electronics

MDPI

Article

Trust Management Scheme of IoV Based on Dynamic Sharding Blockchain †

Hongmu Han [1], Sheng Chen [1], Zhigang Xu [1,*], Xinhua Dong [1] and Jing Zeng [2]

[1] School of Computer Science, Hubei University of Technology, Wuhan 430068, China; hanhongmu@hbut.edu.cn (H.H.); 202111167@hbut.edu.cn (S.C.); xhdong@hbut.edu.cn (X.D.)
[2] China Gridcom Co., Ltd., Shenzhen 518031, China
* Correspondence: xzg@hbut.edu.cn
† This paper is an extended version of our paper published in Han, H.; Chen, S.; Xu, Z.; Dong, X.; Tian, W. GPChain: Optimizing Cross-Shard Transactions and Load Imbalance in Sharded Blockchain Networks. In Proceedings of the Internet of Things—ICIOT 2023, Shenzhen, China, 17–18 December 2023.

Abstract: With the rapid development of communication technologies, the demand for security and automation of driving has promoted the development of the Internet of Vehicles (IoV). The IoV aims to provide users with a safer, more comfortable, and more efficient driving experience. However, the current IoV also faces a series of potential security risks and privacy breaches, which has further propelled research on trust management for vehicular networks. The introduction of the blockchain has resolved the issue of data security in IoV trust management. However, the blockchain is limited by its own performance and scalability, making it unsuitable for large-scale networks. In order to enhance the transaction-processing efficiency of blockchain-based trust management solutions and address their scalability limitations, this paper presents a graph partition-based blockchain-sharding protocol. Simulation results on real-world datasets demonstrate that the proposed scheme exhibits better scalability compared to existing blockchain-based approaches and can accommodate larger-scale device access.

Keywords: IoV; blockchain; sharding; trust management; graph partitioning

Citation: Han, H.; Chen, S.; Xu, Z.; Dong, X.; Zeng, J. Trust Management Scheme of IoV Based on Dynamic Sharding Blockchain. *Electronics* **2024**, *13*, 1016. https://doi.org/10.3390/electronics13061016

Academic Editor: Zbigniew Kotulski

Received: 23 January 2024
Revised: 3 March 2024
Accepted: 5 March 2024
Published: 7 March 2024

1. Introduction

Vehicle-to-Everything (V2X) refers to a system that uses wireless communication technology and the Internet to interconnect vehicles, the road infrastructure, and traffic management centers. Within V2X, the Vehicular Ad-Hoc Network (VANET) is an important component that utilizes vehicle-to-vehicle and vehicle-to-road infrastructure communications to facilitate information exchange and the sharing of road conditions among vehicles. This aims to improve traffic safety, efficiency, and driving comfort. The VANET enables real-time data exchange, sharing of real-time road condition information, and cooperative driving among vehicles, providing new opportunities and challenges for the development of traffic management and intelligent transportation systems.

In the open, highly mobile, and dynamically changing topology of VANETs, security mechanisms based on authentication and access control cannot fully defend against attacks from authenticated and authorized internal nodes. Trust-management mechanisms, by establishing, maintaining, and verifying trust relationships between nodes, can reduce the risk of node interactions, improve system security and robustness, and complement authentication and access control. Traditionally, trust-management models are divided into three types: centralized, semi-centralized, and distributed. Most existing trust-management models use semi-centralized or distributed models because they can respond more quickly to changes in vehicle reputation than centralized models and can better address single point of failure issues.

The blockchain is a decentralized distributed ledger technology that ensures the security, immutability, and decentralization of data through encryption and distributed data storage. The core concepts of the blockchain include cryptography, consensus mechanisms, and smart contracts, and its emergence provides a new approach to solving data security and trustworthiness issues. The significant advantages of the blockchain, such as high security, decentralization, consistency, and reliability, make it the best distributed solution for establishing trust models between entities in the IoV.

However, existing blockchain-based trust-management solutions have focused more on how to calculate trust values more efficiently and accurately, while overlooking the impact of the blockchain's operational efficiency and scalability on the solutions. In fact, existing blockchains still face significant performance and scalability limitations before they can be widely applied. Although mainstream improvement solutions, such as blockchain sharding [1], can greatly enhance the theoretical performance of blockchains and offer excellent scalability, research has also discovered the issue of hot shards, which significantly reduces the actual throughput compared to the theoretical performance.

Hot shards [2] refer to the situation in a sharding blockchain where transactions involve parties that are unreasonably distributed across different shards, resulting in an excessive cross-shard transaction volume that blocks the normal operation [3] of the affected shards. This issue primarily arises from the power-law distribution of transaction addresses, where popular account addresses attract a large number of accounts from different shards to transact with them. Since transactions require confirmation from both shards involved, the shard containing the popular account experiences a longer authentication time, and other shards also need to wait for the same duration for confirmation from the hot shard. Consequently, network congestion occurs. This phenomenon creates a centralized network spontaneously formed by users through transactions in a decentralized network, significantly affecting the actual performance of sharded blockchains.

In this paper, we propose a graph partition-based blockchain-sharding protocol and employ it for evidence storage in trust management. By extracting the community graph [4] of trust evidence transactions stored on the chain and partitioning it accordingly, we achieve a more efficient transaction processing and reduce the load imbalance between partitions, thereby minimizing the occurrence of hot shards. The main contributions of this paper are as follows:

- We redefine the challenges of cross-shard transactions and the load imbalance caused by hot shards in the blockchain as a graph-partitioning problem. By utilizing our newly developed graph-partitioning algorithm, we are able to effectively address these challenges and achieve more balanced graph partitioning outcomes for power-law graphs, utilizing vertex-cut.
- In order to address the shortcomings of existing blockchains in transaction processing efficiency and scalability, we employ an enhanced graph-partitioning algorithm to optimize the sharding of the blockchain. This enables more efficient storage of trust transaction data and provides higher scalability, thereby accommodating larger-scale vehicle trust data.
- Through a comparison with other similar approaches on existing real-world datasets, our proposed solution demonstrates better performance in terms of accuracy, scalability, and various other aspect.

The remainder of this paper is organized as follows. Section 2 analyzes related work in the field. Section 3 introduces the system architecture. Section 4 describes the trust-management model. In Section 5, a graph partition-based blockchain-sharding protocol is proposed. Section 6 presents the simulation results. Finally, Section 7 provides a summary of this paper.

2. Related Work

2.1. Blockchain-Based Trust Management

Blockchain-Based Trust-Management models (BC-TMS) [5] typically utilize blockchain as a storage infrastructure for trust assets to achieve trust management. The consensus algorithm of the blockchain ensures the consistency of trust values across multiple regions, while the incentive mechanism encourages more vehicles to actively participate in VANET. In Ref. [6], a combination of Dirichlet distribution, reputation regression, and revocation punishment is proposed, showcasing improved trust management effectiveness compared to beta distributions. Ref. [7] considers direct and indirect feedback, along with additional factors, resulting in better identification outcomes. Refs. [8,9] focus more on blockchain-based trust management, with a distinction between Ref. [8] leaning towards incentive mechanisms and [9] emphasizing blockchain sharding schemes and smart contracts. It is evident that existing solutions primarily focus on accurate trust management and incentives, without addressing the performance metrics of the blockchain, such as TPS and scalability.

2.2. Blockchain Sharding

Numerous sharding solutions have been suggested by researchers to improve the scalability of blockchain systems. For instance, in Monoxide [10], a new relay transaction mechanism is employed to manage cross-shard transactions. Huang et al. [11] introduced a state-sharding protocol named Brokerchain, which accomplishes load balancing across shards by partitioning states and relaying transactions through brokers. Additionally, Xi et al. [12] proposed a dynamic sharding scheme based on a hidden Markov model, enabling adaptive updates of shards through finer-grained partitioning. Additionally, Xu et al. [13] put forward a clustering-based non-exhaustive genetic algorithm for solving the sharding problem. Furthermore, articles [14–16] optimized blockchain sharding by leveraging deep learning techniques to achieve enhanced performance. Finally, Cai et al. [17] presented a solution named Benzene, which reduces the cross-shard communication and storage overhead using a Trusted Execution Environment (TEE) and a dual-chain collaborative architecture. Current approaches primarily center on addressing sharding's scalability issue using machine learning techniques, often neglecting the fundamental factors contributing to the sharding problem.

2.3. Balanced Graph Partitioning

The objective of balanced graph partitioning is to achieve load balancing and minimize communication costs simultaneously. However, as this problem has been proven to be NP-hard, it is challenging to identify the optimal solution within a limited time frame. Graph-partitioning algorithms can be categorized into static and dynamic graph partitioning based on their running processes. Static graph-partitioning algorithms require access to the global data of the graph before partitioning, and typically employ spectral methods, heuristic methods, or multi-level partitioning methods to partition the graph data. For instance, the classic algorithm Metis [18] utilizes a multi-level partitioning strategy, while other methods, such as NE [4] and PowerLyra [19], use vertex-cut and hybrid-cut, respectively. By contrast, dynamic graph partitioning, also known as streaming graph-partitioning algorithms, does not require access to the global data of the graph before partitioning. Instead, it partitions the graph by continuously traversing the information of each edge (or vertex). Typical examples include Greedy [20], Grid [21], and HDRF [22]. Dynamic graph-partitioning methods offer better scalability and can accommodate larger-scale data. However, compared to static graph-partitioning methods, they typically exhibit lower accuracy. Both approaches have their respective strengths in handling different types of graph data. Previous research has indicated that for power-law distributed data, vertex-cut demonstrates better performance compared to edge-cut [22,23].

2.4. Cross-Shard Transaction

In sharded blockchains, maintaining the atomicity of transactions across shards is critical in scenarios where each shard does not possess the complete state. In Omniledger [24], the authors utilized a client-driven two-phase commit (2PC) mechanism with lock/unlock operations to ensure atomicity for cross-shard transactions. RapidChain [25] achieves atomicity by directing all involved UTXOs to the same shard, thereby converting cross-shard transactions into intra-shard transactions. In Monoxide [10], deduction and deposit operations are separately performed, and a relay transaction mechanism is employed to ensure the final atomicity for cross-shard transactions. Brokerchain [11] uses intermediary nodes called brokers to serve as relays and guarantors for cross-shard transactions. Additionally, in the S-store scheme [26], consistent hashing combined with an Aggregate Merkle B+ (ABM) tree methodology is proposed to reduce data migration between shards. It is evident that the focus of existing solutions is often centered on maintaining the transaction consistency between the involved shards, but the accounts of the entities involved are often overlooked.

3. Scheme Overview

3.1. System Model

As shown in Figure 1, to achieve rapid responsiveness to the Internet of Vehicles (IoV), the proposed blockchain-based trust-management model in this paper is divided into three layers: (1) a services layer, (2) a transactions layer, and (3) a VANET layer. The functions of each layer are as follows:

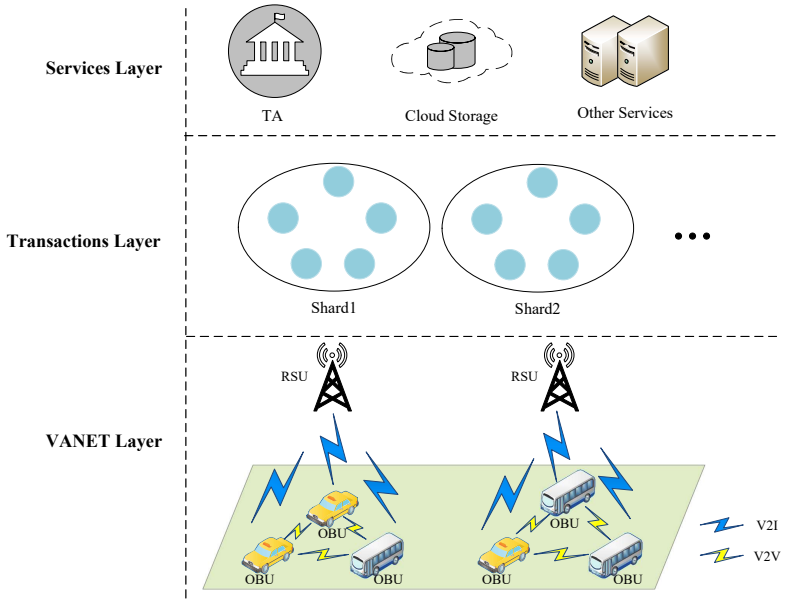

Figure 1. System model.

The services layer consists of entities such as the traffic authority (TA), certification authority (CA), cloud service storage, and service providers. The traffic authority serves as the supreme authority center responsible for registering and managing the authentication of all vehicle information. The CA is responsible for maintaining certificate information for vehicles in its domain, verifying the identities of users within the CA's domain, issuing digital certificates, and publishing certificate revocation lists (CRLs) or an online certificate status protocol (OCSP) to report revoked certificates. The cloud service provider offers users data storage and information services, such as real-time high-precision map navigation and media distribution services.

The transactions layer consists of sharded blockchains running on multi-access edge computing (MEC). It is responsible for storing the received VANET data on the blockchain

and providing tamper-proof and traceable data support for the services in the services layer. To ensure the scalability of the blockchain network, the blockchain is composed of multiple shards. When vehicle information is stored at the corresponding address on the blockchain, the data are saved within the corresponding shard. The detailed architecture will be described later.

The VANET layer consists of roadside units (RSUs) and vehicles. Generally, data are generated through vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communication and uploaded to the blockchain for storage. After the computation and verification of a vehicle's trust value, it is stored and managed by the shard where the vehicle is located, as well as the nearest RSU.

3.2. Sharding Blockchain

In this section, we will describe the sharding blockchain protocol, called the Graph Partition Chain (GPChain), designed for our trust management scheme. Serving as a sharding solution for the hot shards problem, this paper addresses the issue of hot shards caused by random node allocation and the power-law distribution of transactions by partitioning the transaction state graph, with the aim of increasing scalability. Considering the blockchain's operation in the context of the vehicular ad-hoc network environment, the proposed blockchain in this paper is based on a permissioned chain design. The symbols used in this paper are shown in abbreviation section. Similar to most state-sharding methods, an epoch is defined as a fixed period of blockchain operation, upon which the consensus and account reconfiguration processes are performed. All shards have the same structure and run the Practical Byzantine Fault Tolerance (PBFT) consensus protocol to handle shard transactions. The protocol conducts transaction packaging and consensus during the consensus period, partitions the account state graph, and achieves a consensus on the results during the account-reconfiguration period. The proposed solution, as illustrated in the Figure 2, describes three main stages of this process.

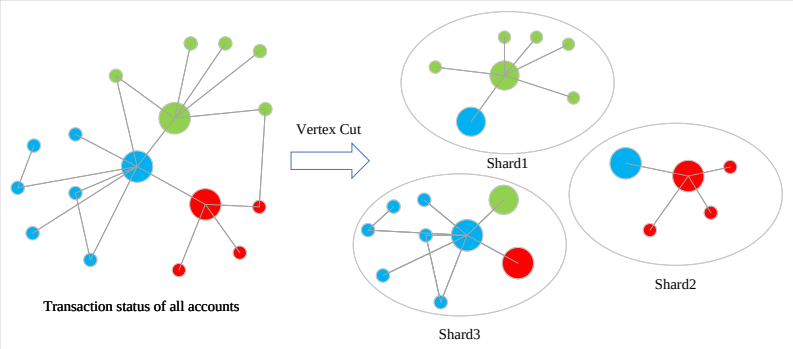

Figure 2. Process of GPChain.

Transaction State Graph Generation Phase: During the account reconfiguration period of each epoch, the primary node, P_{Shard}, from the previous round, which successfully achieved PBFT consensus within each shard, assumes the role of the primary node for that shard. It is tasked with gathering shard transactions for the current round and creating the sub-transaction state graph G' for that shard, where v represents the account address and e denotes a transaction between two accounts. Subsequently, the Raft consensus protocol selects a leader node, denoted as P_{Leader}, from all the P_{Shard} nodes in the shards. The shard where P_{Leader} is located is responsible for collecting the sub-transaction state graphs G' from each shard and creating the weighted undirected global state graph G for that epoch.

State Graph Partitioning Phase: The shard containing the P_{Leader} is responsible for partitioning the state graph. Initially, P_{Leader} loads the weighted undirected state graph and executes a vertex-cut algorithm based on community detection to divide transactions into various partitions, each representing a blockchain shard. Subsequently, the result of the partitioning for each account is determined according to its associated transactions

and the partition it is part of. In cases where a transaction linked to an account exists in multiple shards, duplicate accounts are created to ensure state synchronization across multiple shards.

State reconfiguration phase: Once the state graph-partitioning result is obtained, P_{Leader} shares it with other shards and attains consensus on the result using the Raft protocol. Subsequently, each shard configures its account information based on the partitioned state graph. As a new round of transactions commences, the transactions are assigned to the appropriate shard for processing.

4. Trust-Management Model Using Blockchain

The proposed blockchain-based trust-management model stores transaction evidence between vehicles on the blockchain. In order to derive the trust value of vehicles through this mutual evaluation evidence, we utilize the Dempster–Shafer theory (DST) to integrate the evidence and accurately evaluate the positive and negative behaviors of vehicles. The specific steps of the proposed method are as follows:

Step 1. Vehicle Joining VANET: When a vehicle, which has been authenticated by the TA, intends to join the VANET, it needs to apply for a certificate from the nearest CA. Once the CA verifies the vehicle's identity, it issues a digital certificate to the vehicle. The certificate includes the vehicle's public key, the owner's identity information, and the CA's digital signature. After obtaining the certificate, the vehicle registers its on-chain address in the shard nearest to it as its on-chain account, which will be used for subsequent data management.

Step 2. Evidence Collection: When a vehicle's onboard sensors detect unexpected events, such as traffic accidents or other anomalies on the road, the event, its location, and relevant information are packaged into a message block and broadcast to the RSU and nearby vehicles. Upon receiving the broadcast message, the RSU enters a fixed-duration waiting state, awaiting evidence from other vehicles that received the message within that time period as a response to participating in the event. Due to variations in sensor capabilities among different vehicles, the level of confirmation of the event may differ. To accurately describe the event, we define the evidence provided by vehicles regarding the event as $\theta = \{“True”, “False”, “TrueorFalse”, and\ “NULL”\}$, representing the degree of support for the vehicle's confirmation, disconfirmation, ambiguity, or unknown status of the event. For example, $\theta = \{“0.99”, “0.00”, “0.01”, “0.00”\}$ signifies that the vehicle believes there is a 99% probability that the event is true, with a 1% probability that both true and false are possible. After the waiting period for the RSU concludes, all vehicles that submit evidence are considered to have participated in the event, and the message block is subsequently uploaded to the blockchain.

Step 3. Event Inference: The process of fusing all the evidence for an event using DST [27] can be carried out by a smart contract running on the blockchain, ensuring transparency and reliability. This process aims to derive a more accurate description of the veracity of the event based on the fusion of evidence from all vehicles involved [28,29]. The fused result, similar to the aforementioned θ, represents the probability of the event being in different states, with the state having the highest probability considered as the current result of the event. In cases where the collected evidence is insufficient to calculate an accurate result, the event is deemed invalid.

Step 4. Trust Adjustment: After confirming the veracity of the event, the trust values of the vehicles uploading the event are adjusted based on the discrepancy between the provided evidence confidence and the actual event. For example, if the evidence provided by a vehicle leans towards supporting the event, its trust value is increased. Conversely, if the evidence contradicts the event, the trust value of the vehicle is decreased. If the evidence is uncertain or unknown, the trust value of the vehicle remains unchanged. Vehicles with higher trust values have their evidence treated as more valuable and given higher priority in the evidence-fusion process. Similarly, trust values that are too low can result in evidence being assigned lower priority. When a vehicle's trust value falls below a

certain threshold, its evidence is considered completely unreliable and will be removed from the vehicular network.

Step 5.Vehicle State Update: When a vehicle joins the blockchain, it is assigned to the shard where the nearest MEC resides. This allocation ensures faster response times for the vehicle and facilitates data exchange with other vehicles within the same shard. When a vehicle moves out of a shard, its account transfer to a new shard is determined based on the transaction state within the new shard, as dictated by the results of the graph partitioning on the blockchain, which will be described in the following section. For example, when a vehicle consistently conducts evidence transactions with vehicles located in another shard's region, it increases the probability of being allocated to that shard in the next round of graph partitioning. Conversely, in the absence of such partitioning, a shard has a higher probability of becoming a hot shard due to cross-shard transactions when a vehicle frequently provides evidence to vehicles in multiple shards. This process ensures that vehicles within each shard have closer transaction associations, enhancing the efficiency of data exchange.

5. Sharding-Based Blockchain Optimized through Graph Partitioning

This section presents a blockchain sharding scheme aimed at improving the performance and scalability of the sharded blockchain by addressing the issue of hot shards. The proposed approach utilizes community detection-based graph partitioning to partition the state graph of the transaction data, with the goal of reducing the occurrence of cross-shard transactions. Additionally, a more granular shard state tree is employed to establish a state-synchronization scheme to ensure the consistency of account states across shards. Ultimately, the blockchain will validate new transactions based on the reconfigured account state.

5.1. Community-Based Graph Partitioning

In existing state-sharding blockchains, the manner in which nodes are assigned to shards often overlooks node attributes, such as their degree information, which signifies the number of transactions they engage in with other nodes. Community detection, which aggregates nodes v in graph G into distinct communities based on modularity, offers a viable approach for considering node relationships. This aggregation can strengthen the connectivity within communities and weaken the connectivity between them. The Louvain algorithm [18] is a well-known community-detection method used to determine the community distribution in transaction graphs, grouping nodes with shared characteristics into the same community. When applied to blockchain transaction graphs, the aim is to reduce cross-shard transactions by identifying the same community for addresses with more shared transaction counterparts. While the Louvain algorithm appears effective for this purpose, it primarily partitions nodes based on account addresses, whereas our objective is to partition communities based on transaction edges. Prior studies have demonstrated that vertex-cut algorithms outperform edge-cut algorithms when handling power-law distributed data. Hence, by effectively combining node degree relationships with Louvain's community relationships, we have devised a greedy algorithm rooted in vertex cutting. When implemented to partition power law-distributed blockchain state graphs, this algorithm efficiently segments transaction-associated edges into communities within a finite time, as detailed in the Algorithm 1.

Algorithm 1: Community detection

Input: G_e : A set of transactions in epoch;
N : number of shards
Output: C_{Tx} : Community of TX

Function *community detection(G_e,N)*:
  $C \leftarrow Louvain(G_e)$
  //Divide the transaction graph into vertex set communities
  $C' \leftarrow$ Vector2Edge(C, G_e)
  Avg_c = total(G_e) $\div$ N //Calculate the average size of the divided community
  foreach $c \in C'$ **do**
    if *sum($c > Avg_c$)* **then**
      $C_{Tx} \leftarrow$ HDRF(c,total(G_e) $\div$ Avg_c)
      //Dividing Large Communities through HDRF
    else
      $C_{Tx} \leftarrow c$
    end
  end
  return C_{Tx}
end
Function *Vector2edge(C, G_e)*:
  foreach $Tx \in G_e$ **do**
    if *Two accounts belong to the same community* **then**
      Tx is divided into this community
      //Two accounts belonging to Tx are placed in the same community
    else
      Tx is divided into smaller communities
      //smaller community among the two accounts that Tx belongs to
    end
  end
  return C' //Convert Vertex communities to Edge communities
end

In our proposed solution, we initially utilize the Louvain algorithm to designate communities for each vertex. Subsequently, we iterate through each transaction in graph G and prioritize assigning the transaction to the community of the node with the lower degree among the transaction participants. This approach is aimed at minimizing the replication factor in the vertex-cut and ensuring that high-degree vertices possess fewer copies. Furthermore, to prevent excessively large communities, we utilize the High-Degree Replicated First (HDRF) algorithm to trim oversized communities and divide them into suitable sizes. The HDRF algorithm is tailored for power-law distributions and performs effectively in datasets featuring highly connected vertices. Through these techniques, we partition transactions into communities of diverse sizes, thereby reducing communication overhead between communities. Nonetheless, to fully implement our hybrid-cut algorithm, it is essential to distribute the communities across multiple shards. The greedy method, a widely applied heuristic algorithm known for its high accuracy and low computational cost, is adopted to allocate communities of edge sets to shards, as delineated in Algorithm 2.

Like typical greedy approaches, we arrange the communities by edge count and allocate them across multiple shards to reduce the imbalance in shard sizes. This method aims to attain balanced loads across shards. Ultimately, we can determine the shards to which each account is assigned based on the partitioned transactions in each shard.

Algorithm 2: Greedy partition

Input: C_{Tx} : Community of TX ;
N : number of shards
Output: N_i : set shard n of vector i
Function *greedy partition*(C_{Tx},N)**:**
  $C' \leftarrow$ Sort(C_{Tx})
  foreach $c \in C'$ **do**
    minLoadShard $\leftarrow c$
    // Allocate edge set c to the minimum load shard
  end
  $AvgLoad = totalLoad \div N$
  $diff = maxLoad - minLoad$
  while $diff > AvgLoad$ **do**
    $c_{min} \in maxLoad$
    $minLoad \leftarrow c_{min}$
    $diff = maxLoad - minLoad$
    //Move the smallest community in maxLoad to minLoad
  end
  return N_i
end

5.2. Account Status Reconfiguration

The sharding protocol divides the blockchain into multiple smaller shards, each capable of storing only the transaction blocks within its respective shard. In Ethereum, four primary tree structures are used to store different types of information, with the most important being the world state tree, which holds the current state of all accounts. In a state-sharding blockchain, the absence of cross-shard states prevents individual shard state trees from directly storing the states of vertices in other shards. Moreover, with each shard block requiring a state tree as the root hash of the state, it becomes impractical for all shards to collectively store the world state tree. Consequently, each shard is permitted to build its own shard state tree, maintaining the mapping of accounts to addresses with the addition of a mapping to indicate the associated shard for each account, as depicted in Figure 3.

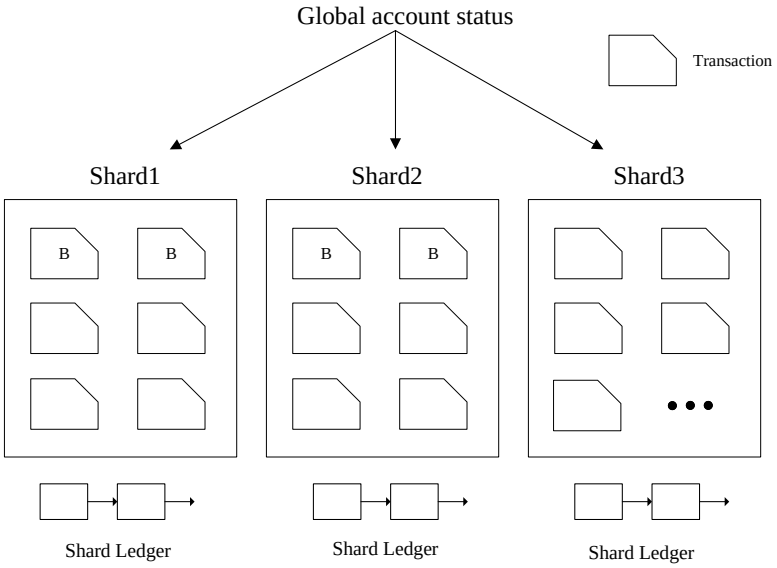

Figure 3. Mapping account addresses to multiple shards.

For instance, if account B is assigned to both shard1 and shard2, both shard1 and shard2 will have the state tree leaf vertex for account B, maintaining the balances B_{shard1} and B_{shard2} for account B within each shard. Transactions within a shard will only use the balances within that shard. When checking the total balance of account B, it can be obtained by querying the shard list where account B is located. Although it may appear as if account B has been split into two accounts, the shard allocation list for each account in each shard can easily retrieve the state of account B from all the shards it belongs to, achieving the final global state. After account partitioning, it is essential to allocate the balance owned by each account in each shard. Simply evenly distributing the balance among each shard may lead to some shards running out of account balances earlier due to varying numbers of transactions within each shard. Consequently, the allocation of account balances is determined by the following formula:

$$V_i = \frac{Tx_i}{Tx_{total}} \times V_{total} \tag{1}$$

The formula utilizes V_i to denote the balance of the current shard after partitioning, Tx_i for the shard's transaction volume during that round of partitioning, Tx_{total} for the total transaction volume related to the account, and V_{total} for the total balance of the account.

6. Simulation and Results

6.1. Environment and Settings

Experimental Prototype: In order to evaluate the proposed GPChain and trust model, we implemented a shard simulator and trust-management process using Python, and executed them on a local workstation. The workstation was equipped with two 2.4 GHz 20-core Intel Xeon E5 processors and 32 GB of memory. The shard simulator simulated the operation of the sharded blockchain by replaying the historical transaction data collected.

DataSet: The blockchain dataset used in our evaluation was collected from the Ethereum public transaction dataset, which includes approximately 500,000 transactions between block heights 5,181,526 and 5,184,886. The distribution of the dataset is shown in Figure 4. The dataset was replayed into the experimental prototype at a certain rate, and accounts were assigned to different shards based on their respective baseline scheme methods.

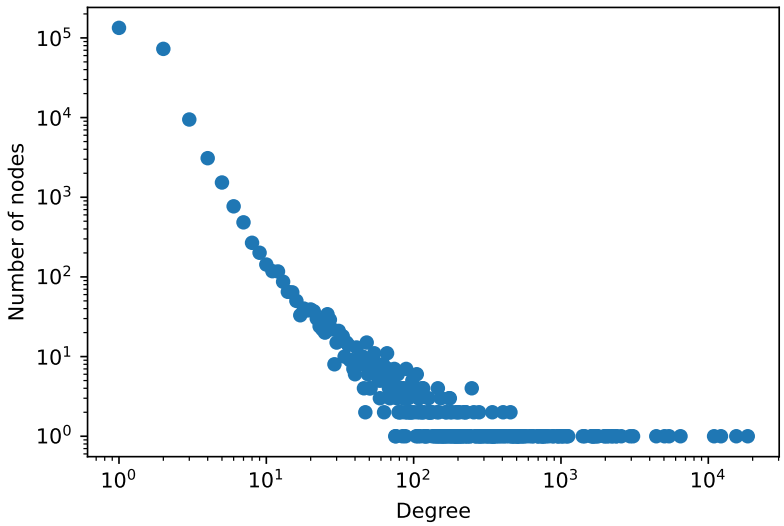

Figure 4. Distribution of dataset.

The evaluation of the trust management utilized the T-drive dataset [30,31], which includes the time, latitude, and longitude information of 10,357 taxis in the urban area of Beijing.

BaseLine: The blockchain considers three baseline scenarios: HDRF [22] represents a dynamical vertex cut scheme, Monoxide [10] represents a random node partition scheme, and Brokerchain [11] represents an edge cut followed by an account partitioning approach.The trust management compared the Dirichlet-Based Trust Management (DBTM) from paper [6], keeping all parameters consistent with those in the reference paper.

Metrics: We first conducted experiments on our proposed graph-partitioning algorithm and compared it with the existing vertex-cut HDRF algorithm, which is most suitable for power-law graphs. We compared the replication factor, load standard deviation, and runtime. Subsequently, we contrasted our approach with the bad reputation ratio of the DBTM. Finally, we compared the workload and load standard deviation of our proposed blockchain scheme with existing blockchain schemes. In all experiments, the replication factor represents the number of partitions, with lower values indicating better performance. The load standard deviation reflects the dispersion of the workload across different subsets, with lower values indicating lower dispersion.

6.2. Experimental

To begin with, we compared the vertex-cut approach proposed in this paper with the HDRF scheme, across various partition numbers and different dataset sizes, in terms of their replication factor, load-relative standard deviation, and runtime. The results are depicted in Figures 5–7:

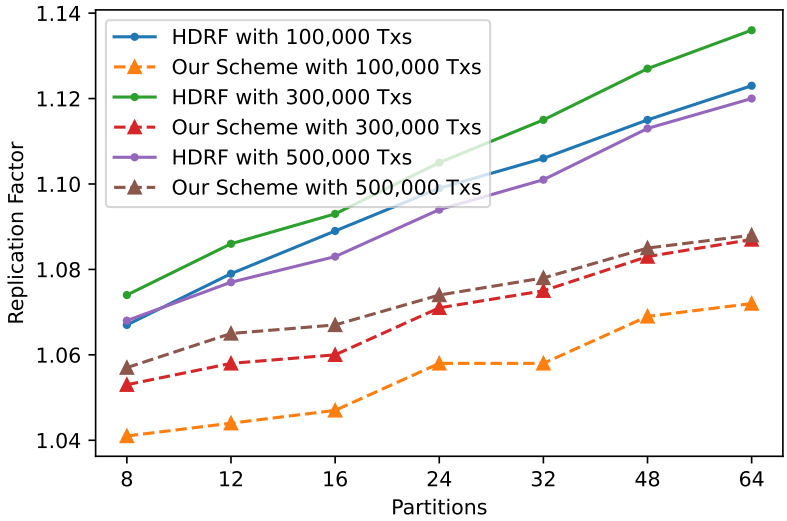

Figure 5. Replication factor of the two schemes.

Observing Figures 5 and 6, it is noticeable that an increase in the number of transactions led to a higher replication factor and standard deviation of the partition load. The enhanced vertex-cut algorithm, preserving community relationships between accounts, exhibited better performance with smaller partitions, while HDRF maintained a relatively stable replication factor across various partition sizes. In terms of the load standard deviation, HDRF's preference for partitioning strategies that minimize the replication factor is evident. However, in scenarios with a high number of duplicate transactions, the penalty term failed to effectively balance the partition load and replication factor. On the other hand, the approach proposed in this paper, involving the secondary partitioning of communities, achieved a more effective balance across multiple partitions.

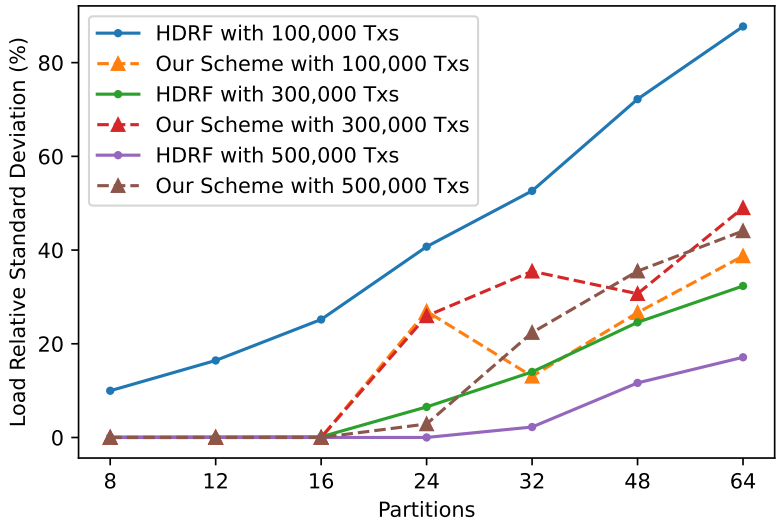

Figure 6. Load-relative standard deviation of the two schemes.

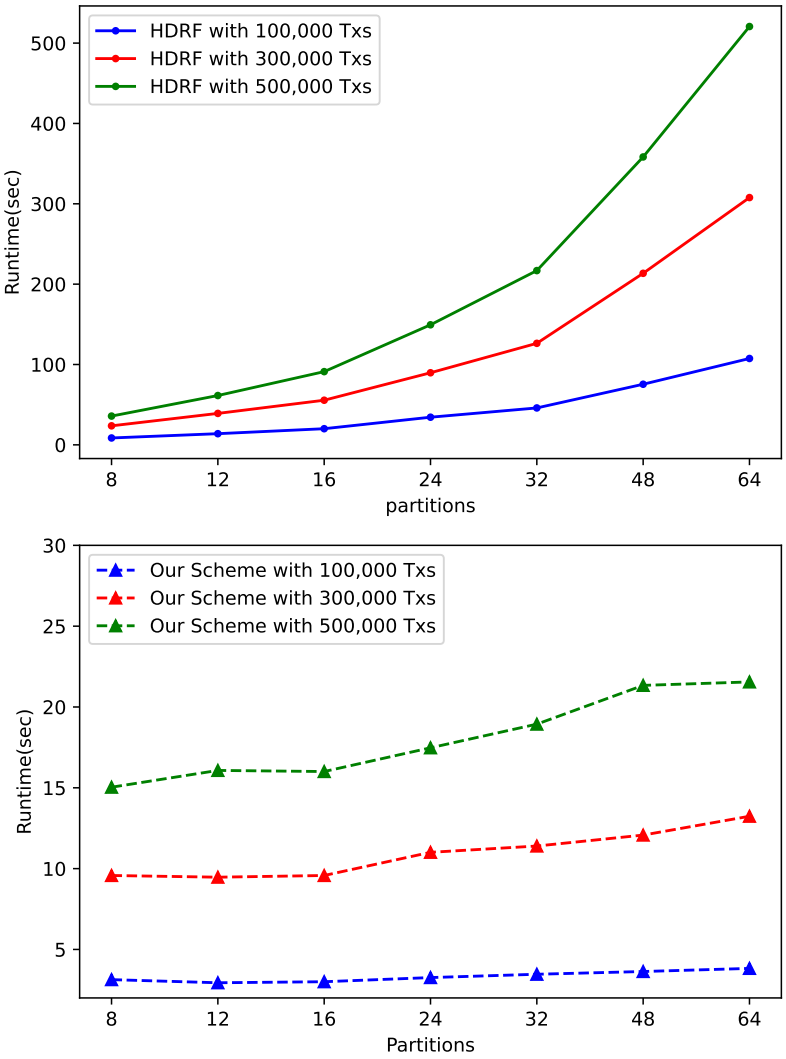

Figure 7. Runtime of the two schemes.

Figure 7 displays the disparity in the runtime performance of the two algorithms. It is evident that HDRF's runtime efficiency is more greatly influenced by the number of

partitions and the size of the dataset. This is due to the fact that our proposed approach reduces the data scale by performing community discovery prior to partitioning, whereas HDRF requires a matching process for each edge for the number of partitions, resulting in substantial overhead.

Subsequently, we compared the variation trends of negative reputations for normal and malicious vehicles in the trust-management model based on the DST theory proposed by us and the DBTM trust-management model under the same parameters, as shown in Figure 8.

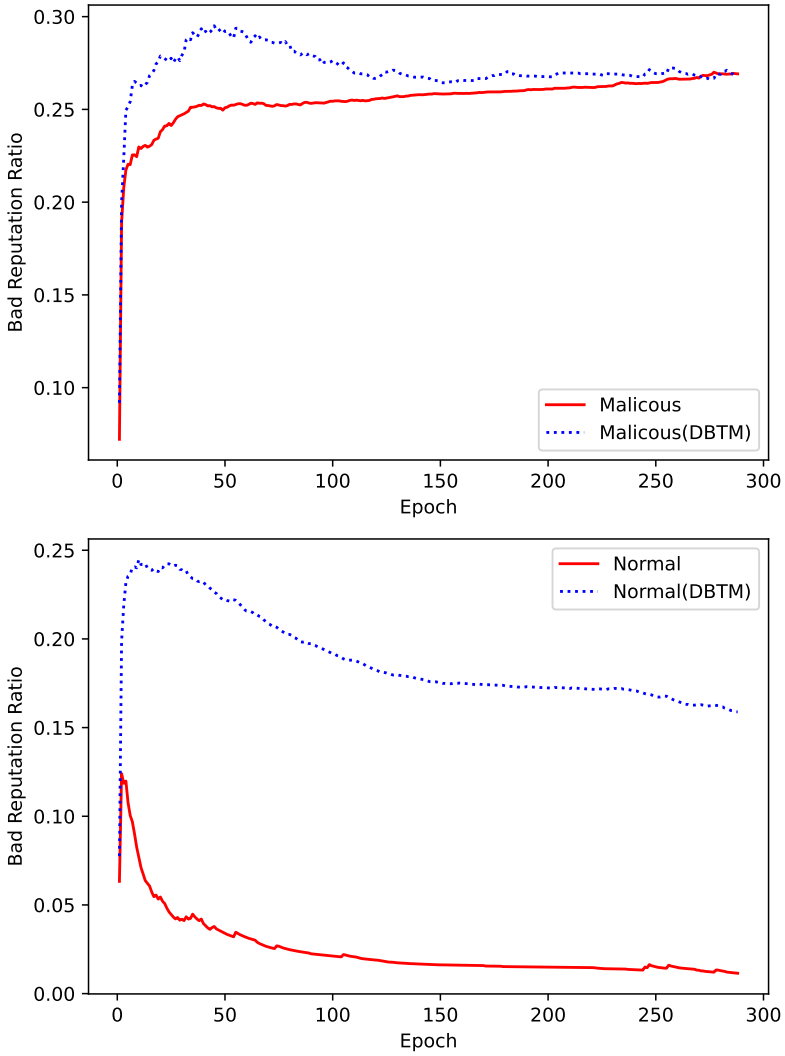

Figure 8. Bad reputation ratio of the two schemes.

It can be observed that with the increase in runtime, the negative reputation rates of malicious vehicles in both models will eventually stabilize within a certain range; however, in our proposed approach, normal vehicles attain lower negative reputation rates. This is attributed to the fact that the DST approach is able to integrate a relatively more accurate probability from multiple sources of evidence, indicating a lower probability of being falsely accused or attacked, and thus obtaining a negative reputation.

Finally, we partitioned the blockchain transaction dataset in chronological order into training and testing sets. We used the training set to obtain the properly partitioned shard states and then replayed varying sizes of the testing set into the shards to assess the performance of each approach.

The results are as follows. Figure 9 compares the proposed solution in this paper with existing sharding blockchain solutions in terms of cross-shard transaction rates. Our proposed solution exhibits the lowest cross-shard transaction rates.

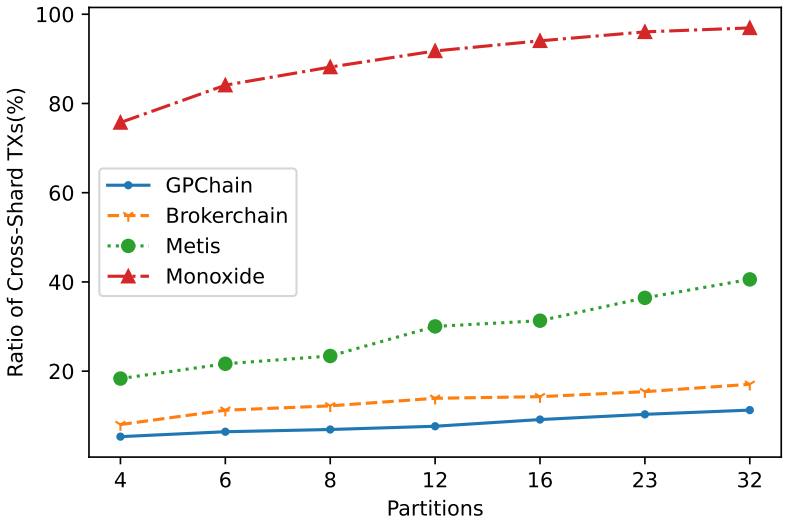

Figure 9. Cross-shard Txs ratio for different schemes.

Figures 10–12 present the differences in the workload and the relative standard deviation of the workload under different numbers of shards and data scales for different solutions. Table 1 contains more detailed information. It can be observed that GPChain demonstrated superior workload performance across various data scales and shards, with the relative standard deviation of the workload maintained at a lower level. This indicates that our proposed solution can accommodate a larger transaction volume compared to other solutions under similar hardware conditions. Monoxide, as a representative of a sharding blockchain without any optimization, exhibited the highest workload, but also the lowest relative standard deviation of the workload. Brokerchain, as an improved solution on Metis, demonstrated similar performance across various parameters, outperforming the Metis solution to a certain extent, indicating that edge-cut algorithms can significantly optimize the workload, albeit at the expense of increasing the disparity in the workload between shards.

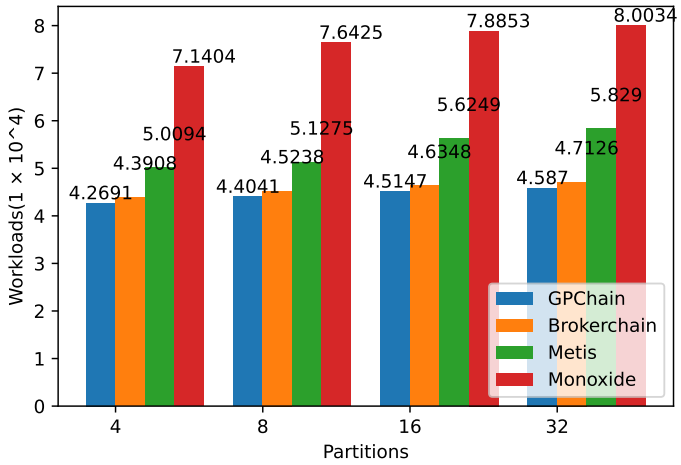

Figure 10. Total load of different partitions.

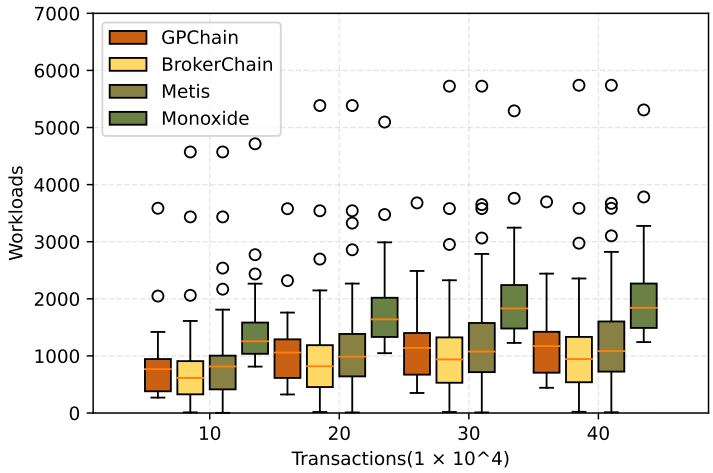

Figure 11. Total load of different transactions.

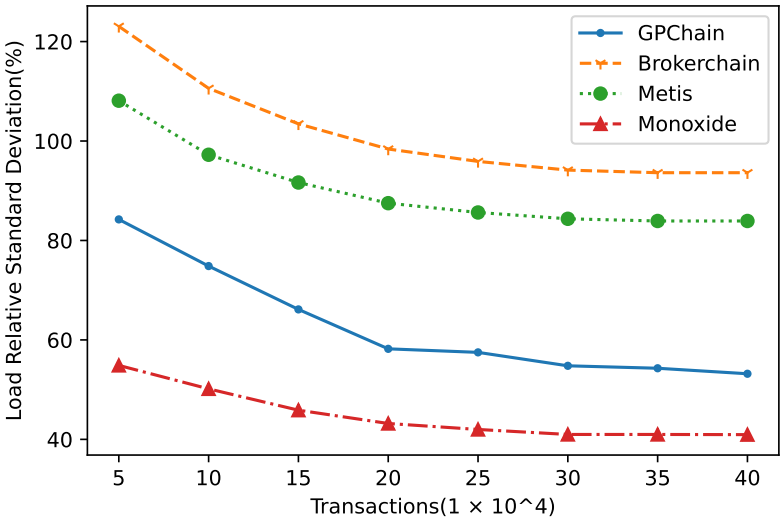

Figure 12. Load standard deviation of different schemes.

Table 1. Detailed workload.

Partitions	**8**			**16**			**32**			**64**		
	Max.	**Min.**	**Avg.**	**Max.**	**Min.**	**Avg.**	**Max.**	**Min.**	**Avg.**	**Max.**	**Min.**	**Avg.**
Monoxide	18,514	17,017	17,851	11,769	8134	9553	7399	3401	4928	4445	1535	2501
Metis	24,100	7689	12,523	14,329	3470	6409	6888	1372	3515	4585	275	1821
Brokerchain	22,725	5401	10,977	13,141	2424	5654	6691	708	2896	4313	58	1472
GPChain	12,483	9608	10,672	6745	4137	5505	4357	1519	2821	2449	559	1433

7. Conclusions and Future Work

In this paper, we present a graph partition-based blockchain-sharding protocol and employ a multi-layer architecture to achieve open, trustworthy, and efficient trust management. The trust-management model based on the blockchain that we propose consists of two parts: evidence collection and storage based on the blockchain. The collected evidence between vehicles is fused using the DST theory to derive trust information, and the graph partition-based sharding blockchain scheme provides a more efficient division and storage for trust and evidence transactions among vehicles. Finally, simulation experiments

using real-world datasets demonstrated better performance in both trust management and scalability aspects. Currently, we have focused on the research and discussion of the improved blockchain-sharding protocol for storage and scalability in blockchain-based trust management. For trust management in the IoV, there are still numerous factors to consider, such as the more detailed status information of vehicles. As future work, we aim to conduct further research on how to utilize the blockchain and smart contracts for the better management of these factors.

Author Contributions: Conceptualization, H.H. and S.C.; methodology, H.H.; software, S.C.; validation, Z.X.; formal analysis, H.H.; investigation, H.H.; resources, S.C.; data curation, H.H.; writing—original draft preparation, S.C.; writing—review and editing, S.C.; visualization, S.C.; supervision, Z.X. and X.D.; project administration, J.Z.; funding acquisition, J.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This work is supported by the Key-Area Research and Development Program of Hubei Province 2022BAA040, the Science and Technology Project of Department of Transport of Hubei Province 2022-11-4-3, and the Innovation Fund of Hubei University of Technology BSQD2019027, BSQD2019020, and BSQD2016019.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available in this article.

Conflicts of Interest: Author Jing Zeng was employed by the company China Gridcom Co., Ltd. The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as potential conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

v	Vertex of account
e	Edge of transaction
G	Transaction state graph
G'	Shard's transaction state graph
P_{Shard}	Primary Node of shard
P_{Leader}	Primary Node of all shard

References

1. Luu, L.; Narayanan, V.; Zheng, C.; Baweja, K.; Gilbert, S.; Saxena, P. A Secure Sharding Protocol For Open Blockchains. In Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security, Vienna, Austria, 24–28 October 2016. [CrossRef]
2. Li, Y.; Wang, J.; Zhang, H. A survey of state-of-the-art sharding blockchains: Models, components, and attack surfaces. *J. Netw. Comput. Appl.* **2023**, *217*, 103686. [CrossRef]
3. Zhang, Z.; Wang, X.; Yu, G.; Ni, W.; Liu, R.P.; Georgalas, N.; Reeves, A. A Community Detection-Based Blockchain Sharding Scheme. In Proceedings of the 5th International Conference, Held as Part of the Services Conference Federation, SCF 2022, Honolulu, HI, USA, 10–14 December 2022. [CrossRef]
4. Zhang, C.; Wei, F.; Liu, Q.; Tang, Z.G.; Li, Z. Graph Edge Partitioning via Neighborhood Heuristic. In Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, 13–17 August 2017. [CrossRef]
5. Liu, Y.; Wang, J.; Yan, Z.; Wan, Z.; Jäntti, R. A Survey on Blockchain-Based Trust Management for Internet of Things. *IEEE Internet Things J.* **2023**, *10*, 5898–5922. [CrossRef]
6. Yang, Z.; Wang, R.; Wu, D.; Yang, B.; Zhang, P. Blockchain-Enabled Trust Management Model for the Internet of Vehicles. *IEEE Internet Things J.* **2023**, *10*, 12044–12054. [CrossRef]
7. Liu, Q.; Gong, J.; Liu, Q. Blockchain-Assisted Reputation Management Scheme for Internet of Vehicles. *Sensors* **2023**, *23*, 4624. [CrossRef] [PubMed]
8. Yang, Z.; Yang, K.; Lei, L.; Zheng, K.; Leung, V.C.M. Blockchain-Based Decentralized Trust Management in Vehicular Networks. *IEEE Internet Things J.* **2019**, *6*, 1495–1505. [CrossRef]
9. Singh, P.K.; Singh, R.; Nandi, S.K.; Ghafoor, K.Z.; Rawat, D.B.; Nandi, S. Blockchain-Based Adaptive Trust Management in Internet of Vehicles Using Smart Contract. *IEEE Trans. Intell. Transp. Syst.* **2021**, *22*, 3616–3630. [CrossRef]

10. Wang, J.; Wang, H. Monoxide: Scale out Blockchains with Asynchronous Consensus Zones. In Proceedings of the 16th USENIX Symposium on Networked Systems Design and Implementation (NSDI 19), Boston, MA, USA, 26–28 February 2019.
11. Huang, H.; Peng, X.; Zhan, J.; Zhang, S.; Lin, Y.; Zheng, Z.; Guo, S. BrokerChain: A Cross-Shard Blockchain Protocol for Account/Balance-based State Sharding. In Proceedings of the IEEE INFOCOM 2022—IEEE Conference on Computer Communications, London, UK, 2–5 May 2022. [CrossRef]
12. Xi, J.; Xu, G.; Zou, S.; Lu, Y.; Li, G.; Xu, J.; Wang, R. A Blockchain Dynamic Sharding Scheme Based on Hidden Markov Model in Collaborative IoT. *IEEE Internet Things J.* **2023**, *10*, 14896–14907. [CrossRef]
13. Xu, M.; Feng, G.; Ren, Y.; Zhang, X. On Cloud Storage Optimization of Blockchain With a Clustering-Based Genetic Algorithm. *IEEE Internet Things J.* **2020**, *7*, 8547–8558. [CrossRef]
14. Gao, N.; Huo, R.; Wang, S.; Huang, T.; Liu, Y. Sharding-Hashgraph: A High-Performance Blockchain-Based Framework for Industrial Internet of Things With Hashgraph Mechanism. *IEEE Internet Things J.* **2022**, *9*, 17070–17079. [CrossRef]
15. Yang, Z.; Li, M.; Yang, R.; Yu, F.R.; Zhang, Y. Blockchain Sharding Strategy for Collaborative Computing Internet of Things Combining Dynamic Clustering and Deep Reinforcement Learning. In Proceedings of the ICC 2022—IEEE International Conference on Communications, Seoul, Republic of Korea, 16–20 May 2022. [CrossRef]
16. Jia, D.; Xin, J.; Wang, Z.; Wang, G. Optimized Data Storage Method for Sharding-Based Blockchain. *IEEE Access* **2021**, *9*, 67890–67900. [CrossRef]
17. Cai, Z.; Liang, J.; Chen, W.; Hong, Z.; Dai, H.N.; Zhang, J.; Zheng, Z. Benzene: Scaling Blockchain with Cooperation-Based Sharding. *IEEE Trans. Parallel Distrib. Syst.* **2023**, *34*, 639–654. [CrossRef]
18. Karypis, G.; Kumar, V. A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs. *SIAM J. Sci. Comput.* **1998**, *20*, 359–392. [CrossRef]
19. Chen, R.; Shi, J.; Chen, Y.; Zang, B.; Guan, H.; Chen, H. PowerLyra: Differentiated Graph Computation and Partitioning on Skewed Graphs. *ACM Trans. Parallel Comput.* **2018**, *5*, 13:1–13:39. [CrossRef]
20. Gonzalez, J.E.; Low, Y.; Gu, H.; Bickson, D.; Guestrin, C. PowerGraph: Distributed Graph-Parallel Computation on Natural Graphs. In Proceedings of the 10th USENIX Symposium on Operating Systems Design and Implementation (OSDI 12), Hollywood, CA, USA, 17–30 October 2012.
21. Jain, N.; Liao, G.; Willke, T.L. GraphBuilder: scalable graph ETL framework. In Proceedings of the International Conference on Management of Data, New York, NY, USA, 23 June 2013. [CrossRef]
22. Petroni, F.; Querzoni, L.; Daudjee, K.; Kamali, S.; Iacoboni, G. HDRF: Stream-Based Partitioning for Power-Law Graphs. In Proceedings of the 24th ACM International Conference on Information and Knowledge Management, Melbourne, Australia, 18–23 October 2015. [CrossRef]
23. Xie, C.; Yan, L.; Li, W.; Zhang, Z. Distributed Power-law Graph Computing: Theoretical and Empirical Analysis. In Proceedings of the Advances in Neural Information Processing Systems 27 (NIPS 2014), Montreal, Canada, 8–13 December 2014.
24. Kokoris-Kogias, E.; Jovanovic, P.; Gasser, L.; Gailly, N.; Syta, E.; Ford, B. OmniLedger: A Secure, Scale-Out, Decentralized Ledger via Sharding. In Proceedings of the 2018 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, USA, 20–24 May 2018. [CrossRef]
25. Zamani, M.; Movahedi, M.; Raykova, M. RapidChain: Scaling Blockchain via Full Sharding. In Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, Toronto, ON, Canada, 15–19 October 2018. [CrossRef]
26. Qi, X. S-Store: A Scalable Data Store towards Permissioned Blockchain Sharding. In Proceedings of the IEEE INFOCOM 2022—IEEE Conference on Computer Communications, London, UK, 2–5 May 2022. [CrossRef]
27. Urbani, M.; Gasparini, G.; Brunelli, M. A numerical comparative study of uncertainty measures in the Dempster-Shafer evidence theory. *Inf. Sci.* **2023**, *639*, 119027. [CrossRef]
28. Bezerra, E.D.C.; Teles, A.S.; Coutinho, L.R.; Silva, F.J.d.S.e. Dempster-Shafer Theory for Modeling and Treating Uncertainty in IoT Applications Based on Complex Event Processing. *Sensors* **2021**, *21*, 1863. [CrossRef] [PubMed]
29. Hamda, N.E.I.; Hadjali, A.; Lagha, M. Multisensor Data Fusion in IoT Environments in Dempster-Shafer Theory Setting: An Improved Evidence Distance-Based Approach. *Sensors* **2023**, *23*, 5141. [CrossRef] [PubMed]
30. Yuan, J.; Zheng, Y.; Xie, X.; Sun, G. Driving with knowledge from the physical world. In Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, CA, USA, 21–24 August 2011. [CrossRef]
31. Yuan, J.; Zheng, Y.; Zhang, C.; Xie, W.; Xie, X.; Sun, G.; Huang, Y. T-drive: driving directions based on taxi trajectories. In Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems, San Jose, CA, USA, 2–5 November 2010. [CrossRef]