Received 21 May 2022, accepted 24 June 2022, date of publication 4 July 2022, date of current version 11 July 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3188296

RESEARCH ARTICLE

Impact of Block Data Components on the Performance of Blockchain-Based VANET Implemented on Hyperledger Fabric

PRIYANKA GABA[1], RAM SHRINGAR RAW[2], (Member, IEEE), MAZIN ABED MOHAMMED[3], JAN NEDOMA[4], (Senior Member, IEEE), AND RADEK MARTINEK[5], (Senior Member, IEEE)

[1]University School of Information, Communication and Technology, Guru Gobind Singh Indraprastha University, Dwarka, New Delhi 110078, India
[2]Department of Computer Science and Engineering, Netaji Subhas University of Technology, Delhi 110031, India
[3]College of Computer Science and Information Technology, University of Anbar, Anbar 31001, Iraq
[4]Department of Telecommunications, Faculty of Electrical Engineering and Computer Science, VSB-Technical University of Ostrava, 70800 Ostrava, Czech Republic
[5]Department of Cybernetics and Biomedical Engineering, Faculty of Electrical Engineering and Computer Science, VSB-Technical University of Ostrava, 70800 Ostrava, Czech Republic

Corresponding author: Jan Nedoma (jan.nedoma@vsb.cz)

This work was supported in part by the Ministry of Education of the Czech Republic under Project SP2022/18 and Project SP2022/34; and in part by the European Regional Development Fund in the Research Centre of Advanced Mechatronic Systems Project within the Operational Programme Research, Development, and Education, under Project CZ.02.1.01/0.0/0.0/16019/0000867.

ABSTRACT Blockchain, a vital technology in today's era, changed the way we share, manage and exchange our data in a centralized way to decentralized architecture. With the increasing demand for Blockchain, various platforms are available to implement public, private, consortium, or permissioned, permissionless Blockchain. Hyperledger, an open-source, permissioned, distributed ledger-based Blockchain, was hosted by Linux. This paper explores Hyperledger Fabric Private Blockchain Network (HFPBN). The architecture of HFPBN with its components and transaction flow is explored in detail. The Blockchain in HFPBN comprises multiple blocks that are linked to each other. The block elements are discussed in detail with their type and size, and after that, the total size of the block depending upon various parameters is calculated. Further, one application of Blockchain, i.e., Vehicular Ad-hoc Networks (VANETs), is discussed in this paper as a case study. The VANET application is implemented on the Hyperledger Fabric platform. Formulas showing the dependency of various parameters like endorsement policy, number of transactions, and number of reads and writes on block size are derived and shown in their relationship through the graph for the VANET system. The impact of block size on various performance parameters like throughput, latency, memory, and CPU utilization for the VANET system is then analyzed using Hyperledger Caliper. An optimal required value of throughput and latency is achieved for Blockchain-based VANET. Also, the Hyperledger Fabric platform seems suitable for many applications as it creates separate Blockchain for different applications.

INDEX TERMS Blockchain, Hyperledger Caliper, Hyperledger fabric, VANET.

I. INTRODUCTION

The interminable evolutions of new technologies in the market for satisfying the world's current need impose a secure, reliable system to store huge amounts of generated data. A centralized system proves inappropriate to satisfy the market's needs. Blockchain, a distributed decentralized system, offers many advantages over a centralized system. Blockchain is implemented using cryptography, peer-to-peer networks, and smart contract [1]. The biggest advantage is that the whole system will not break down if one node breaks down, which happens in the centralized system. Also, the decentralization removes a single node which could act as a sole point of failure for the invader [2]. This system itself subsumes faith in the network, and after that, two unknown parties can indulge in using the Blockchain network. The

The associate editor coordinating the review of this manuscript and approving it for publication was Nitin Gupta.

need for an intermediatory and its cost has also become obsolete because of this trusted network. In Blockchain, multiple transactions are recorded in a single block, and all the blocks are interconnected to make a chain. Therefore, Blockchain presents an immutable feature as a single change in one block changes the entire chain preceding that block and makes that node data different from others [3].

Initially, Blockchain was developed for just bitcoin, a cryptocurrency [4], but in today's era, Blockchain has widespread applications in financial and non-financial industries. Nowadays, there is an increasing interest in understanding the transformative power behind cryptocurrency, and that is Blockchain. Blockchain has become the feasible solution that is revolutionizing emerging technologies like IoT, Cloud, Big data, and many more. Blockchain also fits into various applications like healthcare, supply chain, agriculture, VANET, etc.

VANET is one of the prominent applications of Blockchain. It uses Dedicated Short-Range Communication (DSRC) signals at 5.9 GHz frequency to communicate with other vehicles and RSU [5]. In VANET, if a vehicle uses vehicle to infrastructure (V2I) and vehicle to pedestrian (V2P) communications along with vehicle to vehicle (V2V) communication, it is known as the connected vehicle. Rapid change and advancement in VANET come with enhanced communications between vehicles, RSU, hot spots, and other applications. The problem with previous security systems like cryptography and hashing alone used with VANET, doesn't seem suitable for the problem they face. Although Blockchain uses both cryptography and hashing with other features like smart contracts and immutable makes the VANET system more strong and secure [1].

Despite various advantages, Blockchain also suffers from key challenges like security, scalability, power consumption, and performance issues that need significant solutions [3]. To deal with such challenges, various researchers are doing significant work and making Blockchain adaptable for applications [6], [7]. To eliminate performance issues, we first need to understand its dependency on various parameters. This paper explores that block size is a crucial factor that impacts performance. To understand block size in detail for the VANET system, all the data elements of the block are explored and figured out the dependency of the various variable on block size.

The research challenges addressed in this paper are:

- The conventional VANET is implemented on Blockchain to make the network more secure and reliable.
- The detail of all the block elements in Blockchain is discussed in detail. The aim is to analyze the appropriate size of each element for a particular application that is implemented on Blockchain.
- The dependency of size of a particular block in Blockchain-based VANET is analyzed. Block size depends on various block parameters like endorsement policy, number of transactions, and number of reads and writes in a single transaction.
- Further, the dependency of performance parameters like throughput, latency, memory, and CPU utilization on block size is evaluated. The intent is to decide the appropriate block size which gives the best possible performance.

The paper's organization is as follows: Section II discusses the related work. Section III presents an outline of Hyperledger fabric framework architecture, its components, and transaction flow. Section IV discusses the block structure of HFPBN which explores block's data elements, types, and sizes. The equation for the block size is derived depending on various data elements. Section V explores a case study of VANET implemented on Hyperledger Fabric and its architecture. In section VI, the impact of block data elements like endorsement policy, number of transactions, number of reads and writes in a single transaction on the block size of VANET is discussed. The impact of the block size of the VANET scenario on various performance parameters like throughput, latency, memory, and CPU utilization is presented in section VII along with its comparison with other approaches. The paper is concluded in section VIII. The notations used in the paper are summarized in Table 1.

TABLE 1. Summary of notations.

Notation	Meaning
CA	Certification Authority
MSP	Membership Service Provider
OSN	Ordering Service Node
P_n	Any Peer Node
Org-n	Any organization
B_s	Block Size
BH	Block Header
BD	Block Data
BMD	Block Metadata
GB	Genesis Block
Tx	Transaction
TP	Transaction Proposal
PR	Proposal Response
T_xH	Transaction header
T_xD	Transaction data
nT_x	Number of transactions
CH	Channel
CR	Creator
TP	Transaction proposal
E_s	Endorsements
PR	Proposal Response
nArgs	Number of arguments
nEnd	Number of endorsers
nRd	Number of reads
nDV	Number of data values
nWr	Number of writes
TAR	Transaction Arrival Rate
Tps	Transaction per second
Str	String

II. RELATED WORK

Blockchain is an invention to solve the problems behind the risk of hackers involved in any online transactions.

Blockchain is the mother of cryptocurrency but now has applications in all possible fields. Blockchain stores the complete past records in the form of transactions and uses mathematics, cryptography algorithms, and distributed consensus algorithms to solve problems of centralized servers. The public, private, and consortium are types of Blockchain. Further, these types could be permissioned or permissionless [1].

From the system design's point of view, the Blockchain network is implemented on four levels. Those four levels are the data and network organization, consensus, global state machine, and application layer. The consensus layer is responsible for maintaining an agreement based on that all the nodes in the network will have a common replica of the ledger. Consensus algorithms are of different types, and different Blockchain network uses different consensus algorithm. The authors examined the impact of consensus protocol from the perspective of network deployers, participants, and network users [8]. Hyperledger, Stellar, and Ripple use Byzantine fault tolerance (BFT) consensus algorithm [9]. BFT has multiple variants like Practical BFT, Simplified BFT, Delegated BFT, and Practical BFT is best among these.

Blockchain also encompasses an important concept called a smart contract, a pre-written self-executing code that contains the application's logic [10]. Smart contract was introduced with Blockchain 2.0, but currently, 4 million smart contracts have been implemented on Ethereum. Blockchain network has applications ranging from small business to competitive technology like IoT and cloud. With widespread usage, the need is to design an efficient, privacy-preserving Blockchain.

Various organizations with different visions have already implemented various frameworks and tools to fit different needs and circumstances to design and deploy a Blockchain network. A comparison of the three most famous distributed ledger technologies, Ethereum, Hyperledger Fabric, and Corda, was presented by authors [11]. Authors have compared these three based on platform, mode of operation, smart contract, consensus algorithm, and currency involved. Corda has a simplified design compared to fabric and is exclusively for financial services. Ethereum is a permissionless generic platform that fits into any application because of its numerous smart contracts. The issue with the Ethereum platform is in its performance, scalability, and privacy. On the other hand, the fabric is a BFT consensus-based permissioned network that can resolve the issues of Ethereum.

Hyperledger fabric performance is evaluated effectively even considering transaction endorsement failure and block timeout by authors [12]. Authors have derived formulas for measuring throughput, transaction rejection probability, and mean transaction response delay. The simulation and numerical results of the authors concluded that throughput increases with the increase in block size. The transaction rejection probability also rises with the rise in the transaction arrival rate. It also depends on block size; for higher block size, the probability is a little less than for a smaller block size. Mean response delay first decreases, but it increases with the transaction arrival rate when it reaches a specific point.

The work of various authors based on Blockchain-based VANET is considered and explored to understand the system and get the scope of improvement in the system. A novel Blockchain for secure message exchange in VANET was proposed by authors [13]. This system used local Blockchain for VANET for a specific country so that the growth of the network can be controlled. The Blockchain will act as a distributed public Ledger in this system to store the past details of the vehicle's trust level sideways with event messages.

Authors in [14] had proposed a Blockchain-based secure data sharing system for IoV using parent and auxiliary Blockchain to store the messages by different entities of different regions. Fair blind signature and threshold secret sharing are applied in this system to maintain privacy. A punish-reward system was applied to inspire the users to participate in the system.

To overcome VANET security issues, authors in [15] have suggested a data security sharing and storage system based on the consortium Blockchain (DSSCB). This digital signature technique uses bilinear pairing for elliptic curves to ensure data transmission reliability and integrity. It is a decentralized, secure, and reliable database that is maintained by the entire network node. A smart contract limits the trigger conditions for chosen nodes while sending and storing data and allocates data coins to vehicles that contribute data in DSSCB.

The sender's identification data are encrypted in VANET to protect privacy. However, a centralized system can decode the identity data using the sender vehicle's private data. As a consequence, attackers frequently target the central server. The authors [16] proposed a Blockchain-based message authentication mechanism for anonymity and decentralization. They introduced a public-private key and MAC for safe authentication. They have also incorporated proof of work (PoW) and Practical Byzantine Fault Tolerance (PBFT) into the proposed authentication method.

Hyperledger fabric platform seems to have better performance and scalability; therefore, we have chosen this platform for implementing Blockchain-based VANET. The performance factors of VANET like throughput, latency, and network utilization are mainly based on block size, which depends on the data kept in the block. Therefore, this paper focuses on analyzing the block data components and their impact on block size and further impact of block size on performance.

III. HYPERLEDGER FABRIC PRIVATE BLOCKCHAIN NETWORK (HFPBN): ARCHITECTURE, COMPONENTS, AND TRANSACTION FLOW

Implementing a private Blockchain network for any particular application can be done on the Hyperledger Fabric platform. Hyperledger fabric founded under Linux is an open-source, private, permissioned, distributed ledger technology-based platform. The modular architecture of HFPBN uses plug-and-play components to deploy a huge variety of applications with much ease. HFPBN was invented to provide security, privacy, confidentiality, scalability, and efficiency [17]. The tools and software required for the complete Blockchain network are provided in a nutshell in Table 2 below. It includes Operating systems, various tools, programming languages, Databases, frameworks, etc., with their appropriate version, and it is required for which part of the Blockchain.

TABLE 2. Tools and software requirements.

Type	Software	Version	Required For
Operating System	Ubuntu	16.04	Required in all machines
Framework	Hyperledger Fabric	1.4	A tool to implement Blockchain
Containerization Tool	Docker	17.03 or later	Required for peers and orderer
Tool	Docker Composer	1.11 or later	Required for peers and orderer
Smart Contract	Node.js	8.10.0 or later	Required for client application
Programming Language	Golang	1.7.1 or later	Required for peers
Endorsement/ Consensus Model	Kafka/ Solo/Raft	-	Required for ordering service node
World State DB	CouchDB/ LevelDB	-	Required for ledger
Performance Tool	Hyperledger Caliper	V0.2 or next	Required for analysis
Software/ Language	HTML5 Jquery Golang Web Browser	-	Required for client application machines
	Npm	5.6.0 or later	Required for client application

A. HFPBN ARCHITECTURE AND ITS COMPONENTS

The HFPBN consists of multiple components connected in a defined format. Each component is responsible for a unique role in the network to create a Blockchain. Components are connected to transfer some data related to transactions or blocks. Multiple entities of the same component in the network could exist depending upon our requirement. Therefore, we need to customize and configure the network before starting it [18]. Various components with their roles are discussed below.

1) CHANNEL

Channel is a secluded subset of the network consisting of members like peers, ordering service nodes, and applications. Channel allows its members to communicate with each other, and thus they can help achieve an identical copy of the associated ledger of the channel. When a peer joins a channel, the channel's policy can use the peer's identity to extract the peer's rights.

2) ORGANIZATION

Several organizations collectively manage the Blockchain network. Each organization will have peers as its members. Any number of peers from multiple organizations can join a single channel. Apart from peers, the organization also has a client application that is designed for that particular organization. The client application is not part of the Blockchain network. Different members of an organization with their roles are explained below.

a: PEERS

Peers are the essential elements of the Blockchain network. Peers can interact with each other using the gossip data dissemination protocol. Peer node encompasses smart contracts, along with their copy of the ledger. Peers are of two types: endorsing peers and committing peers. Endorsing peers are responsible for executing, endorsing the transaction, and forwarding that transaction to the ordering service node. Committing peers to get the block from the ordering service nodes; validates the transaction and commits the block to the ledger. Endorsing peers also acts as committing peers. Peers encompasses a replica of the smart contract and a copy of the updated ledger [19].

i: SMARTCONTRACT

A smart contract is a programmatic code that defines the rules between different organizations to manage access and alterations to the key values in the form of transactions. These smart contracts are packaged into a chaincode, which is then installed on peers and used by the network's channels. The different types of chaincode are application and system chaincode. System chaincode runs as part of the peer process, and application chaincode, also known as user chaincode, runs in docker containers. Different types of system chaincodes are Lifecycle System Chaincode (LSCC), Configuration System Chaincode (CSCC), Query System Chaincode (QSCC), Endorsement System Chaincode (ESCC), and Validation System Chaincode (VSCC).

ii: LEDGER

Ledger embraces evidence about business objects' current and past states. Ledger is used for storing significant realistic data related to the application. It contains the current state and the history of transactions that lead to it. The ledger consists of a World state and a complete Blockchain. The World state depicts the current state of the ledger of Hyperledger fabric, and the complete Blockchain contains the history of transactions that leads to the current state. In the next update command, this current state is picked to get the latest value of the keys, and after updating the ledger, this current state is also changed for the next read or write command. The history of data is immutable means we cannot change the already added transactions; only we can add new ones.

b: CLIENT APPLICATION

A client is an end-user of the network who does not have any Blockchain data stored in itself. A client, rather through an application, sends the request of query or update to multiple peers of the same channel [4]. The client's request is sent to peers as a transaction proposal request which is further executed and endorsed by peers and sent to orderers for ordering and block creation.

c: CERTIFICATION AUTHORITY (CA)

CA is responsible for generating digital certificates for the network nodes. These certificates carry the complete information about the node and behave like an identity for that node in the network. The peer's certificates are generated by the admin of their organization's CA.

3) ORDERING SERVICE NODE (OSN)

OSN's task is to order the ordering of transactions. The nodes responsible for this are known as orderer or ordering nodes. Orderers impose a constraint on channels, those who can read and write data. Ordering nodes belong to a particular organization and get their identities like other network nodes. The ordering service can be implemented using Solo, Kafka, or Raft to achieve consensus. Solo is used for testing purposes; it contains only one ordering node [20].

4) MEMBERSHIP SERVICE PROVIDER (MSP)

MSP is not accountable for providing certificates to the identities; rather, it keeps a list of permissioned identities and controls it. *MSP* is responsible for mapping identity to a particular organization. It also governs the role of peers in an organization and then consequently will get access to the resources. A separate MSP is associated with each organization, which is connected to the *CA* of that organization.

An example of the arrangement of various components in a Blockchain network is shown in figure 1 with their interconnection. The sample network consists of *2* organizations, each having *two* peers, client application, and *CA*. Each *CA* is associated with a different *MSP* of its organization. A channel connected to peers from different organizations and all peers would be maintaining the same copy of the smart contract and their replica of the ledger. Channel is also inter-linked to *OSN* for ordering the transactions and creating a block from those transactions.

B. TRANSACTION FLOW

A block in a Blockchain comprises transactions and is linked to other blocks. In each block, multiple transactions are grouped together. Any client requests a transaction, and the complete transaction flow consists of three phases: simulate, order-validate & commit. The client, an end-user of the system, creates a transaction proposal to query or write some data on the Blockchain network. The client application is sending the transaction proposal to multiple endorsing peers belonging to the same channel. As shown in figure 2, peer P_2

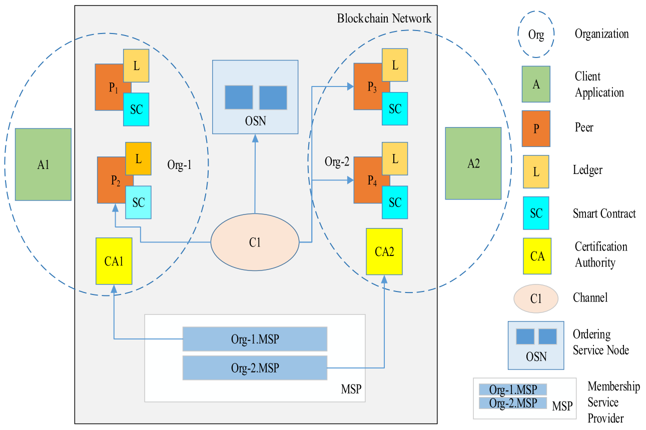
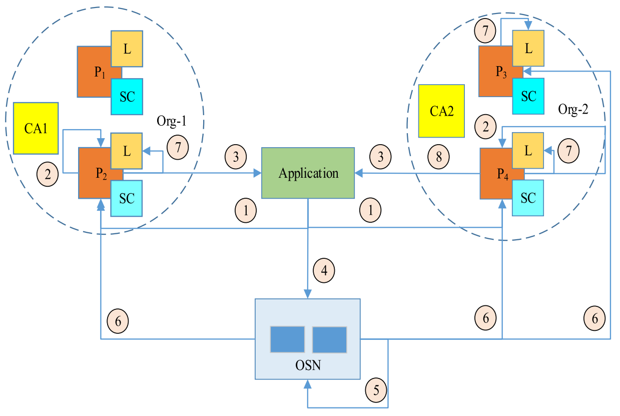


FIGURE 1. Hyperledger fabric private blockchain network components.



1. Application sends transaction proposal request to endorsing peers
2. Peer executes transaction
3. Peer sends endorsement response to application
4. Application sends transaction proposal responses with endorsers signature to OSN
5. OSN apply consensus to create block from transactions
6. OSN delivers new block to all endorsing peers and committing peers
7. Block validation and updating the ledger copy by peers
8. Peers send transaction status to invoking clients.

FIGURE 2. Transaction flow in hyperledger fabric private blockchain network.

of *org-1* and peers P_3 and P_4 of *org-2* belong to the same channel. P_2 and P_4 are endorsing peers, and P_3 is committing peer, so the application sends the transaction proposal request to P_2, P_4 as shown in figure 2, step 1. The endorsing peers P_2 and P_4 have replicas of the ledger and smart contract. In step *2*, the endorsing peers P_2, and $P_{4,}$ simulate the transaction. Using the smart contract, the endorsing peers validate the transactions and inquiry the ledger to get the current state of the ledger. After endorsement, the endorsing peers P_2 and P_4 send the signed endorsement result back to the client [21] as part of step *3*.

The client application will get endorsement responses from multiple endorsing peers. Once the required responses are received, the client application collects all responses and forwards them to the *OSN* [22] in step *4*. *OSN* then uses the consensus to reach the ordering of the transaction in step *5*. The transactions are then bundled together to create a block.

The size of the block is decided based on either the maximum number of transactions, or maximum size of the block, or the maximum time elapsed in the creation of the new block. In step *6*, the new block is broadcasted to all endorsing peers P_2, P_4, and committing peer P_3 of the same channel. All the peers will now validate the new block, as shown in step *7*. After validation, the transactions are committed to the ledger. The clients corresponding to transaction proposal requests are informed about the validation of the transactions as in step *8*. The complete process of transaction flow in HFPBN is carried out in 8 steps, as shown in figure 2.

IV. BLOCK STRUCTURE OF HFPBN

A Blockchain is organized as an arrangement of connected blocks. Each block is connected to the preceding block and hence makes a chain. Once a block is formed and appended to the Blockchain, the data in that block can't be transformed, which makes the Blockchain immutable [23]. Each block consists of multiple transactions, where each transaction of a block signifies either a query or modifies the world state of the ledger. The Blockchain starts with block 0, called Genesis Block (GB), which doesn't contain any transaction. The transactions are recorded from Block 1 onwards. Each block of data comprises three parts: Block Header (BH), Block Data (BD), and Block Metadata (BMD) [24], as shown in figure 3.

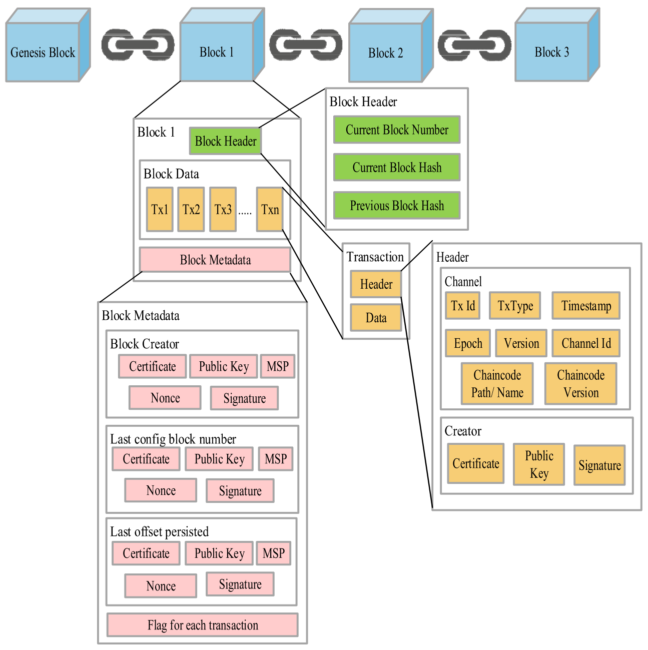


FIGURE 3. Block data components.

BH of each block includes the current block number (8 bytes of uint64 type), current block hash (*32* Bytes str), and previous block hash (*32* Bytes str). The total size of *BH* is *72* bytes, as shown in Table 3 with elements, their type, and size. The current block number is of type integer that initializes with 0 for Genesis Block and increments with 1 for every next block. The current block hash is the SHA256 hash generated from all the transactions of the present block. The preceding block hash is the SHA256 hash of the preceding block to connect this block with the previous one and make a chain.

TABLE 3. Block → header (72 bytes).

Category	Name	Type	Size
Block → Header (72 Bytes)	Current Block Number	uint64	8 Bytes
	Current Block Hash	Str	32 Bytes
	Previous Block Hash	Str	32 Bytes

The BMD section consists of multiple components but is not used to generate the block hash. The 1st component contains the details related to the block creator, which other network nodes could use to verify the block. The 2nd component encompasses the last configuration block number. The 3rd component includes the last offset persisted. These three components contain the creator identity field, nonce, and signature (256 bytes str) field. The creator field contains x.509 certificates (*256* bytes str), public key (*256* bytes str), and *MSPid* (*4* bytes) which dispense these identities to the client. The Nonce field holds any random value for generating a unique hash (*4* bytes). The 4th component contains the Flags for every transaction (byte array), indicating the transaction's validity. The validation of transactions is based on multiple factors like endorsement policy, concurrency violations, verification of read-write set, etc. The ordering service adds the first three components, whereas the block committer adds the 4th component. The total size of BMD is $(2328 + nTx)$ bytes as shown in Table 4 with elements, its type, and size. Here, nTx represents the number of transactions in the block.

TABLE 4. Block → MetaData (2328 + *nTx*).

Name			Type	Size
Block Creator (776 Bytes)	Creator Identity	Certificate	str	256 Bytes
		Public Key	str	256 Bytes
		MSP	str	4 Bytes
	Nonce		int32	4 Bytes
	Signature		str	256 Bytes
Last config Block Number (776 Bytes)	Creator Identity	Certificate	str	256 Bytes
		Public Key	str	256 Bytes
		MSP	str	4 Bytes
	Nonce		int32	4 Bytes
	Signature		str	256 Bytes
Last Offset Persisted (776 Bytes)	Creator Identity	Certificate	str	256 Bytes
		Public Key	str	256 Bytes
		MSP	str	4 Bytes
	Nonce		int32	4 Bytes
	Signature		str	256 Bytes
Flag for each Transaction (Multiple transactions would be there, so 1*nTx)			bool	1 byte

BD section contains an ordered list of transactions. In each transaction, there are two fields: Transaction Header (T_xH)

and Transaction data (*TxD*). The T_xH field contains the information related to the channel. The data field comprises transaction Proposal (*TP*), Endorsement (E_s), and Proposal Response (*PR*), as shown in figure 4. These fields correspond to three steps of transaction flow: simulate, order-validate & commit model, as discussed in the above section.

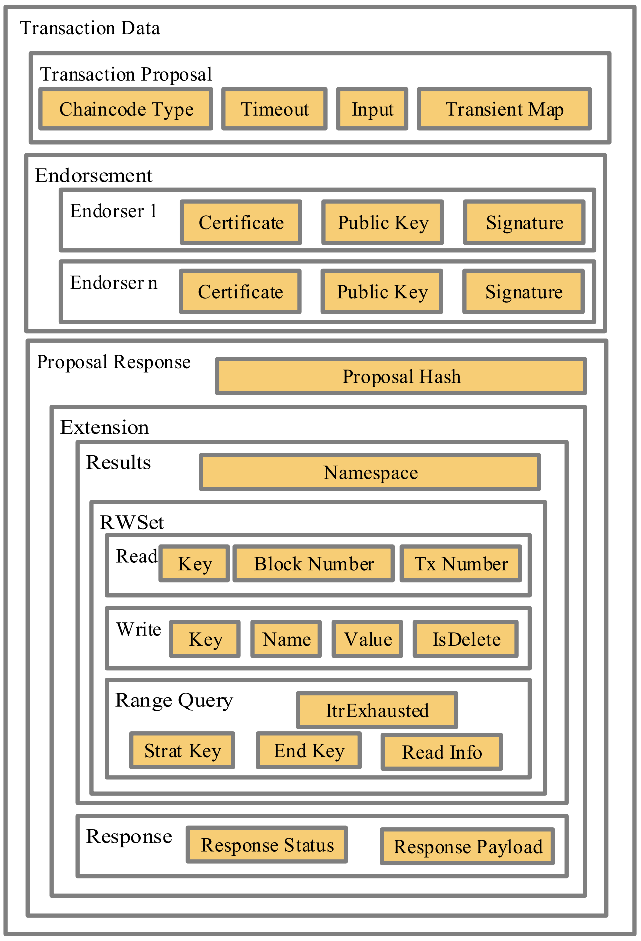


FIGURE 4. **Transaction data.**

T_xH consists of Channel Header (*CH*) and Creator's (CR) details. *CH* contains eight fields to describe the basic details of the transaction are as follows. Every transaction is stored with a unique Transaction Id (string type), a hexadecimal code generated to identify each transaction uniquely. A Transaction type field (int32 type) is a number encoded earlier by the system. The transaction types are message, configuration, configuration update, endorser, orderer, etc. A timestamp (time type) field is used to store the time at which the *TP* is created. The channel id field (string type) stores the channel in which the transaction is proposed. The Version field (int32 type) serializes structured data to denote the protocol buffer. The Epoch field is of type uint64 and is set to zero if not used currently. The last field in the channel header is Payload Visibility, which signifies the chaincode payload visibility. Its value can be set for full visibility, partial, or nothing. Another part of the chaincode header contains details of the chaincode that will execute the transaction. Either the path or name (string type) of the involved chaincode is specified. For invoke type of transactions, the name of chaincode is required, but for a deploy transaction, chaincode path is required. Apart from that chaincode version (string type) is also provided. All these details of *CH* consume 40 bytes of memory space.

The *TP* is signed using the Creator (*CR*) client's credentials which are stored as the transaction's client identity containing its certificate (*256* bytes), public key (*256* bytes str), and signature (*256* bytes str). The total size of T_xH of *BD* is (40 + 768) bytes, as shown in Table 5 with elements, their type, and size.

TABLE 5. **Block → block data → transaction header.**

Category	Name	Type	Size
Block→ Block Data → Transaction Header→ Channel (40 Bytes)	Transaction Id	str	4 Bytes
	Transaction Type	int32	4 Bytes
	Timestamp	time	8 Bytes
	Channel Id	str	4 Bytes
	Version	int32	4 Bytes
	Epoch	uint64	8 Bytes
	Chaincode path /Chaincode name	str	4 Bytes
	Chaincode Version	str	4 Bytes
Block → Block Data → Transaction Header → Creator (768 Bytes)	Certificate	str	256 Bytes
	Public Key	str	256 Bytes
	Signature	str	256 Bytes

Apart from the details mentioned in the header part related to chaincode path and name, other details are also required those are specified in the *TP* section. Those details are chaincode type (int32 type), input parameters that are passed to the chaincode function at the time of calling, and timeout (int32 type). The *TP* contains a payload that contains the action corresponding to a transaction. Presently, the fabric can only carry one action corresponding to one transaction. To specify transactions in payload, it encompasses two components. One is related to the proposal, and the other is action. The proposal field contains input and transient maps. Transient Map (string type) contains secret information that can be passed to chaincode, but it is not stored in the BD for privacy. Input (string type) specifies chaincode invocation; it contains chaincode details specified in the *TP* part. The action part of the payload contains components to store data related to E_s and *PR*.

TP is sent to a single or extra endorsing peers for transaction simulation, execution, and endorsement [25]. The Endorsers' identity details like certificate (*256* bytes str) and public key (*256* bytes str) are stored along with their signature (*256* bytes str) in the E_s section. The total size of *TP* and E_s of T_xD of BD is $(12 + 4 * nArgs) + (768 * nEnd)$ bytes as shown

TABLE 6. Block → block data → transaction data.

Category	Name	Type	Size
Block → Block Data → Transaction Data → Proposal (12+ 4*nArgs) Bytes	Chaincode Type	int32	Bytes
	Timeout	int32	Bytes
	Input Parameters (could be multiple 4* nArgs)	str	Bytes
	Transient Map	str	Bytes
Block → Block Data → Transaction Data → Endorsement (Multiple Endorsers are involved so 768 * nEnd Bytes)	Certificate	str	56 Bytes
	Public Key	str	56 Bytes
	Signature	str	56 Bytes

in Table 6 with elements, their type, and size. Here *nArgs* is the number of arguments given in input.

The *PR* is the third part of the transaction data. It consists of two parts: Proposal Hash, and Extension. Proposal Hash (string type) is the hash of the proposal. The extension part contains two components: Results, and Response. The results section carries the transaction read/write set data. It contains two fields: Namespace and RWSet. Namespace (string type) contains the name of the system or application chaincode. RWSet carries three data types: Read, Write, and Range query [26]. The read set contains the information required for reading transactions, and that is key (string type), Block Number (uint64 type), and Transaction Number (uint64 type). Write set contains information required for writing which involves: key (string type), value (variable type), and IsDelete (bool type). Multiple writes for different keys can be done in the same transaction. The value field contains a pair of names and the value of those changed in this transaction. Its size depends on the values required to change in the transaction for a particular key. The name-value fields are of type string. IsDelete, a Boolean type variable, is to check the status. Range query set contains information like start key (string type), End Key (string type), ItrExhausted (bool type), and ReadsInfo (bool type). The start and end keys are to keep track of multiple keys to be read. ItrExhausted is just to see when to end the multiple read operations. ReadsInfo is a variable to tell whether to perform a read or write operation. The response section contains Response status, of type int32, and Response payload, of type string. The total size of *PR* of *BMD* is $(26 + 20 * nRd + (5 + (8 * nDV) * nWr)$ bytes as shown in Table 7 with elements, their type, and size. Here, *nRd* represents the number of reads, *nDV* is the number of data values, and *nWr* is the number of writes.

The responses from the endorsing peers are gathered and bundled in the transaction and submitted to the ordering service. The responsibility of the ordering service is to order and put the required number of transactions in a block. The newly created block is then broadcasted to all the peers who will validate the transactions. The transactions are then committed to the ledger, which will update the world state.

The total size consumed by a block represented by Bs depends on all fields, as shown in Table 3 to Table 7. Typically, it is based on three major components Block Header, Block Data, and Block Metadata, as shown in equation (1). The size of *BH* as calculated above is 72 bytes. *BD* is based on the Transaction Header (T_xH) and Transaction Data (T_xD) part, as shown in equation (2). T_xH part consists of Channel (*CH*) and creator (*CR*), as shown in equation (3). *CH* component is consuming 40 bytes. *CR* component is consuming 768 bytes. T_xD consists of Transaction proposal (*TP*), Endorsements (E_s), and Proposal Response (*PR*) and is calculated in equation (4). *TP* is consuming $(12 + 4 * nArgs)$ Bytes. E_s is dependent on the number of endorsers, represented by *nEnd*. The size consumed by the *Es* field is $(768 * nEnd)$ Bytes. *PR* consumes$(26 + 20 * nRd + (5 + (8 * nDV) * nWr)$Bytes. *BMD* field of block consumes $(2328 + nTx)$.

By putting all values to equation (1), total $B_s$are calculated and given in equation (7).

$$B_s = BH + (BD) * nTx + BMD \tag{1}$$

where

$$BD = TxH + TxD \tag{2}$$

And

$$TxH = CH + CR \tag{3}$$

Similarly,

$$TxH = 40 + 768 = 808$$

and

$$TxD = TP + Es + PR$$

Or

$$TxD = 12 + 4 * nArgs + 768 * nEnd + 26 + 20 * nRd + (5 + (8 * nDV) * nWr)$$

Therefore,

$$TxD = 38 + 4 * nArgs + 768 * nEnd + 20 * nRd + (5 + (8 * nDV) * nWr) \tag{4}$$

Putting the values of *TxH* and *TxD* calculated in (3) and (4) to (2) can be represented as:

$$BD = 808 + 38 + 4 * nArgs + 768 * nEnd + 20 * n + (5 + (8 * nDV)) * nWr$$

$$BD = 846 + 4 * nArgs + 768 * nEnd + 20 * nRd + (5 + (8 * nDV)) * nWr \tag{5}$$

and

$$BMD = 2328 + nTx \tag{6}$$

TABLE 7. Block → block data → transaction data → proposal response.

Category	Name						Type	Size
Block → Block Data → Transaction Data → Proposal Response (26+ 20 * nRd + (5+(8* nDV)*nWr)) Bytes	Proposal hash						str	4 Bytes
	Extension		Namespace				str	4 Bytes
		Results	RWSet	Read (Multiple read so 20 * nRd)	Key		str	4 Bytes
					Block Number		uint64	8 Bytes
					Transaction Number		uint64	8 Bytes
				Write (Multiple write so (4+(8* nDV)+1)*nWr)	Key		str	4 Bytes
					Data Value Multiple Data values for a single key (8* nDV)	Name	str	4 Bytes
						Value	str	4 Bytes
					IsDelete		bool	1 Byte
				Range Query	Start Key		str	4 Bytes
					End Key		str	4 Bytes
					ItrExhausted		bool	1 Byte
					Read Info		bool	1 Byte
		Response	Response Status				int32	4 Bytes
			Response Payload				str	4 Bytes

Putting values of *BH* as 72, *BD* and *BMD* calculated in equations (5) and (6) to equation (1) can be represented as:

$$Bs = 72 + (846 + 4 * nArgs + 768 * nEnd + 20 * nRd \\ + (5 + (8 * nDV)) * nWr) * nTx + 2328 + nTx$$

$$Bs = 2400 + nTx + (846 + 4 * nArgs + 768 * nEnd \\ + 20 * nRd + (5 + (8 * nDV)) * nWr) * nTx \quad (7)$$

V. BLOCKCHAIN IMPLEMENTATION ON VANET USING HYPERLEDGER FABRIC

Hyperledger Fabric, a platform for private Blockchain implementation, seems suitable for various applications with modular architecture. The versatile design and modular architecture of fabric satisfy the market's needs in today's era. The examples where organizations could develop their applications using Blockchain to improvise business are Supply chain management, healthcare, real estate, financial sector, VANET, etc.

Vehicular Ad-hoc Network (VANET) allows vehicles on the road to interconnect with each other and with RSU (Roadside Unit) to have a safe journey [27], [28]. With the rise in the number of vehicles and technological advancements, it is important to guarantee the safety and reliability of the network [29]. Earlier, the data on VANET was stored on a centralized server, which acts as a hub for all nodes to communicate with each other. The issues of attack with centralized node create problems to the stored VANET's data, which could be tackled well by Blockchain. Blockchain, a distributed, immutable, peer-to-peer network, allows VANET data to be stored in a decentralized fashion to make it more secure [30]. The Hyperledger Fabric, a private permissioned platform of Blockchain, fits perfectly for many vehicles in dynamic VANET. The HFPBN creates different Blockchains for different applications, and even different Blockchains could be created for a single application. VANET and its applications are huge networks, and traffic data of a country is of no relevance to other countries; therefore, creating a single Blockchain of VANET on the Ethereum platform is not a preferable option. Hyperledger, on the other hand, allows VANET to create a separate Blockchain for different countries. Performance factors also give better results with this option, ensuring low latency and high throughput. The architecture of Blockchain-based VANET using Hyperledger fabric will connect various vehicles and RSU to the client application, allowing them to communicate with the Blockchain network.

The architecture of Blockchain-based VANET is shown in figure 5. It comprises three layers: the VANET, application, and Blockchain layer. The VANET layer consists of vehicles and RSU. The application layer consists of a client application connecting vehicles with the Blockchain network. The last layer is the Blockchain Layer, which consists of all the components required to construct the Blockchain network like peers, organizations, smart contracts, ledger, and OSN.

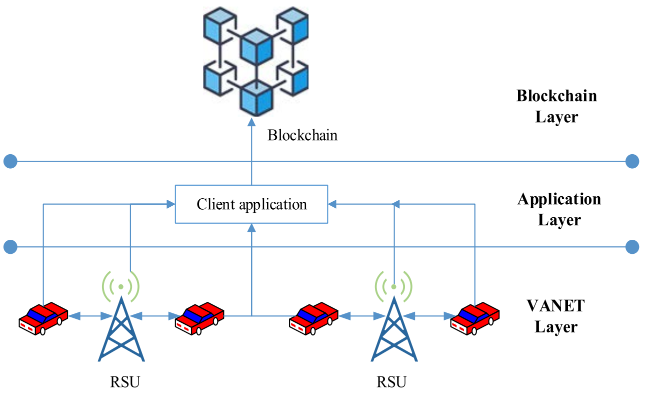


FIGURE 5. Blockchain-based VANET.

Vehicles and RSU of the VANET layer communicate with each other for safety purpose or non-safety reasons. That communication will act as a transaction of the Blockchain. Apart from that, the block's transactions for VANET could store data related to vehicle information, messages transmitted with each other, routing details, road situations, traffic data, accident data, insurance-related data, etc. [31], [32]. In the autonomous vehicle case, much more data needs to be stored as transactions on a Blockchain network like object recognition data, location tracking, movement prediction data, vehicle driving data like speed, source, destination, break, etc. [33], [34]. Vehicles also act as endorsement peers or committing peers in the network to endorse that transaction or to commit the block on the ledger. The vehicles and RSU use the developed application for VANET to initiate the transactions, which are further forwarded to the Blockchain layer for verification and adding them to the block. The initiated transaction by the VANET layer follows the transaction flow in the Blockchain layer. The transaction flow involves multiple phases. Phase 1 is transaction creation by vehicles. Then the created transactions are validated by endorsement peers in phase 2. Phase 3 involves first ordering transactions by ordering service node, and then a block is created from multiple transactions. That new block is broadcasted to all the involved nodes in the network in phase 4. Phase 5 involves validation of that block by the network peers, and after validation, the new block is committed to the ledger by the committing peers. In Phase 6, the peers inform about the status of the transactions to the initiated vehicles.

The block involved in the Blockchain is a crucial factor of the complete Blockchain system and has a great impact on the system's performance. To analyze the performance of Blockchain-based VANET block size is evaluated and analyzed. As discussed above, the block size is dependent on various factors like the number of transactions, endorsement policy, and the number of reads and writes. These are applicable for Blockchain-based VANET; therefore, it's crucial to decide these factors as block size impacts the performance factors. The next sections will discuss in detail the Blockchain-based VANET and its impact on performance factors.

VI. IMPACT OF DIFFERENT BLOCK PARAMETERS ON ITS SIZE FOR BLOCKCHAIN-BASED VANET

A complete block in a Blockchain consists of multiple transactions. Every block consists of some common data related to that block and previous block and individual transaction details of that block. As given in equation (7), B_S is dependent on multiple parameters like how many transactions are stored in one single block, the Endorsement policy, and the number of reads and writes in a single transaction. All these parameters have a different level of impact on B_S. In our proposed application implemented on Blockchain, the information related to vehicles, and their inter-communication regarding the road situation is also stored as a transaction of the block. The blocks are chained together to create a Blockchain for VANET. All the above-mentioned parameters are, therefore, crucially related to the block size of the proposed VANET block. To check the impact of each parameter on $B_S$for the VANET application, all other parameters are considered constant. The equation used for the analysis is given in (7).

A. IMPACT OF ENDORSEMENT POLICY ON BLOCK SIZE

In the Blockchain-based VANET example, there could be multiple instances like the transferred data between vehicles, validating a vehicle node, etc. which could act as a transaction of the block in the Blockchain. All such transactions before adding to a block need to be validated first. For validation, the client forwards the transaction proposal to multiple endorsing peers. The endorsement policy of the VANET system is already decided earlier at the time of constructing the system, which specifies the minimum required number of peers who endorses the transaction proposal. The transaction is validated only when received endorsement results from the required number of peers. It is checked at the time of ordering by the *OSN* [25]. The details like certificate, public key, and signature of all the involved endorsing peers are taken with the transactions for future verification. It implies that the B_S will increase with the rise in the number of required endorsing peers to validate a transaction. To analyze the impact of endorsement policy on B_S, all other variables are considered constant in the above equation (7). The equation (8) below depicts the relationship between B_S and the number of peers required in the endorsement policy. The value of B_S is measured in KB with the varying number of peers in the endorsement policy. The graph in figure 6 shows the B_S is increasing rapidly with the rise in the number of peers even though the number of transactions is considered fixed in each case, which is *10*. Therefore, choosing a suitable number of required peers in endorsement policy is crucial for the B_S.

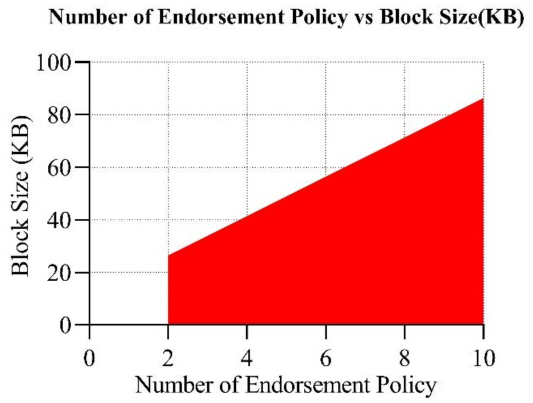


FIGURE 6. Number of endorsement policy vs. block size (KB).

Considering all other factors as constant except *nEnd* and therefore,

$$nTx = 10, nArgs = 2, nRd = 1, nDV = 1, nWr = 1$$

$$\begin{aligned} Bs &= 2400 + 10 + (846 + 4*2 + 768*nEnd \\ &\quad + 20*1 + (5+8)*1)*10 \\ Bs &= 2410 + (846 + 8 + 768*nEnd + 20 + 13)*10 \\ Bs &= 2410 + (846 + 8 + 768*nEnd + 20 + 13)*10 \\ Bs &= 2410 + 8870 + 7680*nEnd \end{aligned}$$

Finally,

$$Bs = 11280 + 7680*nEnd \tag{8}$$

B. IMPACT OF NUMBER OF TRANSACTIONS ON BLOCK SIZE

The count of transactions in a block cannot be assumed constant therefore, the size of each block may vary. The batch size is set to resolve this issue, which contains three parameters that are set keeping the application's needs in mind. These parameters are Max message count, Absolute max bytes, and Preferred max bytes [35]. Max message count is to specify the maximum allowed count of transactions in a block. The absolute max bytes value is to specify the maximum allowed size of a block. Preferred max bytes denote the ideal size of a block, but if a single transaction exceeds this value and is less than Absolute max bytes, it can create a block from that single transaction. To evaluate the impact of just the count of transactions in the VANET system on block size, let's assume the size of each transaction is same. Other parameters like number of peers in endorsement policy, number of reads, and writes are considered constant as specified below and put in the above equation (7) of B_s. The equation (9) below shows the association between B_s and the number of transactions. The variation in B_s by varying the number of transactions in a block is analyzed after assuming that the B_s are still less than the absolute max bytes value. The graph in figure 7 shows that with the rise in the number of transactions from *10* to *50,* the B_s reached *26.02* to *120.7.* The graph implies its impact, which must be considered when choosing the Max message count. Although if in case the number of transactions value is assumed high still by an appropriate value of the other two parameters, preferred maximum bytes and absolute max bytes can control the size of the block.

Considering all other factors as constant except nTx,
Then,

$$\begin{aligned} nEnd &= 2, nArgs = 2, nRd = 1, \\ nDV &= 1, nWr = 1 \\ Bs &= 2400 + nTx + (846 + 4*2 + 768*2 \\ &\quad + 20*1 + (5+8)*1)*nTx \\ Bs &= 2400 + nTx + 2423*nTx \end{aligned} \tag{9}$$

C. IMPACT OF NUMBER OF READS ON BLOCK SIZE

A transaction in the Blockchain for the VANET application is of a query type or update type. In both types, a transaction can ask to read some values from the ledger and might update those values in an update type transaction. Every read needs three parameters: version, block number, and transaction number. The total size required by these parameters is 20 bytes. Every transaction could read multiple values simultaneously, like reading any stored data related to vehicles so that multiple transactions can have many reads. The impact of multiple concurrent reads in a single transaction is not much, but we can still see its relationship with B_s in the equation below. The other parameters like endorsement policy, number of transactions, and number of writes are taken as constant. To see the impact, the number of transactions value has taken *50* in the above equation (7), and the relation between B_s and the number of reads is shown in equation (10) below. The variance in B_s with the rise in the number of reads in a single transaction from *2* to *10* is analyzed. The graph in figure 8 shows that with the rise in the number of reads in a transaction, B_s increases slowly; therefore, the number of reads is not having much impact on block size. It signifies that each transaction of VANET can read multiple values simultaneously in a single transaction, and it will not impact block size much.

Considering all other factors as constant except nRd and as given,

$$nEnd = 2, nArgs = 2, nTx = 50,$$

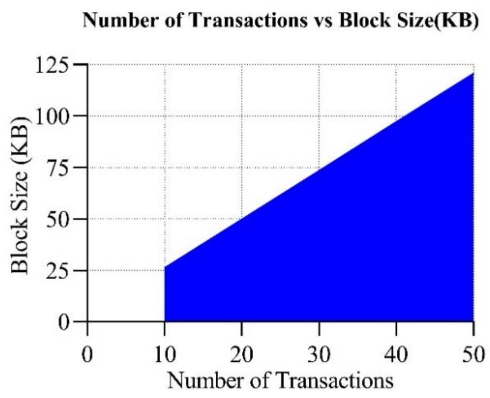


FIGURE 7. Number of transactions vs. block size (KB).

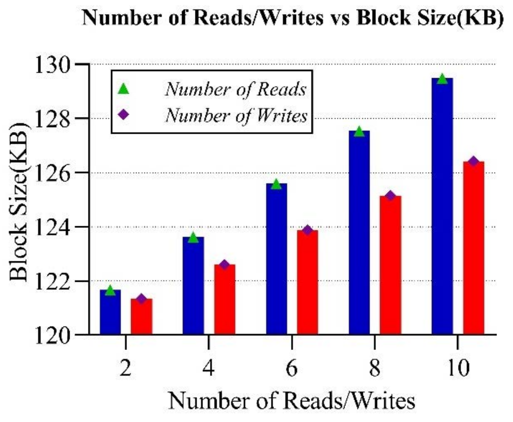


FIGURE 8. Number of reads/writes vs. block size (KB).

$$nDV = 1, nWr = 1$$

Then, B_s further can be determined as:

$$\begin{aligned} Bs &= 2400 + 50 + (846 + 4 * 2 + 768 * 2 \\ &\quad + 20 * nRd + (5 + 8) * 1) * 50 \\ Bs &= 2450 + 120150 + 1000 * nRd \\ Bs &= 122600 + 1000 * nRd \end{aligned} \tag{10}$$

D. IMPACT OF NUMBER OF WRITES ON BLOCK SIZE

The update type of transaction intends to modify the data of the ledger. In the case of VANET, the update type of transaction includes any event data that vehicles are exchanging or any change in the vehicle's details. A single transaction can change multiple keys and multiple data values of the same key at a time. The update transaction cannot alter the current state of the ledger instantly. Instead, it will be done after the transaction proves valid and has been added to the block by the orderer. Other parameters are taken as constant to analyze the impact of multiple key changes on B_s simultaneously. The number of data values updated in a single key is also constant to examine the influence of only the number of writes. The number of transactions value is taken to *50* to see the change. The below equation (11) is calculated after assuming other values as constant in the above equation (7). The graph in figure 8 shows that the B_s is changing slightly with the change in the number of writes. It implies that we can easily change some values in a single transaction by not impacting much of the B_s.

Considering all other factors as constant except nWr, then,

$$\begin{aligned} nEnd &= 2, nArgs = 2, nTx = 50, nRd = 1, nDV = 1 \\ Bs &= 2400 + 50 + (846 + 4 * 2 + 768 * 2 \\ &\quad + 20 * 1 + (5 + 8 * 1) * nWr) * 50 \\ Bs &= 2450 + 120500 + 650 * nWr \\ Bs &= 122950 + 650 * nWr \end{aligned} \tag{11}$$

VII. PERFORMANCE ANALYSIS FOR BLOCKCHAIN-BASED VANET

In this section, various factors that impact various performance parameters are presented. For performance evaluation of Blockchain-based VANET, a sample Blockchain network on Hyperledger Fabric is created. The network consists of a channel, *OSN* with a solo consensus mechanism, two organizations with *two* peers, and *CA*. Individual peer has a replica of the smart contract and ledger.

The endorsement policy with the "*OR*" principle is used herein, which signifies that any single endorser is enough to execute the transaction. The system configuration taken for the simulation PC is Intel (R) Core (TM) i5 -1035G1 CPU@ 1.00 GB 1.19 GHz, 8 GB RAM, 64- bit operating system, 256 GB SSD, 1 TB hard disk, and running on Ubuntu 16.04 LTS. To examine the performance of this network, a benchmarking tool recognized as Hyperledger caliper is used. The complete configuration required for system setup is mentioned in Table 8.

TABLE 8. Setup parameters.

H/W Configuration	• Intel (R) Core (TM) i5 -1035G1 CPU@ 1.00 GB 1.19 GHz • 8 GB RAM • 64- bit operating system • 256 GB SSD • 1 TB hard disk	
S/W Configuration	Operating System	Ubuntu 16.04 LTS
	For Client Application	• HTML5 • Jquery • Golang • Web Browser
	For Blockchain Network	Hyperledger Fabric • Organization-1 ○ MSP ○ CA ○ Peer-1 ▪ Smart contract ▪ Ledger ○ Peer-2 ▪ Smart contract ▪ Ledger • Organization-2 ○ MSP ○ CA ○ Peer-1 ▪ Smart contract ▪ Ledger ○ Peer-2 ▪ Smart contract ▪ Ledger • OSN • Client Applications • Channel • Gossip Protocol • CouchDB
	For Performance Evaluation	Hyperledger Caliper

Caliper runs the Blockchain network and generates HTML reports showing the network's performance. The various performance parameters shown in the reports includes success and failure rate(graph), latency (min, max, avg), throughput, and resource utilization [36].

Throughput is the rate at which all network nodes commit valid transactions in fixed time duration. Throughput is stated in transactions per second (*tps*). Latency is the total time a transaction takes from submitting a transaction proposal request to when it is committed to the ledger and widely available in the network [37]. Resource utilization measures the CPU used, memory consumed, and disk read/write, respectively. All these performance parameters including other factors are also dependent on block size [38]. The Blockchain-based VANET performance parameters are, therefore, also dependent on the block size which as mentioned in section VI is dependent on parameters like how many transactions are stored in one single block, the endorsement policy, and the number of reads and writes in a single transaction. Therefore, it's significant to identify the optimal block size based on the above dependent parameters. Further, the dependency of block size on the performance parameters of blockchain-based VANET is also crucial for obtaining the best possible performance. The values are analyzed for different B_s over different transaction arrival rates to analyze the variance of performance factors.

A. THROUGHPUT

The graph in figure 9 illustrates the overall transaction throughput, which is analyzed for different B_s at different Transaction Arrival Rates (*TAR*) for our VANET application. *TAR* is measured in transactions per second (*tps*). In this analysis, the transaction arrival rate varies from *50 tps* to *250 tps*. B_s is taken from *20* to *100* depending on the number of transactions in the block. Transaction throughput is also based on the endorsement policy, and in this analysis, the "*OR*" principal endorsement policy is chosen just for simplicity. The graph shows *two* observations; one is between *TAR* and throughput, and the other is between B_s and throughput.

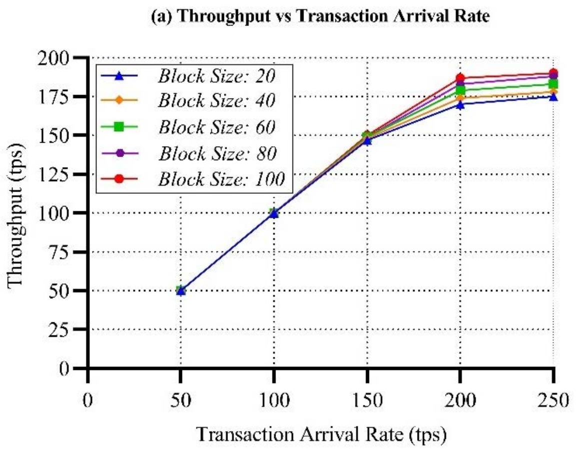


FIGURE 9. Throughput vs. transaction arrival rate.

Throughput does not necessarily increase with the increase in TAR; rather, throughput will become constant at a saturation point of TAR. It is because the capacity of every system to handle multiple transactions at a time is limited. The throughput increases linearly with the increase in TAR until the saturation point of 150; then, the graph is not linear. After 200 tps, the graph gets flat. The second observation for B_s and throughput is that B_s do not impact throughput until saturation. For all block sizes till 150 tps TAR, the throughput is around the same for all B_s at a specific value of TAR. After saturation point, the throughput of higher block size is a little higher than the smaller block size.

B. LATENCY

Latency, another vital performance parameter, is to measure the time taken to complete the transaction of the VANET system. This time is from the point when the transaction is submitted until its result is available in the network. The transaction latency is dependent on all three phases: simulate, order-validate & commit [39]. Latency is analyzed for various B_s from *20* to *100* according to the number of transactions in the block. The value of latency at various B_s is analyzed at different transaction arrival rates from *50 tps* to *250 tps* at an interval of *50*. The latency in seconds is shown on the y-axis in the graph shown in fig 10 gives a few observations. The first observation is that the saturation points of transaction arrival rate is strongly impacted. The saturation rate is around 150; then, latency increases significantly for all B_s. With a greater number of transactions handled by the network, there would be a greater number of transactions in the queue waiting to be validated.

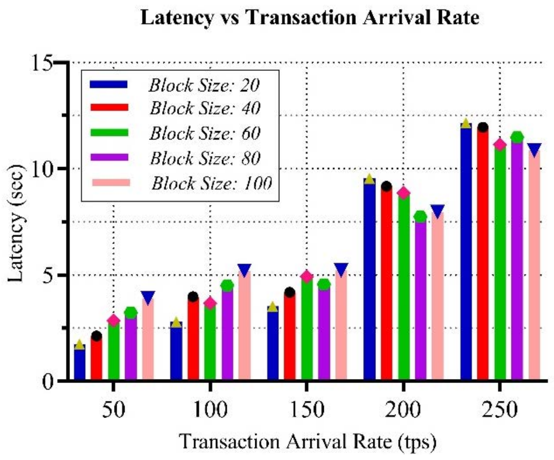


FIGURE 10. Latency vs. transaction arrival rate.

Another observation is that before the saturation point value of *TAR*, with the increase in B_s, latency is increasing for the same value of *TAR*. The minimum latency value at *TAR* of *50 tps* is *1.75* seconds for B_s *20*, and it increased almost *3*-fold till B_s *100*. This increase is because the block creation time at the orderer increases due to an increase in B_s and hence the latency. After saturation point value, latency usually decreases with the increase in B_s. At TAR *250 tps*, the maximum value of latency is *12.16*, which is for B_s *20*, and with the increase in $B_{s,}$ it is decreasing.

C. MEMORY AND CPU UTILIZATION

Figure 11 shows the memory utilization in terms of RAM and CPU utilization with varying B_s for the VANET system. The left side shows the increase in utilized memory in MB with the increase in B_s. The maximum RAM consumption is *98.87* MB which is for *100* B_s. The right side of the graph shows that with the increase in B_s from *20* to *100*, the CPU utilization also increases. The maximum CPU utilization is *27 %* which is for B_s of *100*. However, memory and CPU utilization values completely depends on the test machine.

D. B-VANET COMPARISON WITH OTHER APPROACHES

The performance of the B-VANET approach is compared with some other approaches based on various parameters. The considered parameters are decentralization, non-repudiation and traceability, privacy protection, anonymity, throughput

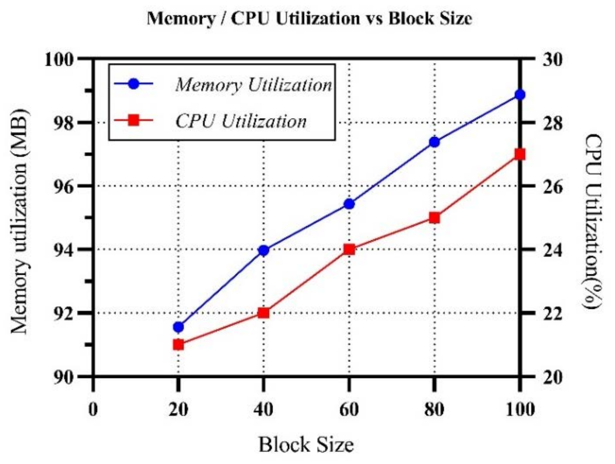


FIGURE 11. Memory / CPU utilization vs. block size.

TABLE 9. Performance comparison of B-VANET with other approaches.

	EAAP	DSSCB	ALICIA	EPAM	B-VANET
Decentralization	×	✓	✓	✓	✓
Non-repudiation and traceable	×	✓	✓	✓	✓
Privacy protection	✓	✓	✓	✓	✓
Anonymity	✓	✓	✓	✓	✓
Throughput analysis	×	×	✓	✓	✓
Latency Analysis	×	×	✓	✓	✓
Dependency on Block Size	×	×	×	✓	✓
Dependency on Endorsement Policy	×	×	×	×	✓

and latency analysis, dependency of performance on block size, and endorsement policy shown in Table 9. The compared approaches are either based on Blockchain or not. The 1st approach is Efficient Anonymous Authentication with Conditional Privacy-Preserving Scheme for Vehicular Ad Hoc Networks (EAAP) [40]. 2nd approach is Data Security Sharing and Storage based on a Consortium Blockchain in a Vehicular Ad-hoc Network (DSSCB) [15]. 3rd approach is Applied Intelligence in Blockchain-based VANET (ALICIA) [41]. 4th approach is An Efficient Privacy-preserving Authentication Model based on Blockchain for VANETs (EPAM) [42].

Unlike EAAP, which uses centralized data storage, the other four approaches use distributed storage based on the consortium Blockchain. No centralized database is required because the strategy uses a database of trusted third-party organizations. This feature also protects against centralized harmful attacks on traditional centralized data storage.

Except for EAAP, all other four techniques are based on Blockchain and require the engaged vehicle node to sign data during the transaction writing phase. Data in the network can only be shared and received by legal and authenticated vehicles. The identity of the legal vehicle registered with the Trusted Authority can be verified. Therefore, the non-repudiation of data in all four techniques is ensured by knowing the source of the data sender using a digital signature mechanism. If there is a problem at any point in the network, the identity of the offender may be determined, ensuring the network's traceability feature.

In EAAP, each message has a valid signature and certificate, but it is mathematically hard to determine the signer. The attacker knows nothing about the signer. So even if the identities are collected, they do not expose the privacy of the vehicle users or RSUs.

They lack sufficient information regarding vehicle users or RSUs. Thus, with EAAP, the vehicles and RSUs remain anonymous.

The digital signature authentication mechanism in Blockchain transforms the vehicle's genuine identification into an anonymized identity, preserving anonymity by the four techniques based on Blockchain. Similarly, for these Blockchain-based techniques, an attacker will have great difficulty determining the vehicle's genuine identity, ensuring identity privacy protection.

EAAP technique as not based on Blockchain, and can't depict the dependency on block size and endorsement policy. This technique also does not analyze the performance parameters like throughput and latency.

DSSCB technique although based on Blockchain, the authors have not analyzed the dependency of performance on block size and endorsement policy. Also, they have not analyzed throughput and latency in their work.

In ALICIA, authors have analyzed throughput with respect to transactions per second, and latency with respect to block size for transactions. The authors have not analyzed the dependency of performance on block size and endorsement policy.

In EPAM, the authors have analyzed the performance parameters throughput and latency with respect to transaction arrival rate for different block sizes. The authors in EPAM have shown the dependency of these performance parameters on block size but not on endorsement policy.

Our proposed approach B-VANET has analyzed the performance of our system based on throughput and latency with respect to transaction arrival rate for different block sizes. Also, the proposed system shows the dependency of performance parameters like throughput and latency on block size and endorsement policy. This is to choose their optimal value to achieve the best performance.

VIII. CONCLUSION

Hyperledger Fabric is one of the prominent frameworks to develop and deploy Blockchain for any application. Fabric is a private, permissioned, modular, extensible Blockchain system for running distributed applications. The HFPBN is distinguished because of its performance, scalability, privacy, and strong consensus algorithm. The network comprises multiple components like channels, organizations, peers, smart contracts, ledgers, ordering service nodes, certification authority, and membership service provider. All the components have their designated role in making the fabric work and creating a transaction, which in turn collectively creates a block. VANET, one of the leading applications of Blockchain, is explored in this paper as a case study and implemented using fabric. Blockchain makes the VANET system more secure. For Blockchain-based VANET, the complete block size increases with the increase in the number of transactions, changing endorsement policy, and the number of reads and writes in a transaction. The proportionate impact of these factors on block size is analyzed using equations and graphs. Block size plays a crucial role in performance factors like throughput, latency, and network utilization. The results of our analysis for the VANET system show that till the saturation point, the block size has no impact on throughput, but after that saturation point, the higher block size will have higher throughput. The latency is also impacted by block size; the latency usually rises with block size until the saturation point. After saturation point, latency usually reduces with the rise in block size. Further, this system's memory and CPU utilization also rise with the rise in block size. The proposed B-VANET approach is compared with other Blockchain-based approaches to show our approach has focussed that the dependency of performance parameters on block size and endorsement policy is crucial for attaining the best possible performance.

REFERENCES

[1] M. Niranjanamurthy, B. N. Nithya, and S. Jagannatha, "Analysis of blockchain technology: Pros, cons and SWOT," *Cluster Comput.*, vol. 5, no. 2, pp. 1–15, 2018, doi: 10.1007/s10586-018-2387-5.

[2] W. Gao, W. G. Hatcher, and W. Yu, "A survey of blockchain: Techniques, applications, and challenges," in *Proc. 27th Int. Conf. Comput. Commun. Netw. (ICCCN)*, Jul. 2018, pp. 1–11, doi: 10.1109/ICCCN.2018.8487348.

[3] F. Knirsch, A. Unterweger, and D. Engel, "Implementing a blockchain from scratch: Why, how, and what we learned," *EURASIP J. Inf. Secur.*, vol. 2019, no. 1, p. 1–14, Dec. 2019, doi: 10.1186/s13635-019-0085-3.

[4] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *Decentralized Bus. Rev.*, p. 21260, Oct. 2008.

[5] S. Raza, S. Wang, M. Ahmed, and M. R. Anwar, "A survey on vehicular edge computing: Architecture, applications, technical issues, and future directions," *Wireless Commun. Mobile Comput.*, vol. 2019, pp. 1–19, Feb. 2019, doi: 10.1155/2019/3159762.

[6] R. Zhang, R. Xue, and L. Liu, "Security and privacy on blockchain," *ACM Comput. Surv.*, vol. 52, no. 3, pp. 1–34, May 2020, doi: 10.1145/3316481.

[7] J. B. Bernabe, J. L. Canovas, J. L. Hernandez-Ramos, R. T. Moreno, and A. Skarmeta, "Privacy-preserving solutions for blockchain: Review and challenges," *IEEE Access*, vol. 7, pp. 164908–164940, 2019, doi: 10.1109/ACCESS.2019.2950872.

[8] W. Wang, D. T. Hoang, P. Hu, Z. Xiong, D. Niyato, P. Wang, Y. Wen, and D. I. Kim, "A survey on consensus mechanisms and mining strategy management in blockchain networks," *IEEE Access*, vol. 7, pp. 22328–22370, 2018, doi: 10.1109/ACCESS.2019.2896108.

[9] P. Sajana, M. Sindhu, and M. Sethumadhavan, "On blockchain applications: Hyperledger Fabric and Ethereum," *Int. J. Pure Appl. Math.*, vol. 118, no. 18, pp. 2965–2970, 2018. [Online]. Available: http://www.ijpam.eu

[10] W. Meng, J. Wang, X. Wang, J. Liu, Z. Yu, J. Li, Y. Zhao, and S. S. M. Chow, "Position paper on blockchain technology: Smart contract and applications," in *Proc. Int. Conf. Netw. Syst. Secur.*, in Lecture Notes in Computer Science: Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics, vol. 11058, 2018, doi: 10.1007/978-3-030-02744-5_35.

[11] M. Valenta and P. Sandner, "Comparison of Ethereum, Hyperledger Fabric and Corda," *Frankfurt School Blockchain Center*, vol. 8, pp. 1–8, Jun. 2017, [Online]. Available: https://www.fs-blockchain.decontact@fs-blockchain.dewww.twitter.com/fsblockchainwww.facebook.de/fsblockchain%0Ahttps://medium.com/@philippsandner/comparison-of-ethereum-hyperledger-fabric-and-corda-21c1bb9442f6

[12] L. Jiang, X. Chang, Y. Liu, J. Mišić, and V. B. Mišić, "Performance analysis of Hyperledger Fabric platform: A hierarchical model approach," *Peer-Peer Netw. Appl.*, vol. 13, no. 3, pp. 1014–1025, May 2020, doi: 10.1007/s12083-019-00850-z.

[13] R. Shrestha, R. Bajracharya, A. P. Shrestha, and S. Y. Nam, "A new type of blockchain for secure message exchange in VANET," *Digit. Commun. Netw.*, vol. 6, no. 2, pp. 177–186, May 2020, doi: 10.1016/j.dcan.2019.04.003.

[14] L. Zhang, M. Luo, J. Li, M. H. Au, K.-K.-R. Choo, T. Chen, and S. Tian, "Blockchain based secure data sharing system for Internet of Vehicles: A position paper," *Veh. Commun.*, vol. 16, pp. 85–93, Apr. 2019, doi: 10.1016/j.vehcom.2019.03.003.

[15] X. Zhang and X. Chen, "Data security sharing and storage based on a consortium blockchain in a vehicular ad-hoc network," *IEEE Access*, vol. 7, pp. 58241–58254, 2019, doi: 10.1109/ACCESS.2018.2890736.

[16] J. Noh, S. Jeon, and S. Cho, "Distributed blockchain-based message authentication scheme for connected vehicles," *Electronics*, vol. 9, no. 1, p. 74, Jan. 2020, doi: 10.3390/electronics9010074.

[17] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. De Caro, and D. Enyeart, "Hyperledger Fabric: A distributed operating system for permissioned blockchains," in *Proc. 13th EuroSys Conf.*, Apr. 2018, pp. 1–15, doi: 10.1145/3190508.3190538.

[18] M. Kuzlu, M. Pipattanasomporn, L. Gurses, and S. Rahman, "Performance analysis of a Hyperledger Fabric blockchain framework: Throughput, latency and scalability," in *Proc. IEEE Int. Conf. Blockchain (Blockchain)*, Jul. 2019, pp. 536–540, doi: 10.1109/Blockchain.2019.00003.

[19] C. Gorenflo, S. Lee, L. Golab, and S. Keshav, "FastFabric: Scaling Hyperledger Fabric to 20,000 transactions per second," in *Proc. IEEE Int. Conf. Blockchain Cryptocurrency (ICBC)*, May 2019, pp. 455–463, doi: 10.1109/BLOC.2019.8751452.

[20] H. Yusuf and I. Surjandari, "Comparison of performance between Kafka and Raft as ordering service nodes implementation in Hyperledger Fabric," *Int. J. Adv. Sci. Technol.*, vol. 29, no. 7, pp. 3549–3554, 2020.

[21] H. Sukhwani, N. Wang, K. S. Trivedi, and A. Rindos, "Performance modeling of Hyperledger Fabric (permissioned blockchain network)," in *Proc. IEEE 17th Int. Symp. Netw. Comput. Appl. (NCA)*, Nov. 2018, pp. 1–8, doi: 10.1109/NCA.2018.8548070.

[22] C. Harris, "Improving telecom industry processes using ordered transactions in Hyperledger Fabric," in *Proc. IEEE Globecom Workshops (GC Wkshps)*, Dec. 2019, pp. 1–6, doi: 10.1109/GCWkshps45667.2019.9024541.

[23] M. Nofer, P. Gomber, O. Hinz, and D. Schiereck, "Blockchain," *Bus. Inf. Syst. Eng.*, vol. 59, no. 3, pp. 183–187, Jun. 2017, doi: 10.1007/s12599-017-0467-3.

[24] P. Thakkar, S. Nathan, and B. Viswanathan, "Performance benchmarking and optimizing Hyperledger Fabric blockchain platform," in *Proc. IEEE 26th Int. Symp. Modeling, Anal., Simulation Comput. Telecommun. Syst. (MASCOTS)*, Sep. 2018, pp. 264–276, doi: 10.1109/MASCOTS.2018.00034.

[25] M. Soelman, V. Andrikopoulos, J. A. Perez, V. Theodosiadis, K. Goense, and A. Rutjes, "Hyperledger Fabric: Evaluating endorsement policy strategies in supply chains," in *Proc. IEEE Int. Conf. Decentralized Appl. Infrastructures (DAPPS)*, Aug. 2020, pp. 145–152, doi: 10.1109/DAPPS49028.2020.00019.

[26] E. Zhou, H. Sun, B. Pi, J. Sun, K. Yamashita, and Y. Nomura, "Ledgerdata refiner: A powerful ledger data query platform for Hyperledger Fabric," in *Proc. 6th Int. Conf. Internet Things, Syst., Manage. Secur. (IOTSMS)*, Oct. 2019, pp. 433–440, doi: 10.1109/IOTSMS48152.2019.8939212.

[27] R. S. Raw, M. Kumar, and N. Singh, "Security challenges, issues and their solutions for VANET," *Int. J. Netw. Secur. Appl.*, vol. 5, no. 5, pp. 95–105, 2013.

[28] W. Liang, Z. Li, H. Zhang, S. Wang, and R. Bie, "Vehicular ad hoc networks: Architectures, research issues, methodologies, challenges, and trends," *Int. J. Distrib. Sensor Netw.*, vol. 11, no. 8, Aug. 2015, Art. no. 745303, doi: 10.1155/2015/745303.

[29] P. Gaba and R. S. Raw, "Vehicular cloud and fog computing architecture, applications, services, and challenges," in *IoT and Cloud Computing Advancements in Vehicular Ad-Hoc Networks*, R. S. Rao, V. Jain, O. Kaiwartya, and S. Nanhay, Eds. Pennsylvania, PA, USA: IGI Global, 2020, pp. 268–296.

[30] R. S. Raw, "The amalgamation of blockchain with smart and connected vehicles: Requirements, attacks, and possible solution," in *Proc. 2nd Int. Conf. Adv. Comput., Commun. Control Netw. (ICACCCN)*, Dec. 2020, pp. 896–902, doi: 10.1109/ICACCCN51052.2020.9362906.

[31] K. K. Rana, S. Tripathi, and R. S. Raw, "Opportunistic directional location aided routing protocol for vehicular ad-hoc network," *Wireless Pers. Commun.*, vol. 110, no. 3, pp. 1217–1235, Feb. 2020, doi: 10.1007/s11277-019-06782-4.

[32] P. Singh, R. S. Raw, and S. A. Khan, "Link risk degree aided routing protocol based on weight gradient for health monitoring applications in vehicular ad-hoc networks," *J. Ambient Intell. Humanized Comput.*, pp. 1–23, Apr. 2021, doi: 10.1007/s12652-021-03264-z.

[33] R. Hussain and S. Zeadally, "Autonomous cars: Research results, issues, and future challenges," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 2, pp. 1275–1313, 2nd Quart., 2019, doi: 10.1109/COMST.2018.2869360.

[34] G. Rathee, A. Sharma, R. Iqbal, M. Aloqaily, N. Jaglan, and R. Kumar, "A blockchain framework for securing connected and autonomous vehicles," *Sensors*, vol. 19, no. 14, pp. 1–15, 2019, doi: 10.3390/s19143165.

[35] S. Shalaby, A. A. Abdellatif, A. Al-Ali, A. Mohamed, A. Erbad, and M. Guizani, "Performance evaluation of Hyperledger Fabric," in *Proc. IEEE Int. Conf. Inform., IoT, Enabling Technol. (ICIoT)*, Feb. 2020, pp. 608–613, doi: 10.1109/ICIoT48696.2020.9089614.

[36] Q. Nasir, I. A. Qasse, M. A. Talib, and A. B. Nassif, "Performance analysis of Hyperledger Fabric platforms," *Secur. Commun. Netw.*, vol. 2018, pp. 1–14, Sep. 2018, doi: 10.1155/2018/3976093.

[37] *Hyperledger Blockchain Performance Metrics*, The Hyperledger White Paper Working Group, Linux Found., San Francisco, CA, USA, 2018, pp. 1–17. [Online]. Available: https://hyperledger.org/

[38] S. Hua, S. Zhang, B. Pi, J. Sun, K. Yamashita, and Y. Nomura, "Reasonableness discussion and analysis for Hyperledger Fabric configuration," 2020, *arXiv:2005.11054*.

[39] X. Xu, G. Sun, L. Luo, H. Cao, H. Yu, and A. V. Vasilakos, "Latency performance modeling and analysis for Hyperledger Fabric blockchain network," *Inf. Process. Manage.*, vol. 58, no. 1, Jan. 2021, Art. no. 102436, doi: 10.1016/j.ipm.2020.102436.

[40] M. Azees, P. Vijayakumar, and L. J. Deboarh, "EAAP: Efficient anonymous authentication with conditional privacy-preserving scheme for vehicular ad hoc networks," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 9, pp. 2467–2476, Sep. 2017, doi: 10.1109/TITS.2016.2634623.

[41] S. R. Maskey, S. Badsha, S. Sengupta, and I. Khalil, "ALICIA: Applied intelligence in blockchain based VANET: Accident validation as a case study," *Inf. Process. Manage.*, vol. 58, no. 3, May 2021, Art. no. 102508, doi: 10.1016/j.ipm.2021.102508.

[42] X. Feng, Q. Shi, Q. Xie, and L. Liu, "An efficient privacy-preserving authentication model based on blockchain for VANETs," *J. Syst. Archit.*, vol. 117, Aug. 2021, Art. no. 102158, doi: 10.1016/j.sysarc.2021.102158.

PRIYANKA GABA received the M.Tech. degree from the Department of Computer Science, Kurukshetra University, Kurukshetra, India, in 2013. She is currently pursuing the Ph.D. degree with the University School of Information, Communication and Technology, Guru Gobind Singh Indraprastha University, New Delhi, India (Associated Supervisor with Netaji Subhas University of Technology, East Campus). She is an Assistant Professor with the School of Computing Science and Engineering, Galgotias University, Greater Noida, India. Her research interests include vehicular security, blockchain, and vehicular ad-hoc networks.

RAM SHRINGAR RAW (Member, IEEE) received the B.E. degree in CSE, in 2000, the M.Tech. degree in IT, in 2005, and the Ph.D. degree in computer science and technology from the School of Computer and Systems Sciences, Jawaharlal Nehru University, New Delhi, India, in 2011. He has been an Assistant Professor with the Department of Computer Science and Engineering, Netaji Subhas University of Technology, New Delhi, since 2011. He has published six books, five patents and more than 100 research papers with good impact factors in reputed international journals and conferences, including IEEE, Elsevier, Springer, Wiley & Sons, Taylor & Francis, and Hindawi. He has guided more than 60 B.Tech. students for their projects and more than 25 M.Tech. and Ph.D. students for their dissertation and thesis. He has received four awards for best research paper and research works. His current research interests include mobile ad hoc networks, vehicular ad hoc networks, flying ad-hoc networks, and the IoT cloud.

MAZIN ABED MOHAMMED received the B.Sc. degree in computer science from the University of Anbar, Iraq, in 2008, the M.Sc. degree in information technology from UNITEN, Malaysia, in 2011, and the Ph.D. degree in information technology from UTeM, Malaysia, in 2019. He is currently a Lecturer with the College of Computer Science and Information Technology, University of Anbar. His research interests include artificial intelligence, biomedical computing, and optimization.

JAN NEDOMA (Senior Member, IEEE) is currently an Associate Professor and the Head of the Optoelectronics Laboratory with the Department of Telecommunications, Faculty of Electrical Engineering and Computer Science, Technical University of Ostrava. He is also a member of the scientific council, a member of doctoral, habilitation, and professors committees, and a guarantor of bachelor's study programs at the mentioned faculty. During his scientific career, he was the leader or a co-investigator of more than 25 projects and has more than 170 journal articles and conference papers in his research areas. He holds ten valid Czech patents. His research interests include optical communications and optical atmospheric communications, optoelectronics, optical measurements, measurements in telecommunication technology, signal processing, fiber-optic sensors, and biomedical engineering.

RADEK MARTINEK (Senior Member, IEEE) is currently a Full Professor of Cybernetics with the Faculty of Electrical Engineering and Computer Science, Technical University of Ostrava. He is currently the Vice-Dean of science and research and the Deputy Head of the Department of Cybernetics and Biomedical Engineering. His research interests include hybrid and bio inspired methods for advanced signal processing, pedagogical practice, results and deployment of novel experimental algorithms in the field of cybernetics and biomedical engineering. He is the author of more than 300 publications with over 2000 citations and an H-index of 21. He also holds ten Czech national patents and is a leader or co-leader of dozens of projects with a budget of millions of euros.