Graphical Consensus-based Sharding for Efficient and Secure Sharings in Blockchain-enabled Internet of Vehicles

Li, W., Zhao, Z., Ma, P., Xie, Z., Palade, V. & Liu, H.

Original citation & hyperlink:
Li, W, Zhao, Z, Ma, P, Xie, Z, Palade, V & Liu, H 2023, 'Graphical Consensus-based Sharding for Efficient and Secure Sharings in Blockchain-enabled Internet of Vehicles', IEEE Transactions on Vehicular Technology, vol. (In-Press), pp. (In-Press).
https://doi.org/10.1109/tvt.2023.3311445

DOI 10.1109/tvt.2023.3311445
ISSN 0018-9545
ESSN 1939-9359

Publisher: IEEE

Graphical Consensus-based Sharding for Efficient and Secure Sharings in Blockchain-enabled Internet of Vehicles

Wenqi Li, *Student Member, IEEE*, Zheng Zhao, Pingchuan Ma, Zeqiang Xie, Vasile Palade, *Senior Member, IEEE*, Hongbo Liu

***Abstract*—Blockchain-enabled Internet of Vehicles provides a reliable collaboration environment for traffic entities and promotes road safety and traffic efficiency through real-time sharings between vehicles and infrastructure. However, a bottleneck is likely to emerge as the Internet of Vehicles scales up. In this paper, we propose a novel Graphical Consensus-based Sharding (GCS) framework, which is underpinned by four important strategies: 1) A graphical consensus is adopted as its intra-shard consensus, where the consensus group is set up according to its maximal connected subgraph, and the leader is elected by its reliability weight. The consensus group is refreshed intermittently by alternating the leader role. 2) Within GCS, the intra-shard data are stored in the local chain, while a block-based directed acyclic graph, rather than a chained structure, is employed as the main chain. The local chain is used to respond to requests within each shard, and the main chain supports the cross-shard sharings. GCS will parallelly optimize the throughput of the blockchain-enabled Internet of Vehicles. 3) GCS further introduces the shard backup and the node scheduling to handle shard failure and overheating by using new backup strategies and a temporal-spatial graph convolutional network prediction model, respectively. 4) An off-chain transmission algorithm is presented for secure sharing between the infrastructure and the vehicles. Simulation results show that the number of Transactions Per Second is 1.69 times higher than that of the non-sharding blockchain, and the pending time is dramatically reduced compared to the mainstreaming sharding approach, which is 1.02s.**

***Index Terms*—blockchain-enabled Internet of Vehicles, sharding, graphical consensus, secure data sharing, throughput.**

I. Introduction

BLOCKCHAIN-enabled Internet of Vehicles (BIoVs) establishes a trustworthy and reliable collaboration environment between the vehicles and the infrastructure through the blockchain technology, which enhances the road safety and traffic efficiency by real-time and low-latency sharing of vehicular data [1], [2]. However, as the scale of BIoVs expands, the data sharing suffers from a bottleneck that hinders its scalability and throughput [3], [4].

Manuscript received xx, xxxx; revised xx, xxxx; accepted xx, xxxx. This work is supported in part by the National Natural Science Foundation of China (Grant Nos. 61772102, 62176036).

W. Li, Z. Zhao, P. Ma, Z. Xie and H. Liu are with the College of Artificial Intelligence at Dalian Maritime University, with an affiliate appointment in the Dalian Key Laboratory of Urban Traffic Safety and Intelligent Security, Dalian 116026, China (e-mail: wqli@ieee.org, {zhaozheng, mapc, zeqiang, lhb}@dlmu.edu.cn).

V. Palade is with the Centre of Computational Science and Mathematical Modeling, Coventry University, Coventry CV1 5FB, UK (e-mail: vasile.palade@coventry.ac.uk).

Recently, the blockchain sharding was introduced into BIoVs, as shown in Fig. 1. There already are some roadside units (RSUs) equipped with multi-access edge computing servers, infrastructure with rich sensors, vehicles with onboard sensors, etc. These entities compose the sharding BIoVs, and its blockchain is divided into smaller pieces, each of which is known as the "shard". It also maintains disjoint ledgers (i.e., transactions) by a group of RSUs within one shard, called the "committee" [5], [6], [7]. This approach is called "state sharding" [8]. Users (i.e., vehicles) request service transactions to interact with the sharding BIoVs. Each committee maintains its intra-shard ledger and processes local transactions, which helps implement the systematic parallelization and enhances the throughput and Transactions Per Second (TPS), by introducing the sharding BIoVs [3], [9].

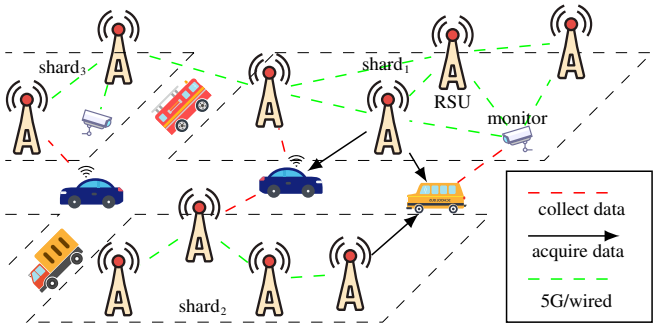


Fig. 1. Sharding BIoVs. The overall BIoVs is divided into multiple shards, each of which handles its transactions in parallel, and vehicles on the road communicate with the nearest RSU based on their locations.

However, this gives rise to some security and efficiency problems when carrying out sharding BIoVs. The RSUs would turn into Byzantine nodes due to network failures and attacks [10], [11]. Whenever the committee contains a relatively higher proportion of Byzantine nodes, or the leader has Byzantine behavior, the shard would fail to reach consensus, known as "consensus failure" [12]. Furthermore, once the number of Byzantine nodes exceeds its threshold, i.e., 1/2, the shard will not process the transactions correctly. This refers to a "shard failure" [13]. Meanwhile, because of the tidal migration and the high mobility of vehicles, some shards may have a heavy backlog of transactions that cannot be processed promptly [14]. This is so-called "shard overheating" [15], [16]. Moreover, although the sharding BIoVs ensures the security of on-chain data, the off-chain sharings between vehicles and

infrastructure are still faced with threats, which is so-called "off-chain attack" [17]. Such issues are important challenges in developing efficient and secure sharing in BIoVs [18], [19], [20].

In this paper, our key motivation is to introduce an efficient and secure sharing methodology to cope with the above challenges by integrating the features of BIoVs. Therefore, this paper proposes a novel Graphical Consensus-based Sharding (GCS) framework to implement an intra-consensus, in which the efficiency is improved by selecting the active and reliable nodes to participate in the graphical consensus. The shard backup is adopted in response to shard failure, which employs backup shards to replace the failed ones. The node scheduling is further used to handle the shard overheating, which involves a prediction model to forecast the road conditions. We also present an off-chain transmission algorithm for secure sharing between infrastructure and vehicles.

The contributions of this paper are as follows:

- We introduce a novel sharding framework for BIoVs named GCS. The graphical consensus is adopted as the intra-shard consensus, where the maximal connected subgraph of the committee gathers the consensus group, and the leader is elected among them according to the reliability weights. The intra-shard data are stored in the local chain, while the block-based directed acyclic graph generates the cross-shard main chain.
- The temporal-spatial graph convolutional network model and shard backup strategies are introduced for GCS in order to deal with the shard overheating and the shard failure. And the off-chain data transmission is proposed for secure sharing between the vehicles and the infrastructure.
- We analyze the proposed GCS theoretically. In particular, we demonstrate that: 1) The off-chain sharings among RSUs and vehicles cannot be tampered with. 2) The on-chain data stored by each shard are secure as long as the Byzantine nodes proportion does not exceed the threshold. The simulation of GCS was carried out using NS-3, and the results show that GCS has a high TPS and a short transaction pending time.

The rest of this paper is organized as follows: Section II surveys the related works. Section III presents the detail of the GCS framework and its theoretical analysis. In section IV, our proposed GCS is simulated in NS-3 and evaluated by three standard metrics. Finally, we draw some conclusions in section V.

II. Related work

A. Consensus in BIoVs

With the ongoing development of BIoVs, many studies have considered the consensus mechanism [21], [22]. For example, Vishwakarma et at. [23] proposed a modified practical Byzantine fault tolerance consensus for permissioned BIoVs, which ensures fairness and gives equal chances to all the miners to generate blocks. Mišić et al. [24] introduced a practical Byzantine fault tolerance (PBFT) that integrated the Proof-of-Stake algorithm. Meanwhile, some researchers focus on selecting a part of the nodes to participate in the consensus to reduce the communication volume [25]. Gao et al. [26] presented a BIoVs framework, in which a small group of preselected nodes runs the PBFT consensus to verify transactions. Kudva et al. [27] introduced a secure and practical consensus for BIoVs, and the Proof-of-Driving is employed to select miners. Hou et al. [28] proposed a modified Proof-of-Reputation consensus, which uses the hash value to reduce the number of consensus nodes. Xu et al. [29] presented an improved PBFT algorithm for BIoVs, in which consensus nodes are selected according to the score and updated every 50 requests. However, the efficiency of PBFT consensus is vulnerable to Byzantine nodes, while it has a bottleneck in the face of the increasing data volume in IoVs. Therefore, designing an efficient and Byzantine-resistant consensus is urgently demanded in latency-sensitive and security-critical scenarios like IoVs.

B. Sharding and blockchain structure

Sharding is considered an effective solution to improve the computing capacity, communication capacity, and storage capacity of blockchain [30], [31], [32]. Elastico [6], as the first proposed sharding framework, divides the nodes into multiple fragments, each of which processes transactions independently. OmniLedger [33] introduced a new node allocation algorithm and cross-shard transactions processing protocol based on Elastico. RapidChain [7] is a sharding-based permissionless blockchain, and its performance has been dramatically improved compared with the previous sharding approaches. Zilliqa [34], as a sharding framework used in industry, achieves network and computational sharding and enhances the throughput. Wang et al. [3] proposed a multi-sharding protocol to decrease the communication costs in BIoVs. Singh et al. [35] proposed an adaptive trust management framework for BIoVs, which adopts sharding to reduce the load and increase the throughput. Wen et al. [36] presented a sharding blockchain-based framework for IoVs to increase the calculation speed and reduce the burden. Xie et al. [5] proposed the DAG blockchain with a sharding approach, in which all committees maintain a global DAG for transaction verification. Li et al. [37] introduced a DAG-based consensus for privacy preservation and traffic policing. Zhang et al. [9] presented a parallel consensus by adopting the network sharding and DAG-lattice, where each participant has their account ledger. With the exponential increase of users in BIoV, it is necessary to adopt sharding and the DAG structure to improve the throughput.

III. Design of GCS

This section first introduces in detail the proposed GCS framework, followed by a theoretical analysis. The GCS framework adopts the state sharding approach and it contains four parts, including the graphical consensus, the sharding blockchain design, the node scheduling and shard backup, and the secure off-chain sharings. The overall architecture is shown in Fig. 2, and the main symbols used in the paper are shown in Table I.

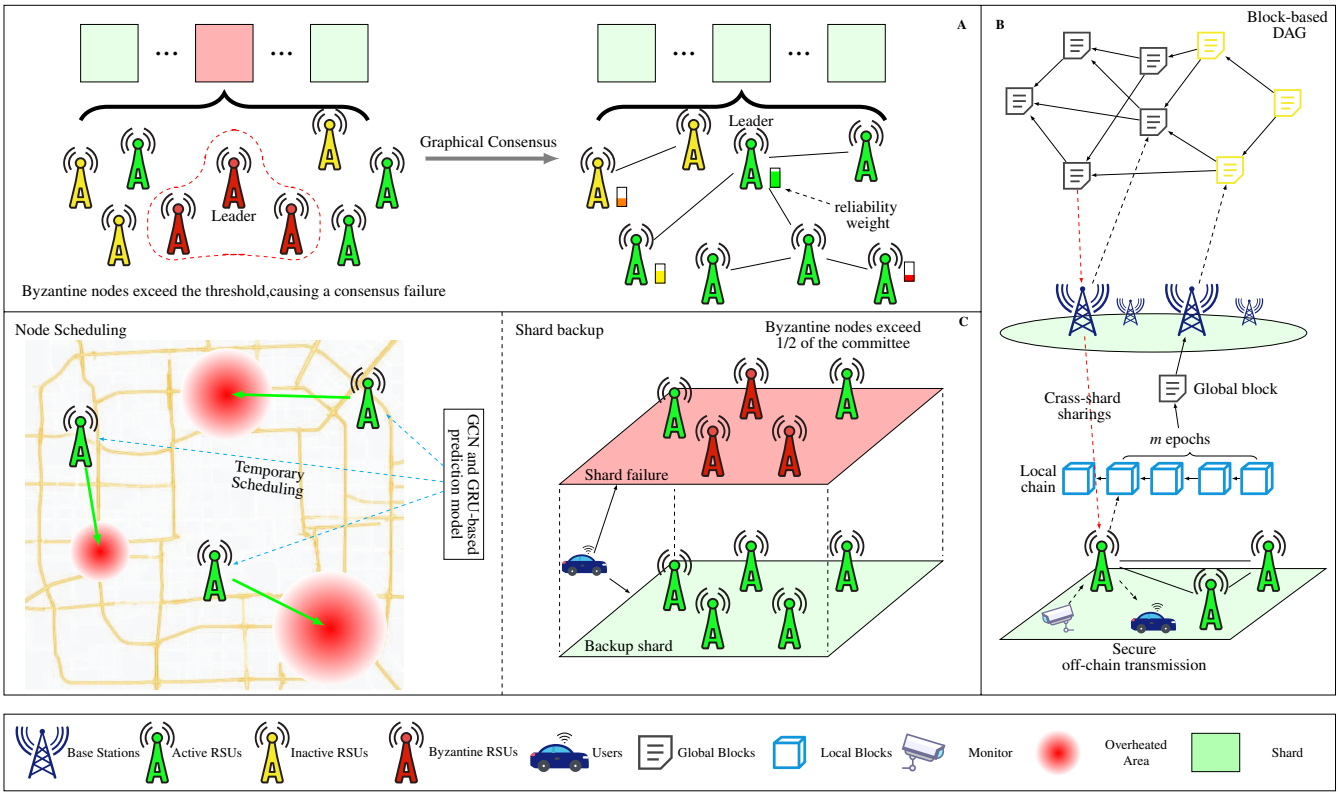


Fig. 2. The overall architecture of GCS. The participants are RSUs, base stations (BSs), monitors, and vehicles, where the BSs have strong computing and storage capacity, monitors are part of the infrastructure and vehicles that are equipped with rich sensors. RSUs within a certain area form the shard, and the graphical consensus is used as the intra-shard mechanism, shown in A. The structure of the local chain and block-based DAG are illustrated in B. Meanwhile, the node scheduling and shard backup approaches are shown in C.

TABLE I
SUMMARY OF MAIN NOTATIONS

Notation	Description
$\mathcal{C}$	Consensus group, a set of nodes participating in consensus
$\mathcal{S}$	The set of all the shards
l	Leader node of each consensus group
b	Number of Byzantine nodes
w	Reliability weight
α	Shards connectivity
$\langle p, k\rangle$	Public and private key pair of data sender
$\langle \acute{p}, \acute{k}\rangle$	Public and private key pair of data receiver
TPS	Transactions Per Second
FP	Failure probability
PT	Average pending time of a transaction

A. Graphical consensus

RSUs are divided into individual shards according to their geographical location. A consensus group is set up and the leader is elected to initiate the graphical consensus before the shard can process users' transactions. This will eliminate RSUs that are disconnected from others due to control by attackers and network failures.

Definition 1. [Consensus group $\mathcal{C}$] Given the graph

$$G = (V, E) = \{g_1, g_2, \cdots, g_n\},$$

where V is the set of RSUs within each shard, E represents the communications between them in the past x epochs. There exists a subgraph $g_i = (V_i, E_i)$, $V_i \in V$, $E_i \in E$ that satisfies,

$$\begin{aligned} &|V_i| = \max\{|V_1|, \cdots, |V_n|\}, \\ &s.t. \quad P(v_j \to \cdots \to v_k \,|v_j, v_k \in V_i) = 1, \\ &\qquad P(v_m \to \cdots \to v_n \,|v_m \in V \backslash V_i, v_n \in V_i) = 0, \end{aligned}$$

where $|V_i|$ represents the number of nodes in set V_i and the RSUs in V_i form the consensus group $\mathcal{C}$ of each shard.

In one shard, each node can accumulate reliability weight by honest behaviors, i.e., packing blocks and voting correctly in the past n epochs. The reliability weight w of each node can be calculated as in Eq. (1).

$$w = \sum_{k=1}^{n} R(k), \tag{1}$$

where $R(k)$ calculates the numbers that a node shows honest behaviors at epoch k. It is worth mentioning that once a node exhibits Byzantine behaviors, w will be reset to zero, and it cannot participate in consensus in the upcoming m epochs as a penalty.

Definition 2. [Leader node l] Given the consensus group $\mathcal{C}$, the leader node $l \in \mathcal{C}$ satisfies:

$$l = \arg\max_{i \in \mathcal{C}} \{w_i\}. \tag{2}$$

The Practical Byzantine Fault Tolerance [38] (PBFT) algorithm is used within the consensus group of each shard to

verify the transactions, and the diagram is shown in Fig. 3. After receiving the transaction requests, the leader first sends the Pre-prepare message to all the consensus nodes. And on receipt of the Pre-prepare message, the node broadcasts the Prepare message. In a consensus group with $3b+1$ nodes, once any node receives $2b+1$ Prepare messages, it will broadcast the Commit message. The users' requests are verified, and the consensus result will be returned to the client after receiving $2b+1$ Commit messages by the leader.

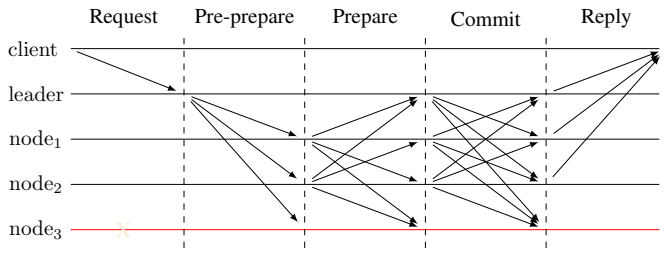


Fig. 3. The diagram of PBFT.

When the current leader l shows Byzantine behaviors, the view change protocol will be initiated to select a new leader, and the new leader $\hat{l}$ should satisfy:

$$\hat{l} = \arg\max_{j \in \mathcal{C}\backslash l} \{w_j\}, \tag{3}$$

where $\hat{l} \in \mathcal{C}\backslash l$. And after a successful view change, the previous leader node will be recognized as a Byzantine node and penalized.

The graphical consensus is summarized in Algorithm 1.

ALGORITHM 1: Graphical Consensus

Input: Intra shard graph G, the set of transactions TX
Output: Consensus result r

for $v \in V$ **do**
  if $v \in V_i$ **then**
    $\mathcal{C} \leftarrow \mathcal{C} + \{v\}$ ▷V_i refers to Def. 1
    $\leftarrow + \{\omega_v\}$ ▷ω is calculated as Eq. (1)
  end
end
$l \leftarrow \arg\max_{j \in \mathcal{C}} \{\omega_j\}$ ▷Leader election
for $tx \in TX$ **do**
  if $l \in$ *Byzantine* **then**
    $\omega_l \leftarrow 0$
    $\leftarrow \backslash \omega_l$
    $\mathcal{C} \leftarrow \mathcal{C}\backslash l$
    $\hat{l} \leftarrow \arg\max_{j \in \mathcal{C}} \{\omega_j\}$ ▷View change
  end
  $r \leftarrow \text{PBFT}(tx)$ ▷Transactions verification
  return r
end

B. Sharding blockchain design

In our framework, there are two types of structures:

1) Local chain. The local chain of each shard stores the historical ledger independently and is maintained by all the RSUs. The data collected by the monitors will be sent to the nearby RSUs and verified within the consensus group. Then, the authorized data will be packed into a block and stored in the local chain, which is supported by the corresponding RSUs. When the users request intra-shard data, RSUs can retrieve it from the local chain and send it to the user.

2) Block-based DAG. The block-based DAG acts as the main chain of GCS, which is maintained by the BSs. The block-based DAG adopts the Tangle structure as IOTA [39]. Differently, after m epochs in the local chain, a global block that contains all transactions during this period is created and sent to the BSs. Then the BSs will synchronize blocks to the main chain using wired and wireless networks.

Each block in the block-based DAG can accumulate weight by approving the previous blocks. And the cumulative weight L_j of block j is the sum of the cumulative weights of all the previous blocks that are directly and indirectly approved by block j.

In the block-based DAG, the BSs must approve two previous blocks to synchronize the newly generated global block to the main chain, and the chosen probability is calculated as Eq. (4).

$$P_{n \mapsto m} = \frac{\theta \exp(L_m - L_n)}{\sum_{y: y \mapsto m} \theta \exp(L_m - L_y)}, \tag{4}$$

where $y \mapsto m$ represents that block y approves block m, $m \leq y \leq n$ and θ is a constant factor.

When a cross-shard transaction is received by the RSUs, it will be forwarded to the BSs. After verification, the transaction will be recorded on the block-based DAG main chain and the requested data will be returned to the users.

Local chains enhance the efficiency of processing intra-shard transactions, while block-based DAG provides efficient cross-shard data access. This structure will save time in synchronizing blocks into the main chain, and the consistency of the data is also guaranteed.

C. Node scheduling and shard backup

Node scheduling is adopted to handle shard overheating. We utilize temporal-spatial graph convolutional network model to predict overheated areas, and then schedule the resources into these areas. This will speed up the verification of the backlog of transactions in the overheated areas and accelerate the alleviation of shard overheating.

Definition 3. [Road graph G and binary matrix A] Using graph $G = (V, E)$ to represent the topological structure of roads in a city. V is the set of roads. E is the connection between roads, which is represented by a binary matrix $A \in \{0,1\}^{N \times N}$, where N represents the number of roads, and '1' means that the two roads are connected, otherwise '0'.

Definition 4. [Feature Matrix X] Taking the average traffic speed on each road of G as the attribute features and the feature matrix $X \in \mathbb{R}^{N \times n}$ is a $N \times n$-dimensional value, where n represents the length of the historical time series and $X_t \in \mathbb{R}^{N \times t}$ represents the traffic speed of N roads at time t.

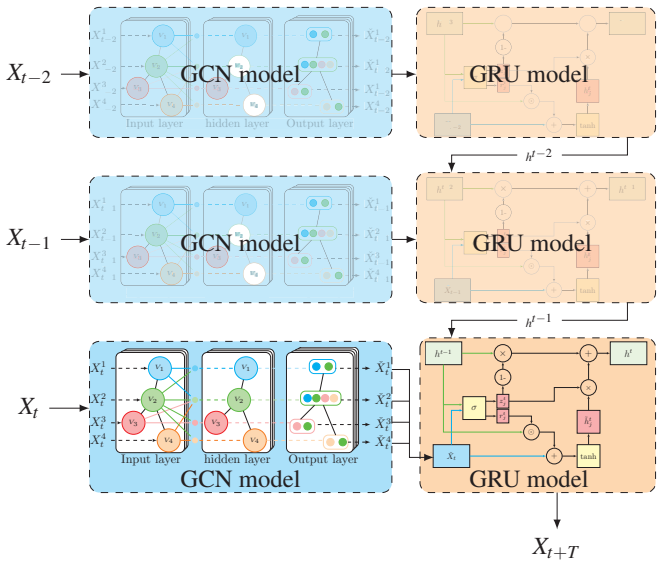


Fig. 4. Temporal-spatial graph convolutional network prediction model

Therefore, the problem of traffic prediction can be formulated as a learning function mapping f based on graph G and feature matrix X, and calculating the average traffic speed of the next period T as follows:

$$[X_{t+1}, X_{t+2}, \ldots, X_{t+T}] = f(G, (X_{t-n}, \ldots, X_t)), \tag{5}$$

where n is the time series length of the learned historical data and T is the time series length to be predicted.

In GCS, the BSs run the prediction model to forecast traffic speeds and locate the overheated shards according to the model's output. The architecture is shown in Fig. 4.

Here, a 2-layer Graph Convolutional Network (GCN) [40] with the activation functions of ReLU$(\cdot)$ and softmax$(\cdot)$ is used to learn spatial features from traffic data, which can be expressed as:

$$\check{X} = f(X, A) = \text{softmax}(\hat{A}\text{ReLU}(\hat{A}XW^{(0)})W^{(1)}), \tag{6}$$

where $\hat{A}$ represents the pre-processing step in GCN, $W^{(0)} \in \mathbb{R}^{n \times H}$ is a $n \times H$-dimensional value, which represents the weight matrix from input layer to hidden layer, H is the number of hidden units, $W^{(1)} \in \mathbb{R}^{H \times T}$ is a $H \times T$-dimensional value that represents the weight matrix from hidden layer to the output layer, respectively.

As described in T-GCN [41], the Gated Recurrent Unit (GRU) model [42] is used to capture the temporal features as follows:

$$h^t = \text{GRU}(\check{X}_t, h^{t-1}), \tag{7}$$

where $\check{X}_t$ is the output of GCN model at time t according to Eq. (6), h^{t-1} is the output state of the GRU model at time $t-1$.

According to the model's output $O = [X_{t+1}, \ldots, X_{t+T}]$, the road i will be considered congested at time k if $X_k^i \leq$ 30km/h, $X_k^i \in O$. And the score Γ of nodes that can be temporarily transferred is calculated as in Eq. (8).

$$\Gamma = \frac{\sum_{j=1}^{z} \Delta t_j}{z} \times \exp\left(-\Delta P\right), \tag{8}$$

where Δt_j represents the time spent on processing the transaction j, z represents the number of transactions that were processed by the node in the last epoch, and ΔP represents the distance between the node and the target shard, respectively.

According to the final score Γ of each node, RSUs are temporarily reallocated into designated shards and determine connectivity with others through IP address. After synchronizing the local chain and checking the data consistency using the Merkle tree, these RSUs will then participate in the consensus group and process transactions. The closer distance between RSUs and the overheated shard guarantees the quality of the communication, and the high capacity of the scheduled RSUs enables the consensus group to complete consensus faster. This will speed up the verification of the backlog of transactions and the alleviation of shard overheating.

To cope with shard failure, we employ shard backup. For each shard, an appropriate shard will be chosen as its backup. The backup shard will keep synchronization. Once a shard fails, its backup will take its place until its recovery.

Definition 5. [Shards connectivity α] For each shard S_i and S_j, the connectivity $\alpha_{i,j}$ can be calculated as:

$$\alpha_{i,j} = \sum_{k=1}^{x} t_{i,j}(k), \tag{9}$$

where $t_{i,j}(k)$ calculates the number of vehicles coming from shard S_j to shard S_i at time k.

Definition 6. [Backup shard $\hat{S}$] For shard $S_i \in \mathcal{S}$, the backup $\hat{S}$ is obtained by Eq. (10).

$$\hat{S} = \arg\max_j \eta_{i,j}, \tag{10}$$

$$\eta_{i,j} = -\exp(\Delta P_{i,j}) \times (Z_j + \alpha_{i,j}),$$

$$s.t.\ \bar{\alpha}_i \times \Delta P_{i,j} \leq \alpha_{i,j} \times \min_{k \in S, k \neq j} \{\Delta P_{k,j}\}, \tag{10a}$$

$$\sum_{i=1}^{|\mathcal{S}|} \varrho_{i,j} \times \Phi_i \leq R_j, \tag{10b}$$

$$\sum_{m=1}^{P} \sum_{n=1}^{Q} (\ell_n^i + \ell_m^j) \leq B_j, \tag{10c}$$

$$\sum_{m=1}^{P} \sum_{n=1}^{Q} (c_n^i + c_m^j) \leq Z_j, \tag{10d}$$

where Z represents the processing capacity of shard, $\bar{\alpha}_i$ is the average connectivity of shard S_i. $\Delta P_{i,j} = \|P_j - P_i\|$, P_i and P_j represent the position of each shard. ϱ is a binary value, '1' means shard S_i is backed up to S_j, otherwise '0'. Φ is the needed storage space, R represents the remaining storage resources, ℓ_n^i is the bandwidth required for the n-th transaction communication of shard S_i, B is the total bandwidth resources, c_n^i represents the required computing resources of the n-th transaction in S_i, respectively.

Constraint (10a) means that the data of S_i are backed up in S_j, the connection degree should be large among all the other connected shards while the distance between the two shards should be as small as possible. Constraint (10b) means that the remaining storage resources of the target shard must be sufficient to back up the historical transactions of the source

shard. Constraint (10c) means that the network condition of the target shard must be sufficient to support all the users' communications. Constraint (10d) means that the computing resources of the target shard should be able to verify the two shards' transactions simultaneously, respectively.

D. Secure Off-chain sharings

To prevent off-chain sharings between vehicles and infrastructure from being tampered with by attackers and to protect the security of sharings, the secure off-chain sharings is proposed.

The system first initializes the functions shown in Table II, and the participants generate a random 32-bit string as the private key k and calculate the public key $p = SECP256K1(k)$.

TABLE II
FUNCTION INITIALIZATION

Function	Implication
SECP256K1 [43]	elliptic curve cryptography [44], [45]
$H(\cdot)$	secure hash algorithm, sha256 [46]
$K(\langle h_1, h_2, \cdots, h_n\rangle, p)$	public key encryption function
$(m, \varsigma, \epsilon) \leftarrow D(M, k)$	private key decryption function
$S(\langle h_1, h_2, \cdots, h_n\rangle, k)$	private key signature function
$(0, 1) \leftarrow \check{V}(s, p)$	signature verification function

When the monitor collects data m, it first calculates the index ς of m using function $H(\cdot)$ and the sign ϵ using function $S(\cdot)$. Then encrypt the collected data m, index ς and sign ϵ using $K(\cdot)$ and the public key $\acute{p}$ of the RSU and send the encrypted message M. When an RSU receives a message M from a monitor, it uses private key $\acute{k}$ and function $D(m, k)$ to decrypt. Then calculate the index ς' of the decrypted data m', which is $\varsigma' = H(m')$, and compare it with ς. Due to the nature of the hash function $H(\cdot)$, once ς' is the same as ς, we can say that the decrypted data m are accurate. The procedure is summarized in Algorithm 2.

Algorithm 3 illustrates the secure data sharings among users and RSUs. First, the RSU releases the authorized signatures ϵ for the users that are requesting data access. Then, the RSU encrypts the signature using the function $K(\cdot)$ and the public key of each user. After decrypting the authorized signature, the users may require data access from the nearby RSUs. After receiving the user's request, the RSU will verify whether it is an authorized user using $\check{V}(\cdot)$ and the public key p of the signed RSU. After identity verification, the RSU sends the requested data to the user by calling Algorithm 2. And finally, the user will get accurate data on the blockchain. Algorithms 2 and 3 provide the off-chain data acquisition procedures and communication security between terminals and RSUs.

ALGORITHM 2: Off-chain Data Transmission Algorithm

Input: Sender node set R with key pairs set $\langle p, k\rangle$, public key $\acute{p}$ and private key $\acute{k}$ of receiver node, packed data set m.
Output: Secure transmitted data M'.

for $r \in R$ **do**
  $\varsigma \leftarrow H(m_r)$
  $\epsilon \leftarrow S(\varsigma, k_r)$
  $\dot{m}_r \leftarrow K(\langle \varsigma, \epsilon, m\rangle, \acute{p})$
  $M \leftarrow \{M \oplus \dot{m}_r\}$
end
for $M_i \neq \varnothing \in M$ **do**
  $(m', \varsigma, \epsilon') \leftarrow D(M_i, \acute{k})$
  $\varsigma' = H(m')$
  if $\varsigma = \varsigma'$ & $\check{V}(\epsilon', p_i)$ **then**
    $M' \leftarrow \{M' \oplus m'\}$
    return M'
  else
    return 0
  end
end

ALGORITHM 3: Secure Data Sharing Algorithm

Input: Public key p and private key k of the RSU, users set U with key pairs $\langle \acute{p}, \acute{k}\rangle$, authorized time period t.
Output: Required data d'.

for $u \in U$ **do**
  $\epsilon \leftarrow S(\langle t, \acute{p}_u\rangle, k)$
  $\dot{m}_r \leftarrow K(\epsilon, p)$
  $M \leftarrow \{M \oplus \dot{m}_r\}$
end
for $M_i \neq \varnothing \in M$ **do**
  $\epsilon \leftarrow D(M_i, k)$
  if $\check{V}(\epsilon, p)$ **then**
    Calling Algorithm 2
  else
    return 0
  end
end

E. Theoretical analysis

In this subsection, we first prove the security of off-chain sharings and then analyze the consensus and shard resiliency and the complexity of GCS.

Security analysis:

Theorem 1. The data transmission mechanism in Algorithm 2 guarantees the collected data is secure and cannot be tampered with during the off-chain transmission stage, which is formulated as:

$$\begin{aligned}&\forall\ m, \bar{m}, \bar{k},\\ &1 \leftarrow (\varsigma \wedge H(m)) \wedge (\check{V}(\epsilon, p)),\\ &0 \leftarrow (\bar{\varsigma} \wedge H(\bar{m})) \wedge (\check{V}(\epsilon, \acute{p}_i)) \wedge (\varsigma \wedge H(\bar{m})),\end{aligned}$$

where $\bar{m}$ and $\bar{k}$ are the forged message and private key, respectively.

Proof. According to Algorithm 2, the edge node (i.e., a monitor) packages the collected data m, index ς and signature

ϵ, then encrypts the message using the public key of the target RSU, which is:

$$\begin{aligned}\varsigma \leftarrow & H(m), 1 \leftarrow \varsigma \wedge H(m),\\ \epsilon \leftarrow & S(\varsigma, k), 1 \leftarrow \check{V}(\epsilon, p),\end{aligned}$$

and thus,

$$1 \leftarrow (\varsigma \wedge H(m)) \wedge (\check{V}(\epsilon, p)).$$

Since function $K(\cdot)$ is an asymmetric encryption function, only the holder of the corresponding private key can decrypt the message. Therefore, when the message between the monitor and RSU is intercepted, the attacker cannot directly obtain the message's plaintext and tamper with it because of the lack of a private key.

However, the attacker can forge the communication between the monitor and the RSU, which is:

$$\begin{aligned}\bar{\varsigma} &\leftarrow H(\bar{m}),\\ \bar{\epsilon} &\leftarrow S(H(\bar{m}), \bar{k}),\\ \bar{\bar{m}} &\leftarrow K(\langle \bar{m}, \bar{\varsigma}, \bar{\epsilon}\rangle, \acute{p}),\end{aligned}$$

where $\bar{\varsigma}$ is the index of the forged message, $\bar{\epsilon}$ is the forged signature by attacker.

After receiving the message $\bar{\bar{m}}$, the RSU first decrypts it using its own private key $\acute{k}$ and obtains:

$$\begin{aligned}1 \leftarrow & \bar{\varsigma} \wedge H(\bar{m}),\\ 0 \leftarrow & \check{V}(\bar{\epsilon}, \acute{p}),\\ 0 \leftarrow & (\bar{\varsigma} \wedge H(\bar{m})) \wedge (\check{V}(\bar{\epsilon}, \acute{p})),\end{aligned}$$

and thus,

$$0 \leftarrow (\bar{\varsigma} \wedge H(\bar{m})) \wedge (\check{V}(\bar{\epsilon}, \acute{p})) \wedge (\varsigma \wedge H(\bar{m})),$$

which means the received data $\bar{\bar{m}}$ are forged and will not be stored in the local chain. Thus, the collected data by the monitor during the off-chain transmission is secure and cannot be tampered with.

The proof of the theorem is complete. □

Theorem 2. During the off-chain sharing stage, the secure sharings mechanism in Algorithm 3 guarantees the following conditions: the data requested by the user is secure, forged data is discarded, and unauthorized users cannot access the data on the blockchain, which are formulated as:

$$\begin{aligned}&\forall\ m, \bar{m}, \bar{\epsilon},\\ &1 \leftarrow (\varsigma \wedge H(m)) \wedge (\check{V}(\epsilon, p)),\\ &0 \leftarrow (\varsigma \wedge H(\bar{m})) \wedge (\check{V}(\epsilon, p)),\\ &0 \leftarrow (\varsigma \wedge H(m)) \wedge (\check{V}(\bar{\epsilon}, p)).\end{aligned}$$

Proof. According to Theorem 1, we have:

$$1 \leftarrow (\varsigma \wedge H(m)) \wedge (\check{V}(\epsilon, p)),$$

And according to Algorithms 2 and 3, when the attacker forges the message $\bar{m}$, we have:

$$\varsigma' \leftarrow H(\bar{m}), 0 \leftarrow \varsigma \wedge \varsigma',$$

and thus,

$$0 \leftarrow (\varsigma \wedge H(\bar{m})) \wedge (\check{V}(\epsilon, p)).$$

Thus, the data $\bar{m}$ will be considered forged and discarded, and the data requested by the user is secure.

According to Algorithm 3, when an unauthorized user tries to forge a signature,

$$\bar{\epsilon} \leftarrow S(\langle t, p_u\rangle, \bar{k}),$$

and as for an illegitimate signature $\bar{\epsilon}$, due to the lack of the private key of the user and RSU, we have:

$$\begin{aligned}0 \leftarrow & \check{V}(\bar{\epsilon}, p),\\ 0 \leftarrow & (\varsigma \wedge H(m) \wedge (\check{V}(\bar{\epsilon}, p)).\end{aligned}$$

Thus, the unauthorized user cannot access the data on the blockchain.

The proof of the theorem is complete. □

Algorithms 2 and 3 form an identification mechanism of GCS proved by Theorems 1 and 2 to demonstrate data security when there are attackers and unauthorized users during the off-chain transmission. Theorem 1 proves the data security when data are collected and transmitted to the local chain for storage, while Theorem 2 proves the security of off-chain sharings among users and RSUs.

Resiliency analysis:

i) Consensus resiliency (Committee resiliency). The shard always generates valid blocks when the Byzantine nodes are less than $1/3$ within the consensus group.

According to Algorithm 2 and Theorem 1, the data transferred by the monitors are accurate. Meanwhile, according to the PBFT algorithm, the consensus always succeeds when the Byzantine nodes within the consensus group are less than $1/3$. Thus, the block generated by graphical consensus is always valid.

ii) Shard resiliency (Total resiliency). The blocks on the local chain are always valid when the Byzantine nodes controlled by the attackers within each shard are less than $1/2$.

When an honest node accepts to synchronize a newly generated block, there must be a valid accept message with a proof signature. However, the attacker cannot forge a new block that is accepted by the shard, because he can not obtain enough accept messages from the honest nodes in the committee. Meanwhile, if the attackers try to forge a sequence of blocks and fork the local chain, it will be regarded as a fork and discarded. This is because there are no more than $1/2$ Byzantine nodes to vote for the forged chain, and the honest nodes always vote on the correct blocks and add new blocks behind them. Therefore, the forged chain cannot obtain more than $1/2$ votes, and it is shorter than the local chain. Thus, the blocks on the local chain are valid and cannot be forged by attackers.

Complexity analysis:

Without loss of generality, we calculate the complexity of consensus per transaction. When a user initiates a transaction request, it will be contained in a block of size b. At the beginning of each epoch, the consensus group is selected within each shard through the graphical consensus. This imposes a communication and computation overhead of $\mathcal{O}(\frac{n}{b})$. After receiving a transaction request by the leader, the PBFT consensus will be initiated to verify it within the consensus

group of size m, which requires $\mathcal{O}(\frac{m^2}{b})$ communication overhead. Thus, the total complexity for each transaction equals $\mathcal{O}(\frac{n+m^2}{b})$.

IV. Experiments and Evaluation

In this section, we first evaluate the TPS and PT in general conditions when transaction generated speed (TGS) varies from 500 Tx/s to 1000 Tx/s, to show the advantages of sharding. Secondly, we simulate the change in TPS and PT when the shard number ranges from 4 to 14 and TGS varies from 1050 Tx/s to 6000 Tx/s. Meanwhile, GCS is also compared with some available mainstream sharding solutions. Finally, the FP of GCS and randomness sharding frameworks [6], [7] are also compared, and the change in PT is observed when the shard failure occurs.

A. Experimental settings

Baselines:

We compare the proposed method with the following baselines:

- **Non-sharding with PBFT**: THis is a non-sharding blockchain architecture using the PBFT consensus algorithm [38], which is commonly used in BIoVs.
- **Elastico** [6]: This is the first proposed sharding approach that scales transaction rates linearly with computation.
- **OmniLedger** [33]: This is considered to be the improved edition of Elastico, which increases data security while improving throughput.
- **RapidChain** [7]: This is the first sharding-based public blockchain that achieves complete sharding of the communication, computation, and storage overhead.
- **Zilliqa** [34]: This is considered to be a unique blockchain sharding platform designed to scale in the open, permissionless distributed network securely.

Metrics:

The common metrics used to evaluate the GCS are:

1) TPS: represents the average number of transactions processed each second over a period of time, and is calculated as:

$$\text{TPS} = \frac{\sum_{i=t_1}^{t_2} \mathbb{X}_i}{t_2 - t_1}. \tag{11}$$

2) PT: indicates the average time spent on a transaction from sending to packing into a block, and is computed as:

$$\text{PT} = \frac{\sum_{i=1}^{m} \sum_{j=1}^{n} T_b^i - T_x^j}{\sum_{k=1}^{T_b^m} \mathbb{X}_k}. \tag{12}$$

3) FP: indicates the probability of shard failure. In the shard systems with randomness allocation [6], [33], it is calculated as in Eq. (13). And the failure probability of GCS is observed by 1000 times of simulations.

$$\text{FP} = \sum_{b=\frac{N}{3}}^{N} \binom{N}{b} f^b (1-f)^{N-b}, \tag{13}$$

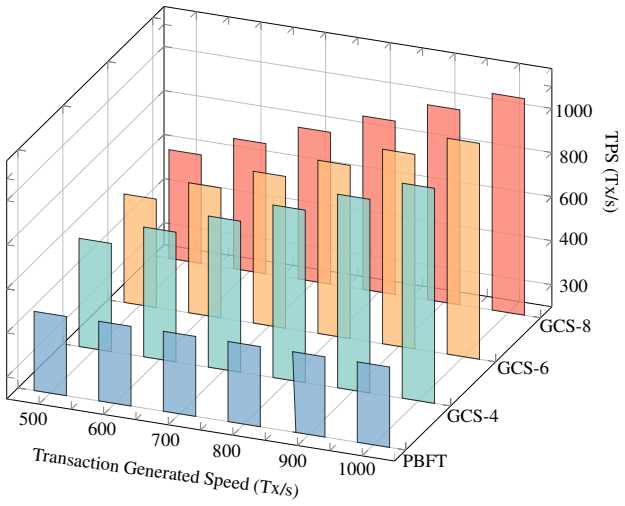
where $\mathbb{X}_i$ represents the set of transactions at time i, T_b and T_x is the timestamp of blocks and transactions, N represents the number of nodes, f represents the proportion of Byzantine nodes, respectively.

Parameter Settings:

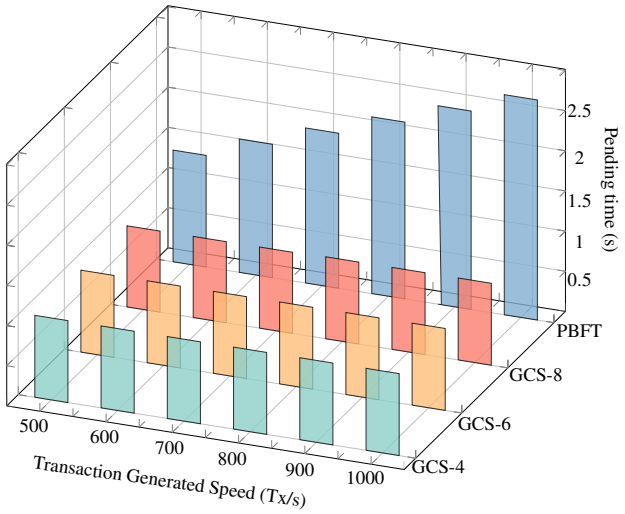
The simulation has been conducted on a Dell workstation with eight cores, 64Gb RAM, and an 11G Nvidia 1080Ti GPU. The whole GCS approach is implemented using NS-3 [47]. The inter-node bandwidth is set to 4M, and the network latency is 40 ms. The TGS is set from 500 Tx/s to 6000 Tx/s. And the average accuracy of the temporal-spatial graph convolutional network prediction model is 0.911 after 100 times of training on the Sz-taxi dataset.

B. General performance

We compared GCS-4, GCS-6, GCS-8, and GCS-12 (GCS-X means there are X shards in GCS, i.e., GCS-4 means the number of shards is 4) with non-sharding blockchain (legend with PBFT), where TGS rises from 500 Tx/s to 4000 Tx/s.



(a) TPS of PBFT and GCS with low TGS

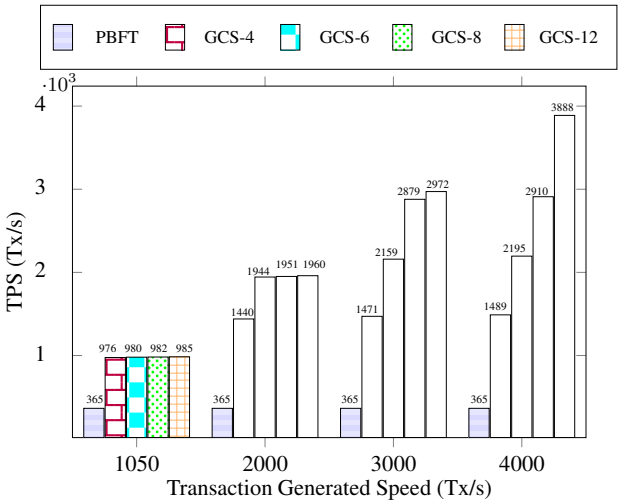


(b) PT of PBFT and GCS with low TGS

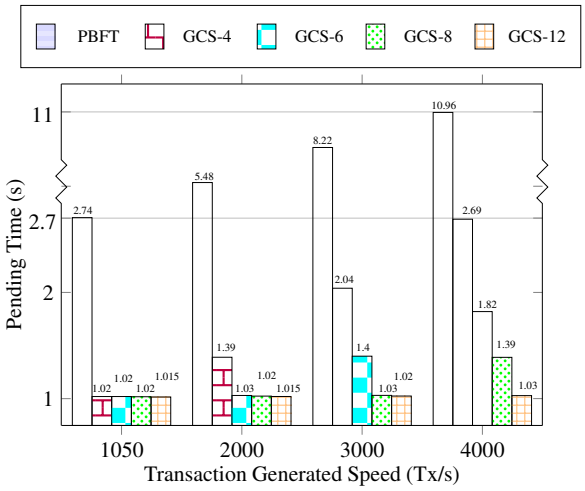
Fig. 5. Comparisons between PBFT and GCS with low TGS

When the TGS rises from 500 Tx/s to 1000 Tx/s, the simulation results are shown in Fig. 5. It can be seen in Fig. 5(a), the TPS of PBFT is about 360 Tx/s whereas the

TPS of GCS rises from 492.8 Tx/s to 982.5 Tx/s, which has dramatically improved. The TPS of GCS-8 can reach 269% of PBFT. And according to Fig. 5(b), the PT of the proposed GCS is 1.02 seconds, and the maximum PT of PBFT is more than 2.74 seconds which is intolerable in a latency-sensitive IoV system.



(a) TPS of PBFT and GCS with high TGS

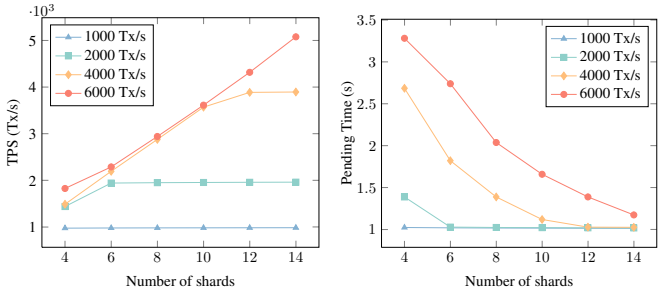


(b) PT of PBFT and GCS with high TGS

Fig. 6. Comparisons between PBFT and GCS with high TGS

When the TGS>1000 Tx/s, the TPS and PT comparisons are shown in Fig. 6. As illustrated in Fig. 6(a), when the TGS≤2000 Tx/s, the difference of TPS between GCS-X is minimal because the load on these shards has not reached the maximum at this time. And as the TGS increases, TPS starts to show a discrepancy. Eventually, when the TGS=4000 Tx/s with a total of 20,000 transactions, the TPS of GCS-12 is 3888 Tx/s, and the TPS of PBFT remains 360 Tx/s. The PT of PBFT is 10.96s at this time and a transaction in the non-sharding BIoVs will wait up to 10.96s to be packed into a block, whereas in GCS, the time is about 1s. In other words, the improved TPS and less PT of GCS can cope with the increasing number of users' requests and meet the demand of low latency in BIoVs.

Fig. 7 illustrates the tendency of TPS and PT for different



(a) TPS comparison (b) PT comparison

Fig. 7. TPS and pending time comparisons with increasing TGS

numbers of shards in GCS as the TGS increases. As shown in Fig. 7(a), when the TGS=1050 Tx/s, the TPS stabilizes at around 1000 Tx/s, because shards are not running at full load. When TGS=2000 Tx/s, the increase in TPS is minor for GCS-4 compared to GCS-8 to GCS-12 because the shards in GCS-4 are already working at full load while others are not. The reason for the variation in trend is also same when TGS=4000 Tx/s and 6000 Tx/s.

In Fig. 7(b), when the TGS=1050 Tx/s, GCS can pack all the transactions in 1 second. When the TGS=2000 Tx/s, in GCS-4, transactions need to wait for 1.4s to be packaged due to too many transactions and limited processing capacity, while others still take only 1 second to pack all transactions. So does it while the TGS=4000 Tx/s.

The TPS comparisons of PBFT and GCS are shown in Table III. As TGS increases, the TPS of PBFT is maintained at 365 Tx/s due to the limit of throughput, and it cannot process such a volume of users' requests. When TGS≤1000 Tx/s, the TPS of GSC-4 to GCS-12 has a minor difference because all the shards are running at a low load. In particular, when TGS comes to 2000 Tx/s, the TPS of GCS-4 is lower than GCS-6 to GCS-12 because GCS-4 is already running at full load while others are not. When we set TGS to 4000Tx/s, we can obtain that the TPS of the proposed GCS rises as the number of shards increases.

The comparisons of mainstream sharding blockchain are shown in Table IV. The PT of GCS has a significant reduction, which is at least 1.02s. Note that when there are few shards and a high TGS, the PT will increase, for example, the PT of GCS-4 is 3.4s when TGS=6000 Tx/s. We measure the TPS of GCS-12 when TGS=4000 Tx/s, which is 3888 Tx/s, and this increases with the rise of the number of shards and TGS.

TABLE III
TPS COMPARISONS

Method	Low TGS						High TGS		
	500	600	700	800	900	1000	2000	3000	4000
PBFT	360	361	363	364	365	365	365	365	365
GCS-4	491	589	686	783	879	976	1440	1471	1489
GCS-6	492	590	689	786	883	980	1944	2159	2195
GCS-8	493	591	689	787	885	983	1951	2879	2910
GCS-10	495	591	689	788	885	986	1957	2920	3430
GCS-12	495	592	689	788	890	986	1960	2972	3888

TABLE IV
SHARDING BLOCKCHAIN COMPARISONS

Sharding blockchain	Total resiliency	Committee resiliency	TPS	Pending time	Sharding scheme	Leader selection	Blockchain structure
Elastico [6]	1/4	1/3	40	800s	Random	EpochRandomness	Chain
OmniLedger [33]	1/4	1/3	500	14s	Random	RandHound	State block+Chain
RapidChain [7]	1/3	1/2	4220	8.5s	Random	Randomness	Chain
Zilliqa [34]	1/4	1/3	2488	10s	Election	Node number	Chain
GCS (ours)	1/2	1/3	3888	1.02s	Area-based	Reliability weights	Block-based DAG

C. Performance with Byzantine nodes

Fig. 8 shows the comparisons of FP between randomness sharding (legend with RandS) and GCS, where GCS (60) means that the number of nodes in each shard is 60, etc. As the proportion of Byzantine nodes (f) increases, the FP also increases. In particular, when $f \leq 18\%$, the FP of GCS (60) is slightly higher than that of RandS, which is about 2×10^{-3} and negligible. When $f = 19\%$, the FP of RandS (60) reaches 6×10^{-3}, which is twice of that in GCS (60). And when $f = 28\%$, the FP of RandS (60) has already exceeded 0.2, which is 34 times increase over that at $f = 19\%$, while the FP of GCS (60) rises to 4×10^{-3}.

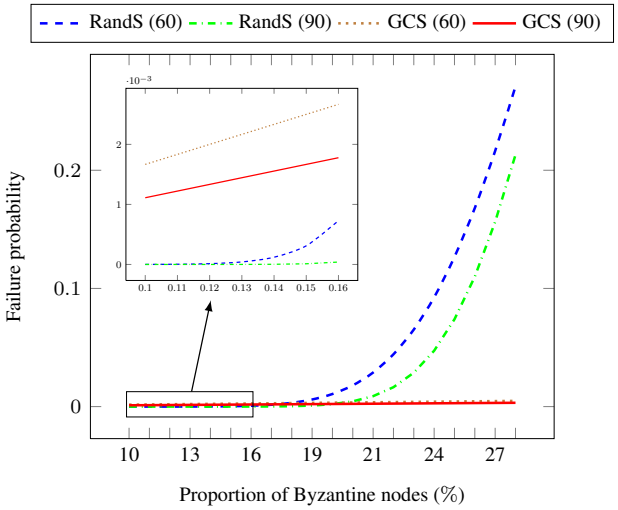
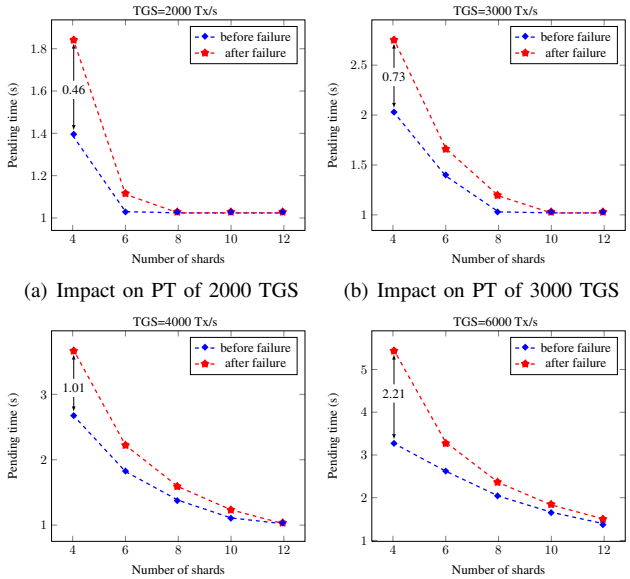


Fig. 8. Failure probability of GCS and common sharding.

Fig. 9 shows the impact on PT before and after shard failure. As illustrated in Fig. 9(a), when the TGS is 2000 Tx/s, the impact of GCS-4 and GCS-6 are slightly higher, which are 0.46s and 0.08s, respectively. From GCS-8 to GCS-12, even if a shard fails, the transactions can be processed by the backup shard in time. And when TGS=3000 Tx/s, as shown in Fig. 9(b), the PT of GCS-4 increases more than that in 2000 TGS, which is 0.73s. As for GCS-6 to GCS-10, the PT has a small increase, respectively, and these shards work at full load. When TGS comes to 4000 Tx/s and 6000 Tx/s, the impact on PT is 1.01s and 2.21s, respectively, shown in Figs. 9(c) and 9(d). However, in other sharding approaches, users' requests will remain pending until the failed shard is recovered.



(a) Impact on PT of 2000 TGS (b) Impact on PT of 3000 TGS

(c) Impact on PT of 4000 TGS (d) Impact on PT of 6000 TGS

Fig. 9. Pending time comparisons before and after shard failure.

V. CONCLUSIONS AND FUTURE WORK

In this paper, we proposed a novel graphical consensus-based sharding framework, called GCS, for efficient and secure sharings in BIoVs. GCS employs the blockchain sharding to enhance the TPS and overcome the bottleneck of data sharing, as BIoVs expands. Firstly, the graphical consensus is introduced as the intra-shard consensus, which improves the throughput of BIoVs and enhances the consensus efficiency. Secondly, we optimized the sharding blockchain design. The local chain stores the intra-shard data and is used to respond to users' requests within each shard, while a block-based DAG, rather than a chained structure, is employed as the main chain for cross-shard sharings. Then, node scheduling and shard backup were introduced to handle the shard overheating and failure. Finally, an off-chain transmission algorithm was proposed for secure sharing between the infrastructure and the vehicles.

The security of GCS has been theoretically proven, and the simulation results show that our approach achieves higher TPS and lower pending time, which will match the demand for the low-latency scenario in BIoVs. In particular, GCS has been shown to achieve a TPS of 983 Tx/s (GCS-8, TGS=1000) and 3888 Tx/s (GCS-12, TGS=4000), which is 1.62 times and 36% higher compared with the non-sharding blockchain and most sharding blockchain approaches, respectively. In terms

of pending time, the average transaction pending time is at least 0.38s shorter (GCS-4, TGS=500) than the non-sharding blockchain. Meanwhile, when facing the Byzantine attack, the pending time of GCS has a minor increment, which is 0.46s (GCS-4, TGS=2000) and 1.01s (GCS-4, TGS=4000), respectively.

Although the PBFT consensus mechanism can secure the final consistency of the local chain data, its unique three-stage process will take up some network traffic, resulting in additional communication overhead. Meanwhile, due to the adoption of the shard backup, each node in the shard will store more data. In particular, extra storage space will be spent on historical blocks. Future work will focus on developing the state-less sharding architecture that allows the nodes to verify transactions without storing the full state. This will reduce communication volume during the consensus and the nodes' storage requirements. Our GCS implementation motivates new research on improving the performance of blockchain and, in particular, on better usage of the blockchain technology in IoVs.

ACKNOWLEDGMENTS

The authors would like to thank Prof. Muhammad Ismail for his kind help and suggestions. The authors also sincerely thank the anonymous reviewers for their very helpful suggestions that have helped improve the presentation of this paper.

REFERENCES

[1] W. Zhuang, Q. Ye, F. Lyu, N. Cheng, and J. Ren, "SDN/NFV-Empowered future IoV with enhanced communication, computing, and caching," *Proceedings of the IEEE*, vol. 108, no. 2, pp. 274–291, 2020.

[2] S. Lee and S.-H. Seo, "Design of a two layered blockchain-based reputation system in vehicular networks," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 2, pp. 1209–1223, 2022.

[3] J. Wang, J. Huang, L. Kong, G. Chen, D. Zhou, and J. J. C. Rodrigues, "A privacy-preserving vehicular data sharing framework atop multi-sharding blockchain," in *2021 IEEE Global Communications Conference (GLOBECOM)*, 2021, pp. 1–6.

[4] F. Ayaz, Z. Sheng, D. Tian, and Y. L. Guan, "A blockchain based federated learning for message dissemination in vehicular networks," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 2, pp. 1927–1940, 2022.

[5] J. Xie, K. Zhang, Y. Lu, and Y. Zhang, "Resource-efficient DAG blockchain with sharding for 6G networks," *IEEE Network*, vol. 36, no. 1, pp. 189–196, 2022.

[6] L. Luu, V. Narayanan, C. Zheng, K. Baweja, S. Gilbert, and P. Saxena, "A secure sharding protocol for open blockchains," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, 2016, pp. 17–30.

[7] M. Zamani, M. Movahedi, and M. Raykova, "RapidChain: Scaling blockchain via full sharding," in *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, 2018, pp. 931–948.

[8] C. Li, H. Huang, Y. Zhao, X. Peng, R. Yang, Z. Zheng, and S. Guo, "Achieving scalability and load balance across blockchain shards for state sharding," in *2022 41st International Symposium on Reliable Distributed Systems (SRDS)*, 2022, pp. 284–294.

[9] X. Zhang, R. Li, and H. Zhao, "A parallel consensus mechanism using PBFT based on DAG-Lattice structure in the Internet of Vehicles," *IEEE Internet of Things Journal*, 2022.

[10] C. Tang, K. Yu, K. Veeraraghavan, J. Kaldor, S. Michelson, T. Kooburat, A. Anbudurai, M. Clark, K. Gogia, L. Cheng *et al.*, "Twine: A unified cluster management system for shared infrastructure," in *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)*, 2020, pp. 787–803.

[11] C.-F. Cheng, G. Srivastava, J. C.-W. Lin, and Y.-C. Lin, "Fault-tolerance mechanisms for software-defined Internet of Vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 6, pp. 3859–3868, 2021.

[12] S. H. Alsamhi, A. V. Shvetsov, S. V. Shvetsova, A. Hawbani, M. Guizan, M. A. Alhartomi, and O. Ma, "Blockchain-empowered security and energy efficiency of drone swarm consensus for environment exploration," *IEEE Transactions on Green Communications and Networking*, 2022.

[13] X. Cai, S. Geng, J. Zhang, D. Wu, Z. Cui, W. Zhang, and J. Chen, "A sharding scheme-based many-objective optimization algorithm for enhancing security in blockchain-enabled Industrial Internet of Things," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 11, pp. 7650–7658, 2021.

[14] F. Li, K.-Y. Lam, Z. Ni, D. Niyato, X. Liu, and L. Wang, "Cognitive carrier resource optimization for Internet-of-Vehicles in 5G-enhanced smart cities," *IEEE Network*, vol. 36, no. 1, pp. 174–180, 2022.

[15] N. Deepa, Q.-V. Pham, D. C. Nguyen, S. Bhattacharya, B. Prabadevi, T. R. Gadekallu, P. K. R. Maddikunta, F. Fang, and P. N. Pathirana, "A survey on blockchain for big data: Approaches, opportunities, and future directions," *Future Generation Computer Systems*, vol. 131, pp. 209–226, 2022.

[16] J. Li, G. Luo, N. Cheng, Q. Yuan, Z. Wu, S. Gao, and Z. Liu, "An end-to-end load balancer based on deep learning for vehicular network traffic control," *IEEE Internet of Things Journal*, vol. 6, no. 1, pp. 953–966, 2018.

[17] C. Liu, H. Guo, M. Xu, S. Wang, D. Yu, J. Yu, and X. Cheng, "Extending on-chain trust to off-chain–trustworthy blockchain data collection using trusted execution environment (tee)," *IEEE Transactions on Computers*, 2022.

[18] P. Zhang, M. Zhou, J. Zhen, and J. Zhang, "Enhancing scalability of trusted blockchains through optimal sharding," in *2021 IEEE International Conference on Smart Data Services (SMDS)*, 2021, pp. 226–233.

[19] B. David, B. Magri, C. Matt, J. B. Nielsen, and D. Tschudi, "GearBox: Optimal-size shard committees by leveraging the safety-liveness dichotomy," in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, 2022, pp. 683–696.

[20] H. Guo, W. Li, and M. Nejad, "A hierarchical and location-aware consensus protocol for IoT-blockchain applications," *IEEE Transactions on Network and Service Management*, 2022.

[21] Z. Su, Y. Wang, Q. Xu, and N. Zhang, "LVBS: Lightweight vehicular blockchain for secure data sharing in disaster rescue," *IEEE Transactions on Dependable and Secure Computing*, vol. 19, no. 1, pp. 19–32, 2022.

[22] R. Friedman, A. Mostefaoui, and M. Raynal, "Simple and efficient oracle-based consensus protocols for asynchronous Byzantine systems," *IEEE Transactions on Dependable and Secure Computing*, vol. 2, no. 1, pp. 46–56, 2005.

[23] L. Vishwakarma, A. Nahar, and D. Das, "LBSV: Lightweight blockchain security protocol for secure storage and communication in SDN-enabled IoV," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 6, pp. 5983–5994, 2022.

[24] J. Mišić, V. B. Mišić, and X. Chang, "Design of proof-of-stake PBFT algorithm for IoT environments," *IEEE Transactions on Vehicular Technology*, pp. 1–14, 2022.

[25] F. Lin, S. Xia, J. Qi, C. Tang, Z. Zheng, and X. Yu, "A parking sharing network over blockchain with proof-of-planned-behavior consensus protocol," *IEEE Transactions on Vehicular Technology*, vol. 71, no. 8, pp. 8124–8136, 2022.

[26] J. Gao, K. O.-B. Obour Agyekum, E. B. Sifah, K. N. Acheampong, Q. Xia, X. Du, M. Guizani, and H. Xia, "A blockchain-SDN-enabled internet of vehicles environment for fog computing and 5G networks," *IEEE Internet of Things Journal*, vol. 7, no. 5, pp. 4278–4291, 2020.

[27] S. Kudva, S. Badsha, S. Sengupta, I. Khalil, and A. Zomaya, "Towards secure and practical consensus for blockchain based VANET," *Information Sciences*, vol. 545, pp. 170–187, 2021.

[28] B. Hou, H. Zhu, Y. Xin, J. Wang, and Y. Yang, "MPoR: A modified consensus for blockchain-based Internet of Vehicles," *Wireless Communications and Mobile Computing*, vol. 2022, 2022.

[29] G. Xu, H. Bai, J. Xing, T. Luo, N. N. Xiong, X. Cheng, S. Liu, and X. Zheng, "SG-PBFT: A secure and highly efficient distributed blockchain PBFT consensus algorithm for intelligent Internet of Vehicles," *Journal of Parallel and Distributed Computing*, vol. 164, pp. 1–11, 2022.

[30] H. Chen and Y. Wang, "SSChain: A full sharding protocol for public blockchain without data migration overhead," *Pervasive and Mobile Computing*, vol. 59, p. 101055, 2019.

[31] G. Wang, Z. J. Shi, M. Nixon, and S. Han, "SoK: Sharding on blockchain," in *Proceedings of the 1st ACM Conference on Advances in Financial Technologies*, 2019, pp. 41–61.
[32] S. Li, M. Yu, C.-S. Yang, A. S. Avestimehr, S. Kannan, and P. Viswanath, "PolyShard: Coded sharding achieves linearly scaling efficiency and security simultaneously," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 249–261, 2020.
[33] E. Kokoris-Kogias, P. Jovanovic, L. Gasser, N. Gailly, E. Syta, and B. Ford, "OmniLedger: A secure, scale-out, decentralized ledger via sharding," in *2018 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2018, pp. 583–598.
[34] T. Zilliqa and P. Barrett, "The Zilliqa project: A secure, scalable blockchain platform," in *Zilliqa*, 2018, pp. 1–18.
[35] P. K. Singh, R. Singh, S. K. Nandi, K. Z. Ghafoor, D. B. Rawat, and S. Nandi, "Blockchain-based adaptive trust management in internet of vehicles using smart contract," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 6, pp. 3616–3630, 2021.
[36] X. Wen, Z. Guan, D. Li, H. Lyu, and H. Li, "A blockchain-based framework for information management in internet of vehicles," in *2021 8th IEEE International Conference on Cyber Security and Cloud Computing (CSCloud)/2021 7th IEEE International Conference on Edge Computing and Scalable Cloud (EdgeCom)*, 2021, pp. 18–23.
[37] Y. Li, X. Tao, X. Zhang, J. Xu, Y. Wang, and W. Xia, "A DAG-Based reputation mechanism for preventing peer disclosure in SIoV," *IEEE Internet of Things Journal*, vol. 9, no. 23, pp. 24 095–24 106, 2022.
[38] M. Castro and B. Liskov, "Practical Byzantine fault tolerance and proactive recovery," *ACM Transactions on Computer Systems (TOCS)*, vol. 20, no. 4, pp. 398–461, 2002.
[39] M. Conti, G. Kumar, P. Nerurkar, R. Saha, and L. Vigneri, "A survey on security challenges and solutions in the IOTA," *Journal of Network and Computer Applications*, p. 103383, 2022.
[40] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *5th International Conference on Learning Representations, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.
[41] L. Zhao, Y. Song, C. Zhang, Y. Liu, P. Wang, T. Lin, M. Deng, and H. Li, "T-GCN: A temporal graph convolutional network for traffic prediction," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 9, pp. 3848–3858, 2019.
[42] K. Cho, B. van Merrienboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder-decoder approaches," in *Proceedings of SSST@EMNLP 2014, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation, Doha, Qatar, 25 October 2014*. Association for Computational Linguistics, 2014, pp. 103–111.
[43] D. R. Brown, "Sec 2: Recommended elliptic curve domain parameters," *Standars for Efficient Cryptography*, 2010.
[44] N. Koblitz, "Elliptic curve cryptosystems," *Mathematics of Computation*, vol. 48, no. 177, pp. 203–209, 1987.
[45] V. S. Miller, "Use of elliptic curves in cryptography," in *Conference on The Theory and Application of Cryptographic Techniques*. Springer, 1985, pp. 417–426.
[46] A. K. Kasgar, J. Agrawal, and S. Shahu, "New modified 256-bit MD 5 algorithm with SHA compression function," *International Journal of Computer Applications*, vol. 42, no. 12, 2012.
[47] G. Carneiro, "NS-3: Network simulator 3," in *UTM Lab Meeting April*, vol. 20, 2010, pp. 4–5.

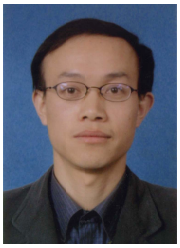
Wenqi Li received his M.S. degree from the College of Information Science and Technology, Dalian Maritime University, China, in 2020. He is curently pursuing a Ph.D. with the College of Artificial Intelligence, the Dalian Maritime University, China. His research interests include IoVs and blockchain technology.

Zheng Zhao received his B.S. degree from Dalian University of Technology, China, in 2010. He received his M.S. and Ph.D. degrees from the Zhengzhou Science and Technology Institute in 2013 and 2017, respectively. His research interests include the next generation Internet and deep learning.

Pingchuan Ma received his B.S. degree from the College of Computer and Information Engineering, Inner Mongolia Agricultural University, China, in 2020. He is pursuing a Ph.D. with the College of Artificial Intelligence, at the Dalian Maritime University, China. His research interests include UAVs and blockchain technology.

Zeqiang Xie received his B.S. degree from the Chongqing University of Posts and Telecommunications, China, in 2014. He received his M.S. and Ph.D. degrees from the University Putra Malaysia in 2017 and 2021, respectively. His research interests include blockchain and distributed computing.

Vasile Palade (Senior Member, IEEE) is a Professor of Artificial Intelligence and Data Science in the Centre for Computational Sciences and Mathematical Modelling at Coventry University, UK. He previously worked with the Department of Computer Science, University of Oxford, UK, and received his Ph.D. from the University of Galati, Romania, in 1999. His research interests are in the area of machine learning and applications, including deep learning and neural networks, various nature inspired optimization algorithms, computer vision, and natural language processing.



Hongbo Liu received his three level educations (B. Sc., M. Sc., Ph.D.) at the Dalian University of Technology, China. He is working with the College of Artificial Intelligence, Dalian Maritime University, with an affiliate appointment in the Institute for Neural Computation at the University of California San Diego, USA.

Professor Liu's research interests are in cognitive computing, machine learning, big data. He is regularly involved in organizing international conferences and workshops as well as an editor to reputed international journals. He received the New Century Excellent Talents Award in 2011.