PRVB: Achieving Privacy-Preserving and Reliable Vehicular Crowdsensing via Blockchain Oracle

Journal:	*IEEE Transactions on Vehicular Technology*
Manuscript ID	VT-2020-02255
Suggested Category:	Regular Paper
Date Submitted by the Author:	08-Jul-2020
Complete List of Authors:	Zhang, Can; Beijing Institute of Technology, School of Computer Science and Technology Zhu, Liehuang; Beijing Institute of Technology, School of Computer Science Xu, Chang; Beijing Institute of Technology, Sharif, Kashif; Beijing Institute of Technology, School of Computer Science and Technology
Keywords:	Crowdsensing, Blockchain, Data Aggregation, Privacy Protection

PRVB: Achieving Privacy-Preserving and Reliable Vehicular Crowdsensing via Blockchain Oracle

Can Zhang, Liehuang Zhu, *Member, IEEE,* Chang Xu, Kashif Sharif *Member, IEEE*

***Abstract*—Vehicular crowdsensing is attracting more and more attention because of its wide sensing coverage and diverse usage in smart cities. However, privacy issues that stem from traditional vehicular crowdsensing scenarios, violate the participant's privacy. Although some privacy-preserving schemes have been designed that aim to protect the sensitive information of sensed data, the reliability cannot be guaranteed because of the system's centralized structure. The introduction of blockchain in crowdsensing applications provides reliable data storage, however, the reliability of data sources remains an open challenge. Under these circumstances, the crowdsourcing service requester may not be able to obtain quality data. To solve these problems, we propose a novel Privacy-preserving and Reliable Vehicular crowdsensing via Blockchain oracle, called PRVB. More specifically, a privacy-preserving vehicular data aggregation scheme is presented to protect the data privacy and unlinkability between participant vehicles and sensed data. Besides, two protocols are designed to protect data privacy and to achieve fair rewards for data providers. Thorough theoretical analysis and experimental evaluations have proved that the proposed PRVB achieves privacy protection, reliability, and fairness with significant computational & communicational efficiency.**

***Index Terms*—Vehicular Crowdsensing, Blockchain, Privacy Protection, Data Aggregation.**

I. Introduction

THE recent developments in the Internet of Things (IoT) and vehicular technologies have increased the number of users electric vehicles (EVs) which are equipped with a diverse range of sensors that can be used to sense the environment around the vehicle and can help the drivers make informed driving decisions. Besides, some EVs are augmented with powerful communication and computation embedded devices called On-Board Units (OBU). Hence, a vehicle can communicate with other vehicles or complete a collaborative computation task with other vehicles.

Vehicular crowdsensing [1] has become an important application area in the vehicular networks and has received significant attention from both academic and industrial researchers [2], [3]. In a typical vehicular crowdsensing scenario, there exists a requester that assigns the sensing task. Each vehicle participating in the task first uses the on-board sensors to collect sensing data (e.g., speed, temperature, location), and then it sends it to the data aggregator via a nearby Road-Side Unit (RSU). The data aggregator collects the sensed data and computes the results that are then sent to the task requester. Finally, the requester pays for the result, and the payment is distributed to workers. Compared with mobile crowdsensing, vehicular crowdsensing systems have high mobility and wide coverage of the sensing area due to the nature of vehicles (and vehicular networks). Hence, vehicular crowdsensing can be widely used for efficient traffic monitoring, environmental condition measurements, and information sharing from the smart city perspective.

However, security issues become more and more severe in recent years that motivate the researchers to focus on how to design some privacy-preserving mechanisms [4], [5] in various applications. In the vehicular crowdsensing scenario, the traditional methodologies bring certain security issues [6] which jeopardize the privacy of sensing vehicles. For example, the aggregator can infer the trajectory of a vehicle from collected location data, or even determine the living and driving habits with the help of background knowledge (e.g., a city map). Hence, a series of privacy-preserving vehicular crowdsensing schemes [7]–[9] have been presented in the literature to protect the privacy of vehicles (or the owners of the vehicles). Unfortunately, some of the existing schemes rely on the centralized structure of the whole system, which means that the centralized aggregator can become a single-point failure, and hence the system availability cannot be guaranteed.

The recent integration of blockchain technology into different domains has motivated the use of crowdsensing applications to leverage it as their underlying platform to eliminate the issues caused by centralization. Blockchain relies on the data provider called Blockchain Oracle [10] (BO), to provide external data that is used by the smart contracts. In a typical blockchain-based crowdsensing system, the data provider that publishes the sensing data to the blockchain can be considered as a BO. In the existing schemes, although the reliability of crowdsensing data stored on the blockchain is guaranteed, the reliability of BO itself cannot be guaranteed. For instance, some BOs might send incorrect results to the blockchain because of accidental error or dishonest behaviors, which significantly impacts the service requester not only in terms of data quality but also for payments & rewards.

Motivated by these shortcomings of existing schemes, we propose a privacy-preserving and reliable vehicular crowdsensing scheme via blockchain oracle, called PRVB. This scheme

This work is supported by the National Natural Science Foundation of China (Grant Nos. 61972037, 61402037, 61872041, U1836212), and grant No. 2019CX10014 from the Graduate Technological Innovation Project of Beijing Institute of Technology.

Can Zhang, Liehuang Zhu (Corresponding Author), Chang Xu (Corresponding Author), and Kashif Sharif are with the School of Computer Science and Technology at Beijing Institute of Technology, Beijing, China (E-mail: {canzhang, liehuangz, xuchang, 7620160009}@bit.edu.cn).

protects the privacy of both participants and BOs, guarantees the reliability of crowdsensing data, and achieves fairness of the concerned BOs.

Following are the major contributions of this work:

- We present PRVB, a blockchain-based vehicular crowdsensing scheme that leverages the BO to provide reliable crowdsensing services. PRVB consists of two major processes: *Vehicular Data Aggregation* and *On-chain Data Aggregation*. To the best of our knowledge, there is the first privacy-preserving vehicular crowdsensing scheme that achieves the reliability of data source and fairness of data requester & provider.
- We propose a privacy-preserving vehicular data aggregation scheme that achieves unlinkability between vehicle and its sensed data based on pruned prefix binary tree (PPBT) without the help of TA. It also achieves data privacy protection which means only the aggregator (designated BO) can get the sensed data collected by each vehicle. Besides, the data reliability can be achieved by Truth Discovery (TD) algorithm. Note that the proposed data aggregation schemes can also be used for other data aggregation based on relevant scenarios.
- We design two protocols to achieve privacy-preserving on-chain data aggregation with fairness by leveraging smart contracts. The encrypted data of each BO will be sent to the blockchain so that it cannot be revealed to adversaries before the consensus (aggregation) process. During the consensus process, the data will be recovered by PCTD's threshold decryption, and the crowdsensing result will be obtained by the on-chain TD algorithm. Finally, the reward of each BO is based on the contributions, which achieves fairness.
- Comprehensive security analysis and experimental evaluation shows that the proposed PRVB scheme achieves privacy protection, fairness, and reliability with high computational & communicational efficiency.

The rest of this paper is organized as follows. In Section II we describe the related works on data aggregation and blockchain-based crowdsensing systems. We make a brief introduction of blockchain basics, truth discovery, PCTD cryptosystem, and other cryptographic primitives in Section III. Section IV proposes the formal system model, threat model, and design goals. The detailed construction of PRVB is given in Section V. Security analysis and experimental evaluation are given in Section VI and Section VII, respectively. Finally, Section VIII concludes this paper.

II. Related Works

In this section, we introduce the related works about privacy-preserving data aggregation and blockchain-based crowdsensing. Inspired by some of the existing works discussed in forthcoming sections, the objective is to propose a privacy-preserving and reliable vehicular crowdsensing scheme that protects the identity-privacy of users and data-privacy of blockchain oracles while ensuring reliability & financial fairness.

A. Privacy-preserving Data Aggregation

In the typical crowdsensing scenarios, users collect the sensing data, and the aggregator aggregates that data to obtain the crowdsensing results. Hence, data aggregation plays an important role in the crowdsensing process. To solve the privacy issues, some privacy-preserving data aggregation schemes [11]–[15] have been presented in literature.

Lu et al. [11] proposed LPDA, a light-weight privacy-preserving data aggregation scheme to be used in fog computing-based IoT, and provides some enhanced security guarantees. Huang et al. [12] leveraged homomorphic proxy re-encryption and a proxy re-authenticator to support data aggregation over selective data types, while being resistant to both non-collusive and collusive attacks. Gong et al. [13] proposed a privacy-preserving data aggregation scheme with no trusted authority while providing communication efficiency and $(n-k)$-source anonymity. Zhu et al [14] used privacy-preserving signature schemes and homomorphic encryption to achieve privacy-preserving data aggregation in fog-based smart grid scenarios. Based on fog computing, Shen et al. [15] also proposed a privacy-preserving aggregation scheme that supports dynamic groups, where the fog devices can filter out false data and aggregate the encrypted data to save bandwidth.

B. Blockchain-based Crowdsensing

Capitalizing on the properties of blockchain technology, some researchers [16]–[20] have tried to leverage it for solving the problems of centralization in existing crowdsensing schemes. Yang et al. [16] proposed a privacy-preserving scheme based on a private blockchain that can protect the worker's location privacy and allows anonymous payments. Zhao et al. [17] proposed a privacy-preserving reputation management mechanism in mobile crowdsensing systems that can resist malicious users with high utility and security guarantees. Duan et al. [18] designed a novel crowdsensing system that ensures confidentiality on individual data and differential privacy on aggregated data. Experiments based on Ethereum also demonstrate that their design achieves high efficiency with low financial costs (i.e., Gas used in Ethereum).

Besides these, some blockchain-based solutions [19], [20] also consider fairness during the crowdsensing process. Wang et al. [19] leveraged the build-in cryptocurrency system in blockchain to realize fair incentives in crowdsensing applications. More specifically, the user with higher quality can get more rewards, miners can also verify the data quality and transactions without violating users' privacy. Cai et al. [20] also presented a new crowdsensing framework that utilizes privacy-preserving truth discovery on streamed data and blockchain to protect on-chain data privacy and achieve financial fairness [21]. Experiments based on Microsoft Azure cloud and Ethereum show the accuracy, efficiency, and low financial costs of the proposed scheme.

III. Preliminaries

In this section, we introduced the blockchain basics, the truth discovery algorithm, the PCTD cryptosystem, and other cryptographic primitives used in the proposed PRVB scheme.

A. Blockchain Basics

In a blockchain network, there is no central entity or node that can control the whole network, and once the data is stored on the blockchain it cannot be deleted or be tampered with. The blockchain is generally categorized as Public or Private, where, in a *Public Blockchain* where everyone and anyone can participate, while in a *Permissioned Blockchain* only authorized nodes with real-world identities can register to be part of the network [22].

To maintain the consistency of the decentralized blockchain network, a consensus mechanism plays an important role. Public blockchain mainly uses the proof-based consensus (e.g., Proof of Work used in Bitcoin [23] and Ethereum [24]), while permissioned blockchain mainly uses the BFT (Byzantine Fault Tolerance) as the kernel of consensus (e.g., Practical BFT [25] used in Hyperledger Fabric v0.6 [26]). In BFT-based consensus, each consensus round only exists one commit node that generates the block for that round and broadcasts it to the whole network. Other nodes participating in that round (named as witness/validating nodes) validate the received block and commits it to the self-maintained blockchain ledger after success. If more than 2/3 nodes of the network are honest, the consistency of the blockchain can be guaranteed. BFT is more efficient compared with the Proof-based consensus mechanism in public blockchain like Bitcoin [27].

Moreover, some blockchain network such as Ethereum and Hyperledger Fabric support execution programs for the blockchain, named as the smart contracts (in Hyperledger Fabric, it is also called chaincode). The application of smart contracts makes the blockchain dynamic and programmable so that it can be used in various kinds of applications.

B. Truth Discovery

Truth Discovery (TD) [28] can be used to obtain the truth value under noisy environments. For each numeric (or category) data $x_i \in \{x_1, \cdots, x_n\}$, the TD algorithm first initializes a ground truth x_0^* and assigns a random weight for each x_i. Then the truth value and all the weights are updated iteratively until the iterate time exceeds the threshold $iter_{max}$. More specifically, the TD algorithm used in the proposed PRVB scheme is given as Algorithm 1. Here the distance measurement function $d(\cdot)$ between two numeric values can be instantiated as Eq. 1:

$$d(x_i, x^*) = \frac{(x_i - x^*)^2}{std_n}, \tag{1}$$

where, std_m represents the standard deviation of $\{x_1, \cdots, x_n\}$.

In recent years, a series of privacy-preserving TD schemes [29]–[32] have been developed. However, in this work, the data privacy protection during the TD process is beyond the scope. Hence, we leave it as potential future work.

C. PCTD Cryptosystem

In this work, we adopt a Paillier-based threshold decryption scheme during the blockchain consensus process, namely

Algorithm 1 Truth Discovery.

```
Input: A set of data $X = \{x_1, \cdots, x_n\}$.
Output: Estimated truth value $x^*$ and a weight set $W = \{w_1, \cdots, w_n\}$ associated to each $x_i \in X$.
1: init $W$ by $n$ random values.
2: set $x^* = (\sum_{i=1}^{n} x_i)/n$.
3: for $iter = 1$ to $iter_{max}$ do
4:     for $i = 1$ to $n$ do
5:         update $w_i = \log \frac{\sum_{j=1}^{n} d(x_j, x^*)}{d(x_i, x^*)}$.
6:     end for
7:     update $x^* = \frac{\sum_{i=1}^{n} w_i \cdot x_i}{\sum_{i=1}^{n} w_i}$.
8: end for
9: return $x^*$, $W$.
```

PCTD (Paillier-based Cryptosystem with Threshold Decryption) [33]. PCTD achieves (t, n) threshold decryption by splitting the private key into different parts. The PCTD scheme includes the following six algorithms:

- $(pk, sk) \leftarrow KeyPairGen(\lambda)$: This probabilistic algorithm receives a secure parameter λ as input, and it outputs a key pair (pk, sk).
- $c \leftarrow Enc(pk, m)$: This probabilistic algorithm receives a public key pk and a message m as input, and it outputs the encrypted message c.
- $m \leftarrow Dec(sk, c)$: This deterministic algorithm receives a private key sk and a ciphertext c as input, and it outputs the decrypted result m.
- $(psk_1, \cdots, psk_n) \leftarrow KeySplit(n, sk)$: This probabilistic algorithm receives the number of partial keys n and a private key sk as input, and it outputs n partial keys $(psk_1, \cdots, psk_n)$.
- $c_i \leftarrow PDec(c, psk_i)$: This deterministic algorithm receives a ciphertext c and a partial private key psk_i as input, and it outputs the partially decrypted ciphertext c_i.
- $m \leftarrow TDec(c_1, \cdots, c_n)$: This deterministic algorithm receives n partially decrypted ciphertexts $(c_1, \cdots, c_n)$ as input, and it outputs the decrypted message m.

We omit the detailed implementation of PCTD for simplicity, the full version of PCTD can be found in [33].

D. Cryptographic Primitives

To achieve privacy-preservation and integrity, two secure pseudo-random functions F, G, and a secure hash function H are adopted, as illustrated in Eq. 2.

$$F : \{0,1\}^\lambda \times \{0,1\}^* \to \{0,1\}^\lambda \tag{2a}$$
$$G : \{0,1\}^\lambda \times \{0,1\}^* \to \{0,1\}^l \tag{2b}$$
$$H : \{0,1\}^* \to \{0,1\}^\lambda \tag{2c}$$

We also use a secure Public-Key Encryption (PKE) cryptosystem to encrypt the sensitive information for each entity. A PKE cryptosystem consists of the following three functions:

- $(pk, sk) \leftarrow KeyGen(\lambda)$ receives a secure parameter λ as input, and results in generation of the key pair (pk, sk).
- $c \leftarrow Enc(m, pk)$ receives a message m and a public key pk as input, and resultantly generates the encrypted ciphertext c.

- $m \leftarrow Dec(c, sk)$ receives a ciphertext c and a private key sk as input, and gives back the decrypted message m.

Some PKE cryptosystem can also support digital signatures which consist of the following additional two functions:

- $\sigma \leftarrow Sig(m, sk)$ receives a message m and a private key sk as input, and generates a signature σ.
- $res \leftarrow Verify(m, \sigma, pk)$ receives a message m, a signature σ, and a public key pk as input, and outputs the verification result $res \in \{true, false\}$.

The formal definition of the secure pseudo-random function, secure hash function, and secure PKE scheme can be found in [34], hence we omit them for simplicity.

IV. Problem Formalization

In this section, we give the detailed system model, threat model and design goals of the novel proposed scheme.

A. System Model

The proposed system comprises five entities: TA (Trusted Authority), User, RSU (Road-Side Unit), BO (Blockchain Oracle) and Blockchain network, as shown in Fig. 1. We first describe these individually.

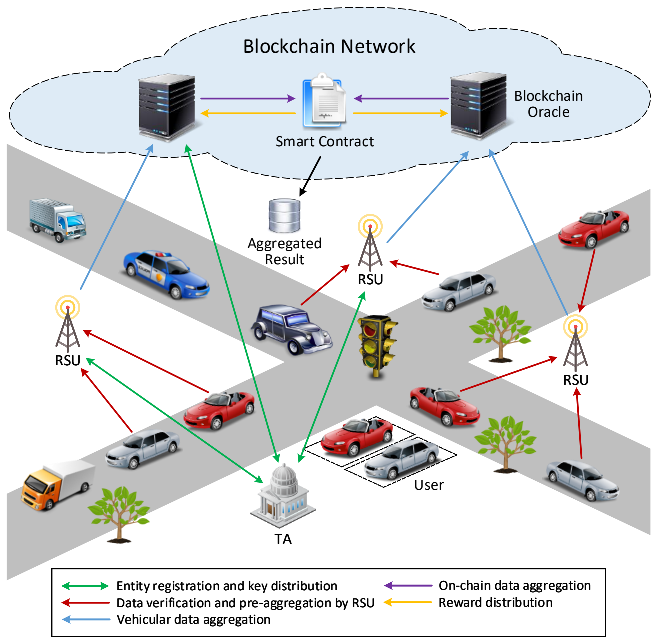
Blockchain Network
Blockchain Oracle
Smart Contract
Aggregated Result
RSU
RSU
RSU
User
TA
Entity registration and key distribution
Data verification and pre-aggregation by RSU
Vehicular data aggregation
On-chain data aggregation
Reward distribution

Fig. 1: System model.

1) TA: TA generates all the public system parameters, registers each RSU and blockchain oracle, and deploys the smart contracts. Note that TA will stay offline after the key distribution process unless a dispute arises. Therefore, the introduction of TA does not affect the decentralization property of the blockchain network.

2) Users: We assume that each user owns an Electric Vehicle (EV). When the requester assigns a vehicular crowdsensing task, the user encrypts the data gathered by EV, obfuscates the identity, and sends it to the nearest RSU.

3) RSUs: Deployed by the vehicular services providers, RSU verifies the encrypted & obfuscated messages sent by the users and performs a pre-aggregate operation. After this, the RSU obtains several encrypted data elements that are unrelated to the sender's identity and sends them to the designated blockchain oracles.

4) BOs: BO plays an important rule because it participates in the blockchain consensus process, and provides reliable external data to the blockchain. After the final aggregation, they will get the corresponding rewards. Note that in the following sections, we also use the term "oracle" to represent blockchain oracle for simplicity, unless otherwise specified.

5) Blockchain Network: When receiving the data sent by BOs, the blockchain will decrypt the data encrypted by each BO under the consensus process, and automatically invoke the smart contract containing the TD algorithm. Finally, the smart contract outputs the aggregated truth values and rewards all the BOs for their contributions.

B. Threat Model

In our security model, we assume that TA is trustworthy which means it cannot be compromised. Besides, during the key distribution process, TA uses a secure channel that cannot be eavesdropped or tampered with, and the corresponding keys cannot be compromised.

We also assume that both RSUs and most users are honest-but-curious. More specifically, they will follow the proposed algorithms and protocols, however, they may try to link the data sent by other users and the corresponding identities to reduce the privacy and anonymity levels. Some users may collude with RSU to obtain the link between the data and the corresponding identity. We also assume that the users within a group are fixed, which means that the user's addition and removal during the aggregation process is not allowed.

The data stored on the blockchain cannot be modified or deleted. However, it is stored publicly so that everyone including attackers can execute an inference attack based on the blockchain data. For instance, attackers may want to obtain the private key of each BO or the owned data before the final aggregation process.

Most BOs can be also considered as honest-but-curious. Different from users and RSUs, the identity of every BO should be published for reliability and transparency. However, they may also try to steal other oracle's data for more profits.

Note that in our proposed threat model, we assume that collusion between oracles & roadside units and collusion among oracles is not allowed. These assumptions are reasonable because, 1) RSUs are controlled by vehicle services providers or government, and 2) BOs are controlled by different organizations or companies, they are not willing to share the data because of their competitive relationship.

C. Design Goals

The proposed PRVB scheme has several design goals that must be achieved to ensure its effectiveness. These are listed below.

1) Privacy Protection: The proposed scheme should protect the privacy of users and BOs. For users, their *identity privacy* should be protected during the whole crowdsensing process, and for BOs, their *data privacy* should be protected before the final aggression process is executed by blockchain.

2) Reliability: Each BO should provide reliable external data to the blockchain, and the execution of the smart contract (from source to result aggregation) should also be reliable. The aggregated results cannot be influenced by extreme values generated by a single-point failure or attack.

3) Fairness: The proposed scheme should guarantee two kinds of fairness, that is: 1) For a smart contract user (i.e., the requester during the vehicular crowdsensing process), if it pays the fees, it will receive reliable blockchain-based services, and 2) For a BO that provides external data if the data is closer to the truth value, it will get more/better rewards.

4) High Efficiency: Due to the limited resources in the VANET, the proposed scheme should be efficient. More specifically, the computation costs during the whole process and the communication costs between vehicles, RSUs and BOs are as low as possible.

V. The Proposed PRVB Scheme

In this section, we give the detailed description of the proposed PRVB scheme. More specifically, PRVB consists of four processes: *System Initialization*, *Vehicular Data Aggregation*, *External Data Publication*, and *On-chain Data Aggregation*. Table I shows the notations used in the following work.

TABLE I: List of Notations.

Notation	Description
λ	Security parameter.
m	The number of users in a group.
n	The number of blockchain oracles (BOs).
r	The number of RSUs.
t	Threshold for PCTD decryption.
F, G	Secure pseudo-random functions.
H	Secure hash function.
k_{ab}	Shared key between user u_a and u_b.
(pk_i, sk_i)	Key pair of entity i for PKE cryptosystem.
(pk, sk)	Public key and master private key for PCTD.
psk_j	BO_j's partial private key.
$\|s\|$	Bit-length of s.
l	Maximum bit-length of PCTD's ciphertext.
$a[i]$	The i-th element for a set/vetcor a.
addr_j	The address of BO j.
addr	The address of smart contract $\mathcal{C}$.
α	Reward factor.
$\mathcal{P}$	Prefix set of binary string.
w	Prefix length.
$S_i^{out}, S_i^{in}, T_i^{out}, T_i^{in}$	User u_i's shared key set.
v_i, V_i	User u_i's obfuscated identity and data vector.
fee	Payment given by crowdsensing requester.

A. System Initialization

In this process, every RSU and BO should use their real-world identities to register with the TA. Then TA generates the system parameters, the pseudonym, and the key pair for RSU and BO, and distributes them to the corresponding entities. Note that in PRVB, for privacy and flexibility concerns, TA is not needed during the vehicular data aggregation process.

Algorithm 2 Prefix Set Partition

```
Input: $m$ participated users $U = \{u_1, \cdots, u_m\}$, a prefix length $w$.
Output: A partitioned prefix set $\mathcal{P} = \{\mathcal{P}_1, \cdots, \mathcal{P}_m\}$.
1: init a prefix set $tmp = \{* \ldots *\}$.
2: init a partitioned prefix set $\mathcal{P} = \perp$.
3: // Note: Line 4-6 are executed by users.
4: for each $u_i \in U$ do
5:     $u_i$: generate $r_i \leftarrow \{0,1\}^w$.
6: end for
7: while $\mathcal{P}.size < m$ do
8:     for each $\mathcal{P}_i \in tmp$ do
9:         set $tmp$ = $tmp \setminus \mathcal{P}_i$.
10:        set ($\mathcal{P}_{il}$, $\mathcal{P}_{ir}$) = $BinaryPartition(\mathcal{P}_i)$.
11:        set $tmp$ = $tmp \cup \{\mathcal{P}_{il}, \mathcal{P}_{ir}\}$.
12:        send $tmp$ to each $u_i \in U$.
13:    end for
14:    // Note: Line 15-18 are executed by users.
15:    for each $u_i \in U$ do
16:        $u_i$: compute $v_i$ = $IVectorGen()$.
17:        $u_i$: send $v_i$ to $\mathcal{R}$.
18:    end for
19:    compute $ctr = \sum_{i=1}^{n} v_i$.
20:    for $i = 1$ to $|tmp|$ do
21:        if $ctr[i] == 0$ then
22:            set $tmp = tmp \setminus tmp[i]$.
23:        else if $ctr[i] == 1$ then
24:            set $\mathcal{P} = \mathcal{P} \cup tmp[i]$.
25:            set $tmp = tmp \setminus tmp[i]$.
26:        end if
27:    end for
28: end while
29: return $\mathcal{P}$.
```

Hence, all the users do not need to register with the TA. The precise steps involved in system initialization are listed below.

1) System Parameter Generation: First TA generates a secret key $K_0 \leftarrow \{0,1\}^\lambda$ used for entity registration. Then it executes PCTD.$KeyGen(\lambda)$ to generate the key pair (pk, sk) and deploys the smart contract $\mathcal{C}$ associated with the address addr. The smart contract is used to achieve reliable on-chain data aggregation and fair incentive mechanisms. Finally, (pk, addr) can be seen as the public parameter and (K_0, sk) is the private parameter held by TA secretly.

2) RSU Registration: For each RSU associated with its identity r_i, TA first computes the RSU's pseudonym RID_i by calculating $RID_i = F(K_0, r_i)$. Next it generates the key pair (pk_i, sk_i) = PKE.$KeyGen(\lambda)$ for generating digital signatures.

3) BO Registration: When a blockchain oracle BO_j wants to register with the system, it must provide both its real-world identity o_j and a blockchain address addr_j for receiving rewards. Assume that the number of BO is n, TA first executes PCTD.$KeySplit(n, sk)$ to split the master private key sk into n parts $(psk_1, \cdots, psk_n)$. Then, TA publish BO$_j$'s identity information (o_j, addr_j) to the blockchain in order to achieve transparency and fairness. Note that each BO$_j$ has its own key pair (pk_j, sk_j), and pk_j can also be used to generate the blockchain address $\mathsf{addr_j}$. Next, TA assigns a unique partial key psk_j for each BO_j for encryption.

4) Parameter & Key Distribution: After all entity registration processes are finished, TA sends (pk, addr) to BO, sends (pk_i, sk_i) to each RSU_i, and sends (pk_j, sk_j) to each BO_j through secure channels as defined in the threat model.

B. Vehicular Data Aggregation

In this process, some adjacent users (EVs) will form a group to achieve privacy-preserving aggregation without the help of TA. First, every user in the group should share some secrets with one other user to obfuscate the collected data and identities. Then each user will be assigned to a unique sequence number by the nearby RSU nearby, following which, each user uses the shared secret and assigned sequence number to generate an obfuscated vector, and sends it to the RSU. After receiving all the vector, RSU pre-aggregates the vector and sends the result to the designated BO. Note that for simplicity, we only consider the scenario that includes one RSU $\mathcal{R}$ (i.e., $r = 1$), a group with m users $U = \{u_1, \cdots, u_m\}$, and $\mathcal{R}$'s designated blockchain oracle is noted as BO_j. These steps are detailed below.

1) Key Sharing: For each $u_a, u_b \in U$ where $a \neq b$, one shared quadruple $(s_{ab}, s_{ba}, t_{ab}, t_{ba})$ will be generated, where the bit-length of each element is λ. More specifically, s_* is used to obfuscate the identity during the sequence number generation process, and t_* is used to perturb the encrypted data. Note that u_a and u_b can use the existing secret exchange method (e.g., Diffle-Hellman method [35]) to share the key k_{ab}. Considered a q-order group $\mathbb{Z}$ with a generator g, u_a chooses a random number a as the private key, and sends g^a to u_b. Likewise, u_b chooses a random number b as the private key, and sends g^b to u_a. Then, both u_a and u_b can obtain the shared key $k_{ab} = g^{ab}$ by calculating $(g^b)^a$ and $(g^a)^b$, respectively. After obtaining k_{ab}, both u_a and u_b compute $s_{ab} = F(k_{ab}, a||b||0)$, $s_{ba} = F(k_{ab}, b||a||0)$, $t_{ab} = G(k_{ab}, a||b||1)$, and $t_{ba} = G(k_{ab}, b||a||1)$.

For user a, after sharing keys with other $m-1$ users, it can get four sets $S_a^{out} = \bigcup_{u \in U \setminus \{u_a\}} s_{au}$, $S_a^{in} = \bigcup_{u \in U \setminus \{u_a\}} s_{ua}$, $T_a^{out} = \bigcup_{u \in U \setminus \{u_a\}} t_{au}$ and $T_a^{in} = \bigcup_{u \in U \setminus \{u_a\}} t_{ua}$, that will be used in the following sub-processes.

2) Prefix Partition: In this sub-process, RSU assigns a prefix $\mathcal{P}_i$ for each user u_i used for sequence number generation without knowing its relationship. To protect this kind of privacy, we propose a Pruned Prefix Binary Tree (PPBT) based privacy preserving prefix partition scheme.

We first introduce the concept of a prefix set. Given a binary string $b_1 b_2 \dots b_w$, where $b_i \in \{0, 1\}$, it can be matched by w prefix sets $\{b_1 b_2 \dots b_{w-1}*, \cdots, b_1 *^{w-1}, *^w\}$, where $*^w$ represents the arbitrary w-bit binary string. We use the symbol $\in$ to represent the match relationship between a binary string and a prefix set (e.g., $b_1 b_2 \dots b_w \in b_1 *^{w-1}$). We also use $\subseteq$ to represent the inclusion relationship between two prefix sets (e.g., $b_1 b_2 *^{w-2} \subseteq b_1 *^{w-1}$). Each prefix set can be partitioned into two non-intersecting sub-prefix sets, for instance, $\mathcal{P} := b_1 *^{w-1}$ can be binary partitioned to

Algorithm 3 IVectorGen

```
Input: A prefix set $tmp$, $u_i$'s random number $r_i$ and two
    shared key set $S_i^o, S_i^i$.
Output: An obfuscated identity vector $v_i$.
 1: set $t = |tmp|$.
 2: init a $t$-dimension vector $v_i$.
 3: for $k = 1$ to $t$ do
 4:     if $r_i \in tmp[i]$ then
 5:         set $b_{ik} = 1$.
 6:     else
 7:         set $b_{ik} = 0$.
 8:     end if
        set $v_i[k] = b_{ik} + \sum_{s \in S_i^{out}} F(s,k) - \sum_{s \in S_i^{in}} F(s,k)$.
 9: end for
10: return $v_i$.
```

$\mathcal{P}_l := b_1 0 *^{w-2}$ and $\mathcal{P}_r := b_1 1 *^{w-2}$. We use the function $(\mathcal{P}_l, \mathcal{P}_r) = \textit{BinaryPartition}(\mathcal{P})$ to describe this process.

The detailed description is shown in Algorithm 2 executed by $\mathcal{R}$ except in Line 4-6 and Line 15-18 that are executed by the users. First, $\mathcal{R}$ initializes a prefix set tmp and a partitioned prefix set $\mathcal{P}$, and each user u_i generates a random number r_i used for prefix partition. Then $\mathcal{R}$ generates n prefix sets $\{\mathcal{P}_1, \cdots, \mathcal{P}_n\}$ that satisfies: 1) $\mathcal{P}_a \cap \mathcal{P}_b = \emptyset, \forall a \neq b$, and 2) for each $\mathcal{P}_i$, $\mathcal{P}_i \subseteq *^w$, as shown in Line 7-28.

In Line 7-28, first $\mathcal{R}$ uses *BinaryPartition*() to partition each $\mathcal{P}_i \in tmp$ into two subsets. Then all the partitioned subsets will be added to tmp, and all the original sets will be removed. $\mathcal{R}$ then sends tmp for all users. Next for each user u_i, it computes a vector v_i generated by Algorithm 3 to obfuscate its identity, and sends it to $\mathcal{R}$.

Algorithm 3 is designed to generate u_i's identity obfuscation vector associated with the selected random number r_i. First u_i initializes a t-dimension vector v_i where the i-th dimension is related to the i-th element in tmp. If $r_i \in tmp[k]$, which means r_i is matched by the prefix $tmp[i]$, it sets the indicator $b_{ik} = 1$, otherwise it sets $b_{ik} = 0$. Next, it obtains $v_i[j]$ by computing $v_i[k] = b_{ik} + \sum_{s \in S_i^{out}} F(s,k) - \sum_{s \in S_i^{in}} F(s,k)$, where S_i^{out} and S_i^{in} are two shared key sets.

After receiving all the n vectors, $\mathcal{R}$ adds them to get the counter ctr. Note that:

$$
\begin{aligned}
ctr[k] &= \sum_{i=1}^{m} v_i[k] \\
&= \sum_{i=1}^{m} (b_{ik} + \sum_{s \in S_i^{out}} F(s,k) - \sum_{s \in S_i^{in}} F(s,k)) \\
&= \sum_{i=1}^{m} b_{ik} + \sum_{i=1}^{m} \sum_{s \in S_i^{out}} F(s,k) - \sum_{i=1}^{m} \sum_{s \in S_i^{in}} F(s,k) \\
&= \sum_{i=1}^{m} b_{ik} + \sum_{1 \le a,b \le m, a \neq b} F(s_{ab},k) - F(s_{ab},k) \\
&= \sum_{i=1}^{m} b_{ik}.
\end{aligned}
\quad (3)
$$

Hence, the sum of indicators can be recovered without knowing b_{ij}, which represents how many users' random numbers are matched in these partitioned subsets.

Next $\mathcal{R}$ checks the aggregated counter. If $ctr[k] == 0$, which means no user's random number is matched, the corresponding prefix set $tmp[k]$ will be removed (as a prune operation in binary tree search). If $ctr[k] == 1$, which means one user's random number is matched, the corresponding prefix set $tmp[k]$ will be added to $\mathcal{P}$. Otherwise, it means at least two user's random numbers are matched, hence, $tmp[k]$ will be binarily partitioned in next round. Finally, $\mathcal{R}$ obtains the partition set $\mathcal{P} = \{\mathcal{P}_1, \cdots, \mathcal{P}_m\}$ and sends $\mathcal{P}$ to every users. Note that each of $\mathcal{P}_k \in \mathcal{P}$ relates to one user's random number, whereas the link between $\mathcal{P}_k$ and r_i is obfuscated, which realizes anonymity.

3) Assigning Sequence Number: After receiving $\mathcal{P}$, each user u_i will determine which prefix set $\mathcal{P}_k \in \mathcal{P}$ can match the random number r_i. If $r_i \in \mathcal{P}_k$, it means u_i's sequence number s_i will be k. Because the above partition algorithm guarantees that each $\mathcal{P}_k$ is associated with a user's random number, each user can be assigned a unique sequence number used to obfuscate the aggregated data.

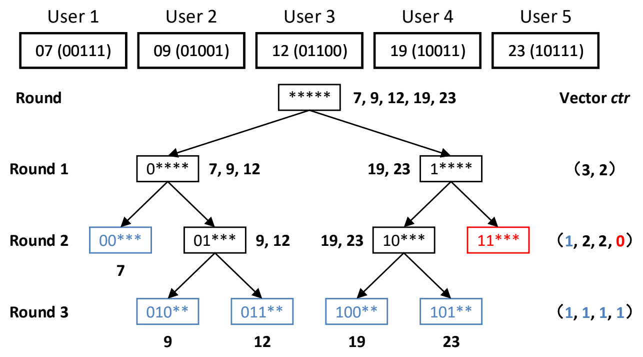

Fig. 2: An example of sequence number assignment.

Fig. 2 illustrates an example of assigning sequence numbers, where the whole process graph can be considered as a PPBT. The non-pruned leaf node (colored as blue) represents the final partitioned result and all the nodes in depth i represent the set tmp in the Round i. In this example, we set $w = 5$, and the random numbers from u_1 to u_5 are 7, 9, 12, 19 and 23, respectively. In Round 1, the initial prefix set $*^5$ is partitioned into $0*^4$ and $1*^4$. Based on the binary representation of each random number, we can find that $7, 9, 12 \in 0*^4$ and $19, 23 \in 1*^4$. After $\mathcal{R}$ aggregates the vector, it obtains $ctr = (3, 2)$, which means each of the prefix set in tmp contains at least two users, hence it will be partitioned in next round. In Round 2, it can be inferred from the aggregated vector $ctr = (1, 2, 2, 0)$ that $00*^3$ (colored as blue) contains only one user, hence it will be added to $\mathcal{P}$. Besides, prefix set $11*^3$ (colored as red) contains 0 users so it will be pruned (i.e., removed from tmp in next round), and other two sets ($01*^3$, $10*^3$) will be partitioned in next round. In Round 3, the vector $ctr = (1, 1, 1, 1)$ which means all the sets in tmp will be added to $\mathcal{P}$. Finally, $\mathcal{R}$ obtains the partitioned prefix set $\mathcal{P} = \{00*^3, 010*^2, 011*^2, 100*^2, 101*^2\}$, and sends it to all users $u_1, \cdots, u_m$.

Algorithm 4 DVectorGen

```
Input: User $u_i$'s sequence number $seq_i$, collected vehicular data $d_i$, two shared key set $T_i^{out}, T_i^{in}$, and BO $j$'s public key $pk_j$.
Output: An obfuscated data vector $V_i$.
1: init an $m$-dimension vector $V_i$.
2: for $k = 1$ to $m$ do
3:     if $seq_i == k$ then
4:         set $c_k = \text{PKE}.Enc(pk_j, d_i)$.
5:         set $V_i[k] = c_k + \sum_{t \in T_i^{out}} G(t,k) - \sum_{t \in T_i^{in}} G(t,k)$.
6:     else
7:         set $V_i[k] = \sum_{t \in T_i^{out}} G(t,k) - \sum_{t \in T_i^{in}} G(t,k)$.
8:     end if
9: end for
10: return $V_i$.
```

Note that during the whole process, $\mathcal{R}$ only knows the set tmp and the vector ctr for each round, hence the link between data and identity cannot be inferred. A detailed analysis is given in Section VI.

4) Data Publication: After the sequence number assignment, each user u_i executes the *DVectorGen* algorithm to generate the obfuscated data vector, as shown in Algorithm 4.

Like the *IVectorGen* algorithm, first u_i initializes an m-dimension vector V. Then for each dimension k, if u_i's sequence number $s_i == k$, u_i uses BO$_j$'s public key pk_j to encrypt the collected data d_i, and sets $V_i[k] = c_k + \sum_{t \in T_i^{out}} G(t,k) - \sum_{t \in T_i^{in}} G(t,k)$, which means the seq_i-th dimension of V_i contains the obfuscated ciphertext, while other dimensions only contain obfuscated dummy values. Finally, u_j generates the obfuscated data vector V_i and sends it to $\mathcal{R}$.

When receiving all the vector, $\mathcal{R}$ aggregates them by executing $D = \sum_{i=1}^m V_i$, where:

$$\begin{aligned} D[k] &= \sum_{i=1}^{m} V_i[k] \\ &= c_k + \sum_{i=1}^{m} \Big(\sum_{t \in T_i^{out}} G(t,k) - \sum_{t \in T_i^{in}} G(t,k) \Big) \\ &= c_k + \sum_{i=1}^{m} \sum_{t \in T_i^{out}} G(t,k) - \sum_{i=1}^{m} \sum_{t \in T_i^{in}} G(t,k) \\ &= c_k + \sum_{1 \le a,b \le m, a \ne b} G(t_{ab},k) - G(t_{ab},k) \\ &= c_k. \end{aligned} \tag{4}$$

Hence, $\mathcal{R}$ can recover the encrypted data vector $D = (c_1, \cdots, c_m)$, and sends $\mathcal{D} = (D, TS, \sigma)$ to designated blockchain oracle BO$_j$, where TS is the current timestamp and $\sigma = \text{PKE}.Sign(sk_{\mathcal{R}}, H(D||TS))$.

When BO$_j$ receives $\mathcal{D}$ sent by $\mathcal{R}$, it first checks whether $T - TS > T_\theta$, where T and T_θ represents the current timestamp and time threshold, respectively. If $T - TS > T_\theta$ holds, $\mathcal{D}$ will be rejected as obsolete data. Then it verifies the signature by executing PKE.$Verify(pk_{\mathcal{R}}, H(D||TS))$. If the verification is successful, it then uses its private key sk_j to decrypt the ciphertext and gets m decrypted data $d_1, \cdots, d_m$, otherwise $\mathcal{D}$ will be rejected. After all the data is collected, BO$_i$ executes the Truth Discovery algorithm to calculate the truth value d_j^* as the external data that will be sent to the blockchain.

C. On-chain Data Aggregation

After calculating the truth value, BO$_j$ will send the external data to the smart contract $\mathcal{C}$. After collecting data from several external oracles, $\mathcal{C}$ also executes the TD algorithm to get the truth value d^* as the final crowdsensing result and sends rewards to all the oracle data providers and all the workers in the blockchain network.

In this scenario, there exist two kinds of BO: provider and worker. The difference is that the provider sends the calculated truth value as external data to the blockchain, while the worker participates in the consensus of the whole blockchain network. Note that a BO can be both provider and worker, and we assume that the number of providers and workers is p and n, respectively, where $p \le n$. To protect the provider's data privacy, the data should be encrypted before the smart contract $\mathcal{C}$ publishes the final aggregated result. Under these circumstances, we propose a novel on-chain data aggregation mechanism based on the PCTD cryptosystem. The detailed aggregation protocol is shown in Protocol 1.

Protocol 1 On-chain Data Aggression

Step 1: The smart contract $\mathcal{C}$ gathers the encrypted data $C^* = \{c_1^*, \cdots, c_p^*\}$ collected by the corresponding p providers $P = \{\mathsf{addr}_1^{\mathsf{p}}, \cdots, \mathsf{addr}_{\mathsf{p}}^{\mathsf{p}}\}$, and sends a blockchain transaction to submit a consensus request $\mathcal{R}_{\mathcal{C}} = (C^*, P, \mathsf{fee})$, where fee is the total financial cost (including transaction fee and crowdsensing reward) given by the requester.
Step 2: During the consensus procedure, each worker $\mathsf{addr}_{\mathsf{j}}^{\mathsf{w}}$ uses its partial private key psk_j to partially decrypt each $c_i^* \in C^*$ by executing PCTD.$PDec(c_i^*, psk_j)$, and gets the partially decrypted result c_i^j. Finally, BO$_j$ broadcasts the consensus information $C^j = \{c_1^j, \cdots, c_n^j\}$ to other oracles.
Step 3: The committer $\mathcal{O}$ collects R and all the C^j that are received (noted as $C' = \{C^1, \cdots, C^k\}$) by workers (noted as $W = \{\mathsf{addr}_1^{\mathsf{w}}, \cdots, \mathsf{addr}_{\mathsf{k}}^{\mathsf{w}}\}$), and packs them to a new block B that will be added to the blockchain.
Step 4: $\mathcal{C}$ parses the information in B and computes all the decrypted data $D^* = \{d_1^*, \cdots, d_n^*\}$, where $d_i^* =$ PCTD.$TDec(c_i^1, \cdots, c_i^k) \in D$.
Step 5: $\mathcal{C}$ executes TD algorithm and gets the final truth value d^* as the crowdsensing result, and the weight set $\{w_1, \cdots, w_p\}$ associated to each provider $\{\mathsf{addr}_1^{\mathsf{p}}, \cdots, \mathsf{addr}_{\mathsf{p}}^{\mathsf{p}}\}$. The result d^* will be sent to the requester and the financial cost fee will be fairly distributed to every participants via **Protocol 2**.

After the *Vehicular Data Aggression* process, each provider $\mathsf{addr}_{\mathsf{j}}^{\mathsf{p}}$ sends the encrypted data c_j^* to $\mathcal{C}$, where $c_j^* =$ PCTD.$Enc(pk, d_j^*)$. Then $\mathcal{C}$ generates the consensus request $\mathcal{R}_{\mathcal{C}}$ and sends it to the blockchain network, as shown in **Step 1**.

Step 2 and **Step 3** represents the blockchain consensus process, which is based on the Byzantine Fault Tolerance (BFT). However, unlike the traditional BFT-based solution, we introduce the *Partial Decryption* operation in the consensus process. More specifically, each worker $\mathsf{addr}_{\mathsf{j}}^{\mathsf{w}}$ partially decrypts all the ciphertexts and broadcasts them to other oracles. Like the traditional consensus mechanism, the broadcast consensus information is signed by $\mathsf{addr}_{\mathsf{j}}^{\mathsf{w}}$'s private key sk_j so that adversaries cannot generate fake consensus information. Then the commit node $\mathcal{O}$ (the commit node selection mechanism is based on the BFT consensus mechanism) collects the partially decrypted result of k workers, where $2n/3 < k \le n$. The left $<$ holds to satisfy the BFT principle, and the right $<$ holds because of the random failure for several nodes or network delay.

In **Step 4**, $\mathcal{C}$ parses the block information and aggregates the partially decrypted result. Note that if $t \le k$ holds, where t is the decryption threshold in PCTD, all the original data $d_i \in D$ can be successfully recovered. In PRVB, we simply set $t = \lceil 2n/3 \rceil$. Finally, in **Step 5**, $\mathcal{C}$ gets the crowdsensing result d^* which guarantees the reliability via TD algorithm, and the rewards will be distributed to every provider and worker based on their contributions, as shown in Protocol 2.

Protocol 2 Reward Distribution

Step 1: $\mathcal{C}$ computes $sum = \sum_{i=1}^{p} w_i$, $\mathsf{fee}_p = \alpha \cdot \mathsf{fee}$, and $\mathsf{fee}_w = (1-\alpha) \cdot \mathsf{fee}$, where $0 < \alpha < 1$ is the reward factor.
Step 2: For each provider $\mathsf{addr}_{\mathsf{i}}^{\mathsf{p}} \in P$, $\mathcal{C}$ distributes the provider reward $\mathsf{fee}_p \cdot w_i / sum$ to $\mathsf{addr}_{\mathsf{i}}^{\mathsf{p}}$.
Step 3: For each worker $\mathsf{addr}_{\mathsf{i}}^{\mathsf{w}} \in W$, $\mathcal{C}$ distributes the worker reward fee_w / p to $\mathsf{addr}_{\mathsf{i}}^{\mathsf{w}}$.

Protocol 2 illustrates our proposed reward distribution protocol that achieves fair incentives. The fee will be partitioned into two kinds of rewards: *provider's reward* $\mathsf{fee}_{\mathsf{p}}$ and *worker's reward* $\mathsf{fee}_{\mathsf{w}}$, by a pre-defined reward factor α. In order to achieve fairness, $\mathsf{fee}_{\mathsf{p}}$ is distributed by the contributions (i.e., the weight during the TD process), and $\mathsf{fee}_{\mathsf{w}}$ will be distributed equally because all the works do the same work (i.e., partial decryption operation).

VI. Security Analysis

In this section, we make a thorough security analysis to show that the proposed scheme PRVB achieves privacy protection, reliability, and fairness.

A. Privacy Protection

For users, we give a sketch proof to show that PRVB can provide the identity privacy un-linkability between the user's identity and its collected data even if RSU colludes with $m-2$ malicious users. For blockchain oracles, we prove that BO's data privacy is also well protected.

1) Identity Privacy & Un-linkability: During the vehicular data aggregation process, each user u_i uses the shared key sets $S_i^{in}, S_i^{out}, T_i^{in}, T_i^{out}$ to obfuscate the link between the collected data and u_i's identity. For the honest-but-curious RSU $\mathcal{R}$, in the *Prefix Partition* and *Assigning Sequence Number* process, $\mathcal{R}$ only knows the temporary prefix set tmp and the corresponding counter vector ctr during each interaction. As observed from Fig. 2, in round 1, $\mathcal{R}$ only knows that there are three users that match the prefix $0*^4$ and two users that match $1*^4$, however, it does not know which user matches the corresponding prefix set. Each user u_i generates an obfuscated identity vector v_i, where $v_i[k] = b_{ik} + \sum_{s \in S_i^{out}} F(s,k) - \sum_{s \in S_i^{in}} F(s,k)$, $0 \leq k < |tmp|$ so that $\mathcal{R}$ cannot distinguish whether b_{ik} is 0 or 1 if the pseudo-random function F is secure. Hence, the sequence number of each user cannot be inferred. Similarly, during the *Data Publication* process, each user u_i generates an obfuscated data vector $V_i[k] = c_k + \Phi$ or $V_i[k] = \Phi$, where $\Phi = \sum_{t \in T_i^{out}} G(t,k) - \sum_{t \in T_i^{in}} G(t,k)$, $0 \leq k < n$. If the pseudo-random function G is secure, $\mathcal{R}$ cannot distinguish which $V_i[k]$ contains the data c_k and which $V_i[k]$ only consists of several obfuscated values. Hence, the link between user's data and identity cannot be inferred either.

We also assume the worst-case scenario that $\mathcal{R}$ colludes with at most $m-2$ malicious users, and they try to link the data between two victim u_a and u_b's identity. In this scenario, the shared keys between u_a and u_b (i.e., $s_{ab}, s_{ba}, t_{ab}, t_{ba}$) cannot be compromised to adversaries. Then the obfuscated identity vector $v_a[k]$ and $v_b[k]$ can be recovered as $b_{ak} + F(s_{ab},k) - F(s_{ba},k)$ and $b_{bk} + F(s_{ba},k) - F(s_{ab},k)$, respectively. If F is secure, without knowing s_{ab} and s_{ba}, only $b_{ak} + b_{bk}$ can be recovered. More specifically, if $b_{ak} + b_{bk} = 1$, the sequence order relationship between u_a and u_b cannot be inferred because in adversaries view, the probability $Pr[b_{ak} = 1] = Pr[b_{bk} = 1] = 1/2$. If $b_{ak} + b_{bk} = 0$ or 2, the result only shows that the two corresponding random numbers r_a and r_b match (or do not match) the same prefix set. Hence the relationship between u_a and u_b cannot be inferred either. Similarly, assume u_a and u_b's sequence number is k_1 and k_2, respectively, the obfuscated data vector $V_a[k_1]$, $V_b[k_1]$ and $V_a[k_2]$, $V_b[k_2]$ will be recovered as:

$$\begin{aligned} V_a[k_1] &= c_{k_1} + G(t_{ab}, k_1) - G(t_{ba}, k_1), \\ V_b[k_1] &= G(t_{ba}, k_1) - G(t_{ab}, k_1), \\ V_a[k_2] &= G(t_{ab}, k_2) - G(t_{ba}, k_2), \\ V_b[k_2] &= c_{k_2} + G(t_{ba}, k_2) - G(t_{ab}, k_2), \end{aligned} \tag{5}$$

respectively. From Eq. 5, if G is secure, without knowing t_{ab} and t_{ba}, adversaries cannot distinguish which element contains the collected data. Hence, under this circumstance, the sequence number of u_a and u_b, and the link between data and the corresponding identity cannot be inferred even if $\mathcal{R}$ collude with at most $n-2$ malicious users.

2) Data Privacy: During the *Vehicular Data Aggregation* process, each user u_i uses BO_j's public key pk_j to encrypt the data d_i and uploads the obfuscated encrypted result c_i to it via $\mathcal{R}$. $\mathcal{R}$ pre-aggregates the vector $C = \{c_1, \cdots, c_m\}$ with m encrypted values. If the PKE cryptosystem used in the blockchain network (because pk_j is generated by blockchain, as mentioned before) is semantically secure, each $c \in C$ cannot be recovered without knowing the private key sk_j. Hence, during this process, the privacy of all the collected data is well protected.

During the *On-chain Data Aggregation* process, each provider BO_j uses PCTD public key pk to encrypt the data d_j^* and uploads the encrypted result c_j^* to $\mathcal{C}$. Due to the semantic security of PCTD cryptosystem [33], there are only two ways for adversaries to recover d_j^*: 1) using sk held by TA, which is impossible because TA cannot be compromised in our threat model, and 2) performing threshold decryption operation via consensus process, where all provider's data will be published. Note that some malicious providers will steal another oracle's uploaded data before, as mentioned earlier, hence it is not necessary to perform such an attack when the consensus begins. Therefore, the privacy of each provider's uploaded data is well protected before the threshold decryption begins.

B. Reliability

Blockchain network can achieve reliable storage because of its decentralization and immutability, as once the data is stored on the blockchain, it cannot be deleted or modified. Besides, the smart contract realizes reliable program execution on the blockchain, which means the contract code can be honestly executed. Hence, both all the on-chain data and the smart contract $\mathcal{C}$ guarantees the reliability of data storage and TD algorithm.

During the threshold decryption process, we set the decryption threshold $t = \lceil 2n/3 \rceil$ which makes sure that the ciphertexts can be recovered successfully due to the BFT environments, where at least $2n/3$ nodes are working honestly. Hence, if the condition of BFT consensus is guaranteed, all the collected data will be recovered successfully.

The TD algorithm used in both *Vehicular Data Aggregation* and *On-chain Data Aggregation* process is to achieve reliable data collection by users and providers, respectively. Under the TD algorithm, malicious values that deviates far from the estimated truth value will be given a low weight, which means that this kind of data cannot disturb the final result d^*. Hence, the fake data sent by malicious users and providers will be filtered or ignored during the two TD processes.

C. Fairness

The Protocol 2 presented in this scheme achieves financial fairness, where the total rewards fee is partitioned into fee_p for providers and fee_w for workers by a reward factor $0 < \alpha < 1$. Note that fee_p is distributed based on the weight of each participating provider, which means if the provider's data quality is higher (i.e., the corresponding weight is higher), it will get higher rewards. Besides, fee_w is equally distributed because all the workers who participated in this consensus round provide the partially decrypted results. Furthermore, for the requester who pays the fee to get the crowdsensing result, as is mentioned above, PRVB achieves reliable crowdsensing with the help of blockchain and TD, hence if the requester pays fees, it will get the correct crowdsensing result. Therefore, we can conclude that the proposed PRVB scheme achieves fairness for both requesters and blockchain oracles.

VII. Experimental Evaluation

To evaluate the computational complexity and communication complexity, we make a thorough experimental evaluation based on the Hyperledger Fabric blockchain platform to show that the proposed PRVB scheme has both high computational and communicational efficiency.

A. Experimental Setup

We implemented the proposed PRVB scheme which consists of three modules: *System Initialization*, *Vehicular Data Aggregation* and *On-chain Data Aggregation*. Note that the *On-chain Data Aggregation* operation is executed on the blockchain platform, hence we deploy it in Hyperledger Fabric 1.4 because of its high efficiency and flexible programmability. We also implemented the Truth Discovery algorithm [28] and PCTD cryptosystem [33] used in PRVB. All programs are written in Golang and compiled on Go 1.13.6 x64 environment.

We set the security parameter $\lambda = 256$ and the prefix length $w = 8$. We use HMAC to instantiate two pseudo-random functions F, G and use SHA256 to instantiate the hash function H. We use ECC and ECDSA on the secp256k1 curve to instantiate the encryption and signature primitive of the PKE cryptosystem, respectively. The sensing data for each user is randomly generated and follows a Normal Distribution with $\mu = 10$ and $\sigma^2 = 0.25$. For TD process, we set the maximum iteration times $iter_{max} = 100$.

The *System Initialization* module is running on a server with an Intel i7-9700K CPU, 16G RAM, and 64-bit Ubuntu 16.04 operating system. Each BO is running on a virtualized server with an Intel Xeon CPU, 8G RAM, and 64-bit Ubuntu 16.04 operating system. The clients of PRVB (i.e., users and RSU) are running on a Tablet PC with an Intel i5-7300U processor with 8GB RAM and 64-bit Ubuntu 16.04 operating system.

To evaluate the influence between the efficiency and the number of users/BOs, we simulate two different scenarios. More specifically, **Scenario 1** is used to evaluate the efficiency of *System Initialization* and *Vehicular Data Aggregation* processes. In this scenario, the number of BOs is set as $n = 4$, and the number of users varies from $m = 10$ to $m = 100$ with increments of 10. **Scenario 2** is used to evaluate the efficiency of *System Initialization* and *On-chain Data Aggregation* processes. In this scenario, the number of users is set as $m = 10$, and the number of BOs varies from $n = 4$ to $n = 20$ with increments of 4. Note that we only set the maximum of n to be 20 because if we set a larger n, the efficiency of the Fabric blockchain platform will become unacceptable. Besides, we use a fixed number of RSU (i.e., $r = 1$) to pre-aggregate the vehicular sensing data in both scenarios. The simulation for each scenario is executed 5 times and the average execution time is considered for evaluation.

B. Time Complexity

A systematic computational complexity analysis of the two simulated scenarios is conducted to prove that the proposed PRVB is computationally efficient.

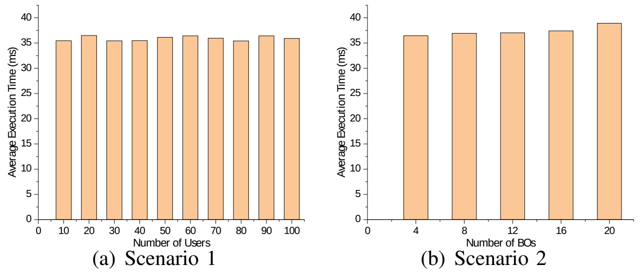

Fig. 3: Time performance for system initialization.

1) System Initialization: In this process, TA first generates system parameters and deploys the smart contract $\mathcal{C}$. Then it registers for each RSU and each BO and finally makes key distributions. The time complexity of *System Initialization* process is $O(r+n)$, where r and n represent the number of RSUs and users, respectively. Note that: 1) The time complexity is unrelated to the number of users n because users do not register with TA for privacy concerns, and 2) we set $r = 1$ as a fixed value, hence the time complexity can be further simplified to $O(n)$.

Fig. 3(a) shows that the execution time is irregular as m increases, hence it proves that the computational costs are unrelated to the number of users. Fig. 3(b) shows that with the number of BO increases, the execution time also slightly increases.

Consequently, the computational cost of *System Initialization* is dependent on the number of registered BOs, and it is negligible in practical scenarios because it only executes once.

2) Vehicular Data Aggregation: In this process, each user should generate shared keys with $m-1$ users within a group, and the total number of shared keys for each user is $4(m-1)$. Then a unique sequence number will be assigned for each user during $\log m$ interactions with RSU, and for each interaction, an obfuscated identity vector that used $2(m-1)$ shared keys will be generated. Next, each user generates an m-dimension obfuscated data vector that also uses $2(m-1)$ shared keys for each dimension, and sends it to RSU which pre-aggregates the vector and sends it to the corresponding BO. Finally, BO decrypts all the data and executes the TD algorithm to obtain the truth value. The time complexity of *Vehicular Data Aggregation* is $O(m^2)$.

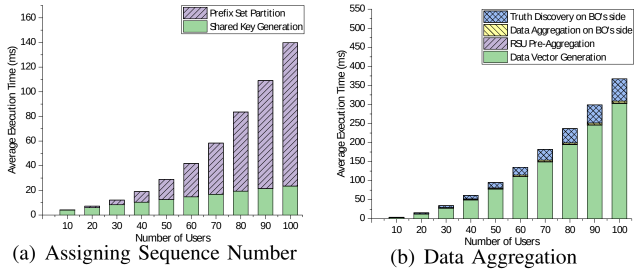

Fig. 4: Time performance for vehicular data aggregation.

The whole process of *Vehicular Data Aggregation* consists of two sub-processes: *Assigning Sequence Number* and *Data*

Aggregation. As can be seen in Fig. 4(a) and Fig. 4(b), as m increases, the time cost of these two sub-processes (of which the time complexity is $O(m \cdot \log m)$ and $O(m^2)$, respectively) increases significantly. We first notice that the most time-consuming operation in the whole process is data vector generation, however, it only costs less than 350ms when the number of users reaches 100, which is acceptable in real-world vehicular crowdsensing scenarios where the number of users is not relatively high. Besides, some computations such as assigning sequence numbers and addition/subtraction of shared keys can be done only once, which can further improve the efficiency during the following aggregation processes. The lest two time-consuming operations are two aggregation operations which only costs less than 6.5ms when $m = 100$. We also notice that as m increases, the execution time of the TD process also increases, which is also reasonable. Note that we omit the analysis of communications delay between these entities because it is negligible compared with the computation time costs.

Consequently, the computational cost of *Vehicular Data Aggregation* is dependent on the number of users, and it is also efficient in practical scenarios.

3) On-chain Data Aggregation: In this process, we assume that each BO participates in the blockchain as both provider and worker. Hence, each BO participates in the consensus process where n threshold decryption operations should be performed. After the consensus, the smart contract $\mathcal{C}$ executes TD algorithm to obtain the aggregated result. The time complexity of *On-chain Data Aggregation* is $O(n)$.

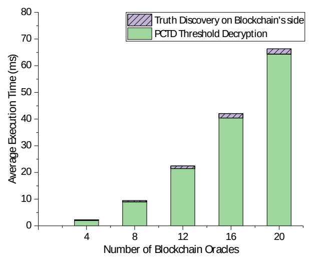

Fig. 5: On-chain data aggregation.

As is illustrated in Fig. 5, as n increases, the execution time also increases, and the maximum time cost under $n = 20$ is lower than 70ms, which is also efficient. Note that we omit the analysis of communications delay and consensus time cost of blockchain because it depends on the blockchain platform itself. The time costs of communication delay and consensus time of blockchain in our experiment are less than 100ms and 1s, respectively, which is efficient compared with other blockchains like Bitcoin and Ethereum.

Consequently, the computational cost of *On-chain Data Aggregation* is dependent on the number of BOs. The evaluation based on Hyperledger Fabric proves the efficiency in real-world blockchain-based scenarios.

C. Communication Complexity

The communication complexity represents the bit changed between two entities. During the *Vehicular Data Aggregation* process, for each user, $m - 1$ keys will be shared by the Diffle-Hellman mechanism, $\log m$ obfuscated identity vectors on average, and an m-dimension obfuscated data vector will be sent to RSU. For RSU, it sends the aggregated vector to the designated BO. The communication complexity of each user is $O(m \cdot \log m)$ at the sequence number generation process (that will be executed only once), and $O(m)$ at the aggregation process. The communication complexity of RSU and BO are both $O(m)$. During the *On-chain Data Aggregation* process, each BO should participate in the consensus process, where it will partially decrypt n ciphertexts and broadcast the decrypted result to other $n-1$ BOs. Hence the communication complexity of each BO is $O(n^2)$.

In our experiment, when $m = 100$, the communication costs of each user during the sequence number generation and aggregation process are only 20.76KB and 25KB, respectively. The communication costs of RSU and BO are both 25KB. When $n = 20$, the additional communication cost of each BO during one consensus process is only 100KB. Different from user and RSU, BO has high bandwidth hence the 100KB additional communication cost is negligible in real-world scenarios.

D. Summary

Table II shows both time and communication complexity, and the above analysis shows that the proposed PRVB scheme achieves both computational and communicational efficiency. Therefore, PRVB is practical in real-world vehicular crowdsensing scenarios.

VIII. Conclusion

In this paper, we propose PRVB that aims to achieve reliable and privacy-preserving vehicular crowdsensing. More specifically, we proposed a novel system model based on blockchain oracle that guarantees the reliability of sensed data before it is published to the blockchain. We also presented a privacy-preserving vehicular data aggregation scheme that makes use of a pruned prefix binary tree to generate a unique sequence number for each user without TA's participant to achieve un-linkability between data and the corresponding identity. Besides, we designed an incentive mechanism for BOs to achieve fair rewards based on data quality and participant level. Theoretical analysis and experimental evaluation prove that the proposed PRVB scheme can well protect the privacy of users and BOs with high computational and communicational efficiency.

References

[1] W. Alasmary, H. Sadeghi, and S. Valaee, "Crowdsensing in vehicular sensor networks with limited channel capacity," in *Proceedings of IEEE International Conference on Communications, ICC 2013, Budapest, Hungary, June 9-13, 2013*, 2013, pp. 1833–1838.

[2] F. Campioni, S. Choudhury, K. Salomaa, and S. G. Akl, "Improved recruitment algorithms for vehicular crowdsensing networks," *IEEE Trans. Vehicular Technology*, vol. 68, no. 2, pp. 1198–1207, 2019.

[3] L. Pu, X. Chen, G. Mao, Q. Xie, and J. Xu, "Chimera: An energy-efficient and deadline-aware hybrid edge computing framework for vehicular crowdsensing applications," *IEEE Internet of Things Journal*, vol. 6, no. 1, pp. 84–99, 2019.

TABLE II: Time and Communication Complexity of PRVB.

Process	Entity	System Initialization	Vehicular Data Aggregation	On-chain Data Aggregation
Time Complexity	TA	$O(n)$	-	-
	User	-	$O(m^2)$	-
	RSU	-	$O(m)$	-
	BO	-	$O(m)$	$O(n)$
Communication Complexity	User	-	$O(m \cdot \log m)$ / $O(m)$	-
	RSU	-	$O(m)$	-
	BO	-	$O(m)$	$O(n^2)$

[4] C. Zhang, L. Zhu, C. Xu, K. Sharif, C. Zhang, and X. Liu, "PGAS: privacy-preserving graph encryption for accurate constrained shortest distance queries," *Inf. Sci.*, vol. 506, pp. 325–345, 2020.

[5] C. Zhang, L. Zhu, C. Xu, C. Zhang, K. Sharif, H. Wu, and H. Westermann, "BSFP: blockchain-enabled smart parking with fairness, reliability and privacy protection," *IEEE Transactions on Vehicular Technology*, pp. 1–1, 2020.

[6] J. Ni, A. Zhang, X. Lin, and X. S. Shen, "Security, privacy, and fairness in fog-based vehicular crowdsensing," *IEEE Communications Magazine*, vol. 55, no. 6, pp. 146–152, 2017.

[7] L. Zhu, C. Zhang, C. Xu, and K. Sharif, "Rtsense: Providing reliable trust-based crowdsensing services in CVCC," *IEEE Network*, vol. 32, no. 3, pp. 20–26, 2018.

[8] L. Zhu, C. Zhang, C. Xu, X. Du, N. Guizani, and K. Sharif, "Traffic monitoring in self-organizing vanets: A privacy-preserving mechanism for speed collection and analysis," *IEEE Wireless Commun.*, vol. 26, no. 6, pp. 18–23, 2019.

[9] C. Zhang, L. Zhu, C. Xu, X. Liu, and K. Sharif, "Reliable and privacy-preserving truth discovery for mobile crowdsensing systems," *IEEE Transactions on Dependable and Secure Computing*, pp. 1–1, 2019.

[10] J. Adler, R. Berryhill, A. G. Veneris, Z. Poulos, N. Veira, and A. Kastania, "Astraea: A decentralized blockchain oracle," in *IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData), iThings/GreenCom/CPSCom/SmartData 2018, Halifax, NS, Canada, July 30 - August 3, 2018*, 2018, pp. 1145–1152.

[11] R. Lu, K. Heung, A. H. Lashkari, and A. A. Ghorbani, "A lightweight privacy-preserving data aggregation scheme for fog computing-enhanced iot," *IEEE Access*, vol. 5, pp. 3302–3312, 2017.

[12] C. Huang, D. Liu, J. Ni, R. Lu, and X. Shen, "Reliable and privacy-preserving selective data aggregation for fog-based iot," in *2018 IEEE International Conference on Communications, ICC 2018, Kansas City, MO, USA, May 20-24, 2018*, 2018, pp. 1–6.

[13] X. Gong, Q. Hua, L. Qian, D. Yu, and H. Jin, "Communication-efficient and privacy-preserving data aggregation without trusted authority," in *2018 IEEE Conference on Computer Communications, INFOCOM 2018, Honolulu, HI, USA, April 16-19, 2018*, 2018, pp. 1250–1258.

[14] L. Zhu, M. Li, Z. Zhang, C. Xu, R. Zhang, X. Du, and N. Guizani, "Privacy-preserving authentication and data aggregation for fog-based smart grid," *IEEE Communications Magazine*, vol. 57, no. 6, pp. 80–85, 2019.

[15] X. Shen, L. Zhu, C. Xu, K. Sharif, and R. Lu, "A privacy-preserving data aggregation scheme for dynamic groups in fog computing," *Inf. Sci.*, vol. 514, pp. 118–130, 2020.

[16] M. Yang, T. Zhu, K. Liang, W. Zhou, and R. H. Deng, "A blockchain-based location privacy-preserving crowdsensing system," *Future Generation Comp. Syst.*, vol. 94, pp. 408–418, 2019.

[17] K. Zhao, S. Tang, B. Zhao, and Y. Wu, "Dynamic and privacy-preserving reputation management for blockchain-based mobile crowdsensing," *IEEE Access*, vol. 7, pp. 74 694–74 710, 2019.

[18] H. Duan, Y. Zheng, Y. Du, A. Zhou, C. Wang, and M. H. Au, "Aggregating crowd wisdom via blockchain: A private, correct, and robust realization," in *2019 IEEE International Conference on Pervasive Computing and Communications, PerCom, Kyoto, Japan, March 11-15, 2019*, 2019, pp. 1–10.

[19] J. Wang, M. Li, Y. He, H. Li, K. Xiao, and C. Wang, "A blockchain based privacy-preserving incentive mechanism in crowdsensing applications," *IEEE Access*, vol. 6, pp. 17 545–17 556, 2018.

[20] C. Cai, Y. Zheng, and C. Wang, "Leveraging crowdsensed data streams to discover and sell knowledge: A secure and efficient realization," in *38th IEEE International Conference on Distributed Computing Systems, ICDCS 2018, Vienna, Austria, July 2-6, 2018*, 2018, pp. 589–599.

[21] S. Hu, C. Cai, Q. Wang, C. Wang, X. Luo, and K. Ren, "Searching an encrypted cloud meets blockchain: A decentralized, reliable and fair realization," in *2018 IEEE Conference on Computer Communications, INFOCOM 2018, Honolulu, HI, USA, April 16-19, 2018*, 2018, pp. 792–800.

[22] S. Biswas, K. Sharif, F. Li, B. Nour, and Y. Wang, "A scalable blockchain framework for secure transactions in IoT," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 4650–4659, June 2019.

[23] S. Nakamoto. (2008) Bitcoin: A peer-to-peer electronic cash system. [Online]. Available: https://bitcoin.org/bitcoin.pdf

[24] G. Wood. (2014) Ethereum: A secure decentralized generalized transaction ledger. [Online]. Available: http://gavwood.com/paper.pdf

[25] M. Castro and B. Liskov, "Practical byzantine fault tolerance," in *Proceedings of the Third USENIX Symposium on Operating Systems Design and Implementation (OSDI), New Orleans, Louisiana, USA, February 22-25, 1999*, 1999, pp. 173–186.

[26] Hyperledger fabric. [Online]. Available: https://www.hyperledger.org/projects/fabric

[27] S. Biswas, K. Sharif, F. Li, S. Maharjan, S. P. Mohanty, and Y. Wang, "PoBT: A lightweight consensus algorithm for scalable IoT business blockchain," *IEEE Internet of Things Journal*, vol. 7, no. 3, pp. 2343–2355, mar 2020.

[28] Q. Li, Y. Li, J. Gao, B. Zhao, W. Fan, and J. Han, "Resolving conflicts in heterogeneous data by truth discovery and source reliability estimation," in *International Conference on Management of Data, SIGMOD 2014, Snowbird, UT, USA, June 22-27, 2014*, 2014, pp. 1187–1198.

[29] X. Tang, C. Wang, X. Yuan, and Q. Wang, "Non-interactive privacy-preserving truth discovery in crowd sensing applications," in *2018 IEEE Conference on Computer Communications, INFOCOM 2018, Honolulu, HI, USA, April 16-19, 2018*, 2018, pp. 1988–1996.

[30] C. Zhang, L. Zhu, C. Xu, K. Sharif, X. Du, and M. Guizani, "LPTD: achieving lightweight and privacy-preserving truth discovery in ciot," *Future Generation Comp. Syst.*, vol. 90, pp. 175–184, 2019.

[31] C. Zhang, L. Zhu, C. Xu, K. Sharif, and X. Liu, "PPTDS: A privacy-preserving truth discovery scheme in crowd sensing systems," *Inf. Sci.*, vol. 484, pp. 183–196, 2019.

[32] G. Xu, H. Li, S. Liu, M. Wen, and R. Lu, "Efficient and privacy-preserving truth discovery in mobile crowd sensing systems," *IEEE Trans. Vehicular Technology*, vol. 68, no. 4, pp. 3854–3865, 2019.

[33] X. Liu, K. R. Choo, R. H. Deng, R. Lu, and J. Weng, "Efficient and privacy-preserving outsourced calculation of rational numbers," *IEEE Trans. Dependable Sec. Comput.*, vol. 15, no. 1, pp. 27–39, 2018.

[34] J. Katz and Y. Lindell, *Introduction to Modern Cryptography, Second Edition*. CRC Press, 2014.

[35] W. Diffie and M. E. Hellman, "New directions in cryptography," *IEEE Trans. Information Theory*, vol. 22, no. 6, pp. 644–654, 1976.

Dear Editor,

Enclosed please find our manuscript: “PRVB: Achieving Privacy-Preserving and Reliable Vehicular Crowdsensing via Blockchain Oracle”, by Can Zhang, Liehuang Zhu, Chang Xu, and Kashif Sharif to be submitted to IEEE Transactions on Vehicular Technology. All authors have seen and agree with the contents of the manuscript, and certify that the submission is original work and is not under review at any other publication. We hope that the editors and reviewers will agree on the interest of this study. If you have any question, do not hesitate to contact us.

Vehicular crowdsensing is attracting more and more attention because of its wide sensing coverage and diverse usage in smart cities. However, privacy issues that stem from traditional vehicular crowdsensing scenarios, violate the participant's privacy. Although some privacy-preserving schemes have been designed that aim to protect the sensitive information of sensed data, the reliability cannot be guaranteed because of the system's centralized structure. The introduction of blockchain in crowdsensing applications provides reliable data storage, however, the reliability of data sources remains an open challenge. Under these circumstances, the crowdsourcing service requester may not be able to obtain quality data. To solve these problems, we propose a novel Privacy-preserving and Reliable Vehicular crowdsensing via Blockchain oracle, called PRVB. More specifically, a privacy-preserving vehicular data aggregation scheme is presented to protect the data privacy and unlinkability between participant vehicles and sensed data. Besides, two protocols are designed to protect data privacy and to achieve fair rewards for data providers. Thorough theoretical analysis and experimental evaluations have proved that the proposed PRVB achieves privacy protection, reliability, and fairness with significant computational & communicational efficiency.

Sincerely yours,

Chang Xu on behalf of all co-authors

Beijing Institute of Technology, Beijing, China

xuchang@bit.edu.cn