Addi represents the time when a leader chooses a transaction from the pool for inclusion in a block, and subsequently, the block is disseminated to stationary nodes for incorporation into their ledgers. Additionally, AckTimei denotes the duration the leader awaits acknowledgments from each stationary node before finalizing the block addition.

The results indicate that existing methods experience longer transaction confirmation times due to their resource-intensive consensus algorithms. This delay contributes to heightened network latency and decreased throughput. Furthermore, the consensus algorithms' block creation process tends to centralize the system, contradicting the decentralized nature of blockchain technology. The proposed algorithm addresses these issues by separating the transaction confirmation and block generation processes, reducing confirmation latency. The block creation is decentralized, and a smart contract is employed for candidate selection, minimizing heartbeat overhead. Transaction confirmations are facilitated by the current leader RSU node, leading to reduced confirmation times.

**Conclusion:**

The paper introduces a blockchain-based data security solution for vehicular networks, highlighting the transformative potential of integrating blockchain technology into the transportation system. The integration opens the door to various applications built on top of blockchain. However, the paper notes a tradeoff between security and timely message dissemination in vehicular networks when utilizing blockchain.

The level of security in the blockchain network is influenced by the consensus mechanism, which, if based on extensive miner resources such as time, computational power, storage, and coins, enhances security. However, this heightened security comes at the cost of decreased throughput. The paper also acknowledges that an increase in the number of transactions in a block can lead to delays in block creation and propagation time. Interestingly, the study observes a positive correlation between the number of transactions generated by vehicles and the number of confirmed transactions.

**Reference:** Sehar, N.u., Khalid, O., Khan, I.A. et al. Blockchain enabled data security in vehicular networks. Sci Rep 13, 4412 (2023). <https://doi.org/10.1038/s41598-023-31442-w>

* 1. Intrusion detection techniques: