Towards Privacy-Preserving Decentralized Reputation Management for Vehicular Crowdsensing

Zhongkai Lu, Lingling Wang(✉), Ke Geng, Jingjing Wang, and Lijun Sun

School of Information and Science Technology, Qingdao University of Science and Technology, No. 99 Songling Road, Qingdao 266061, Shandong, China
{luzhongkai,gengke}@mails.qust.edu.cn,
{wanglingling,wangjingjing,lijunsun}@qust.edu.cn

Abstract. The reputation of a vehicle is a critical role in most vehicular crowdsensing applications, which incentivizes vehicles to perform crowdsensing tasks by submitting high-quality data and getting remunerated accordingly. Unfortunately, existing centralized reputation systems are vulnerable to collusion attacks, and decentralized approaches are susceptible to Sybil attacks. What's worse, both of them have privacy leakage and fairness problems. To address these issues, we take advantage of various cryptographic primitives and the blockchain technology to present a privacy-preserving decentralized reputation management system. Specifically, a compact traceable ring signature is proposed to provide identity privacy protection and resist Sybil attacks. To ensure fairness, the quantification of data quality is fulfilled by combining the rating feedback mechanism with comprehensive updating factors. Additionally, our system allows the reputation update automatically through smart contracts deployed on the consortium blockchain. The authenticity of the reputation can be verified by a zero-knowledge proof when a vehicle shows its reputation. Finally, a proof-of-concept prototype system by Parity Ethereum is presented. Extensive security analysis and implementations demonstrate the feasibility and efficiency of the proposed system.

Keywords: Vehicular crowdsensing · Reputation management · Privacy-preservation · Fairness

1 Introduction

Vehicular crowdsensing (VCS) [1] is an emerging paradigm where vehicles use onboard sensors to collect and share real time traffic information [2] without establishing extra dedicated infrastructure, which can help drivers to improve users' driving experiences and offer other services on roads. Due to these benefits, some practical VCS applications have emerged [3]. In a VCS application,

Z. Lu, K. Geng, J. Wang and L. Sun—Contributed equally.

C. Li et al. (Eds.): APPT 2023, LNCS 14103, pp. 210–240, 2024.
https://doi.org/10.1007/978-981-99-7872-4_13

reputation systems are used to maintain and update the reputation value, which is usually the benchmark for reliable worker selection, rewards calculation [4], and user-level classification, etc. Therefore, a well-designed reputation system is essential for VCS applications.

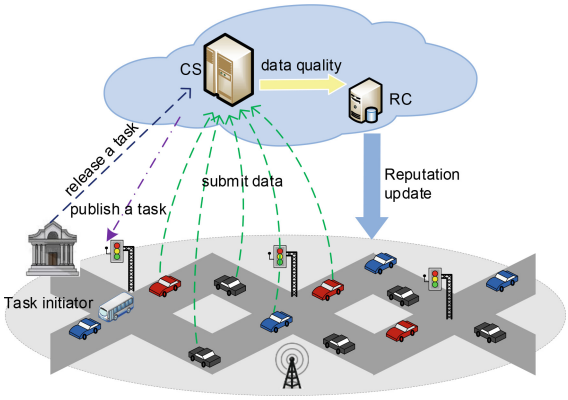

Fig. 1. A centralized reputation system in vehicular crowdsensing

Existing reputation systems in VCS are mainly divided into two types: centralized and decentralized systems. As shown in Fig. 1, in a centralized system, vehicles are assigned with the reputation according to their past behaviours in the former crowdsensing tasks. A central server CS is in charge of sensing tasks distribution and data collection. A reputation center RC is responsible for storing and updating the reputation. Although centralized reputation systems provide some benefits and conveniences, they simultaneously suffer from a single point of failure and collusion attacks. Specifically, RC is the Achilles's heel of the reputation systems. Dishonest vehicles may collude with compromised RC to increase their reputations illegally, or take advantage of system's vulnerabilities to keep large reputations even when they provide poor-quality data [5], thereby incurring fairness problem and revenue losses to VCS applications.

Blockchain, as the most popular distributed technology, has enabled a decentralized reputation system. Although blockchain originally acts as a fundamental technology in Bitcoin, it has been recently adopted in many domains, such as Artificial Intelligence [6], Internet of Things [7], etc. We could achieve a public and tamper-resistant record of the reputation as well as the open access to the reputation by using blockchain. However, existing blockchain-based reputation systems [8–11], do not apply well to the VCS scenario. Specifically, most existing systems are put forward under E-commerce environment, in which the reputation is evaluated in different ways. Besides, in the existing works, almost all reputation opinions and interaction histories are stored on the blockchain, and most interactions of the system are performed via the consensus protocol, where consensus efficiency is a problem that affects communication efficiency. In a VCS

scenario, it is inefficient and impractical for vehicles to frequently access the blockchain. So, it is highly desirable to develop a decentralized reputation system that minimizes the frequency of access to the blockchain. Due to the "open" nature, another problem the decentralized system faces is the Sybil attacks [12]. To avoid Sybil attacks, identity authentication is essential to perform each time when a vehicle submits a message. However, public key authentication might lead to privacy disclosure (e.g., the drivers' identity, driving speed, and driving path), which may affect the vehicle's willingness to participate in a task.

As a consequence, these motivate our research problem to be "*how to evaluate and update the reputation with fairness in a VCS scenario without revealing any privacy of participating vehicles*?"

Though important, it is still a non-trivial task to solve the above challenges in a decentralized system manipulated by untrusted vehicles and attakers. First, authentication in a VCS scenario is usually done by edge nodes, such as roadside units (RSUs), which are usually curious about the privacy of vehicles. Moreover, the reputation is attached to the identity of the vehicle, so recording all reputations on the ledger directly will also result in privacy disclosure. Second, it is not easy to quantify the data quality fairly, which is the baseline of evaluating the reputation. Third, it is a challenge to guarantee trusted update of the reputation while reducing interactions with the blockchain.

To address the privacy issue, anonymous mechanisms, i.e. vehicles generate multiple pseudonyms or anonymous credentials, can protect the vehicles' privacy. However, simply leveraging pseudonyms for vehicle's anonymity cannot resolve this issue, which is vulnerable to de-anonymization attacks [13]. Moreover, anonymous credentials are often issued by a trusted party, and may incur a lot of additional computation and storage overhead [14], which are not suitable to our system. Hence, we present a privacy-preserving mechanism to protect vehicles' privacy while resisting sybil attacks. Moreover, we accomplish a trustworthy anonymous record on the blockchain and apply a zero-knowledge proof to bind the reputation to a specific vehicle.

To address the fairness issue, it is necessary to update the reputation according to the data quality impartially, which requires that the system knows whether the data provided by a vehicle is authentic or not and to what extent is it accurate. Rating feedback mechanisms [15] is a straightforward solution to quantify data quality, which allow other users in the proximity to provide a feedback rating (viz., positive, negative, or neutral) for a submitted data. However, existing works [16,17] either utilize the proportion of only positive feedback to measure data quality, or lack comprehensive consideration about the data structure, leading to unsatisfactory quantitative results and unfairness. So, we present a new reputation evaluating method by modifying existing rating feedback mechanisms. To address the trusted reputation update, we propose a solution for vehicles to update their reputation trustworthily by generating corresponding proofs. Our solution needs not frequent accesses to the blockchain.

To summarize, the main contributions of this paper are as threefolds.

- We propose a privacy-preserving decentralized reputation management system for VCS (PPDR-VCS). Specifically, as for the privacy-preserving mechanism, we construct a traceable ring signature by leveraging non-interactive zero-knowledge proof [18] and the Schnorr signature scheme [19] to fulfill anonymous authentication in a VCS scenario. The proposed system guarantees the vehicles' privacy and also resists Sybil attacks existing in decentralized systems.
- We propose two updating factors to evaluate the reputation impartially. Thereinto, the truthfulness-based factor is quantified by leveraging the rating feedback mechanism where both the total number and ratio of feedback is taken into account for the fairness. The time-based factor ensures the validity of the data to meet the time-sensitive VCS scenario. Moreover, we design smart contracts to automatically update the reputation value once a task finishes, and generate zero-knowledge proofs to confirm the trusted reputation update.
- We make theoretical security and privacy analysis of the proposed system and verify the feasibility of the proof-of-concept prototype by implementing it on Parity Ethereum, and provide a comprehensive evaluation of the performance.

The remainder of this paper is organized as follows. In Sect. 2, we present the system model, threat model, and design goals. Some preliminaries are in Sect. 3. In Sect. 4, we propose PPDR-VCS. Subsequently, privacy and security analysis and performance evaluation are presented in Sect. 5 and 6. Then, we review some related works in Sect. 7. Finally, Sect. 8 draws the conclusion.

2 Problem Statement

In this section, we formalize the system model of PPDR-VCS, threat model and the underlying assumptions, and also identify our design goals.

2.1 System Model

Our PPDR-VCS system is a reputation management system auxiliary to the VCS. The system model mainly consists of four entities: Blockchain network, fog servers, vehicles and certificate authority as shown in Fig. 2.

Blockchain network refers to the consortium blockchain of fog servers in different traffic areas. It has a permissioned ledger, which is shared with the legitimate fog servers, and serves for the reputation management system. The reputation update is executed in a verifiable manner according to smart contracts without a central third party.

Fog servers act as consensus nodes (i.e. validators) in the consortium blockchain. Fog servers can be RSUs, base stations, or any other edge devices equipped with powerful computation and storage capabilities, and they verify and seal new blocks for maintaining the blockchain network. Moreover, fog servers are also in charge of identity and message authentication, data quality calculation and sensory data aggregation. The local fog server maintains a

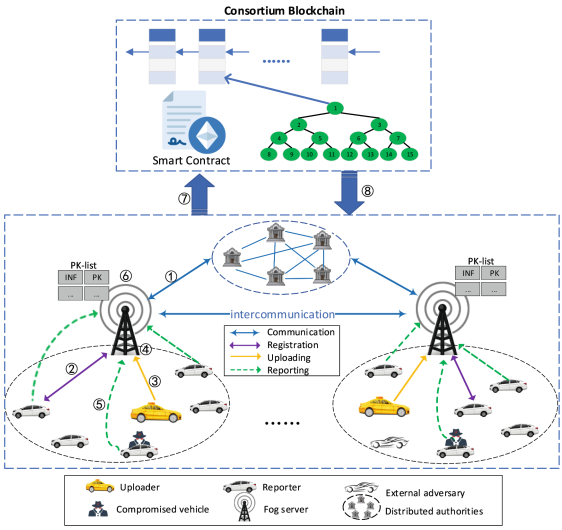

Fig. 2. System model

public key list (called PK-list), including public keys of the registered vehicles in the vicinity of it at a certain time. We assume that fog servers can only be accessed by vehicles from nearby location (e.g., by means of radio networks). Besides, transaction fees and mining rewards involved in the blockchain network are provided as the incentives for supporting fog servers.

Vehicles are divided into two categories: *Uploader* and *Reporter*. An *Uploader* always generates a data report with regard to some traffic information in a VCS task. Meanwhile, *Reporters* will give their feedback on this report to the fog server. In our system, we assume that there are plenty of *Reporters* submitting feedback on a data report. Once a data report is received by the fog server, it will reject another data report with the same traffic information. In this paper, we focus on the privacy protection and reputation update of the *Uploader* not the *Reporters*. Before diving into the details of our system, we firstly give a formal definition of the reputation of vehicles.

Definition 1 (Reputation of Vehicles): The reputation of a vehicle V_i, denoted as R_i, is the synthesized evaluation of the quality of vehicles to complete crowdsensing tasks. The reputation value ranges from 0 to 1. The bigger the reputation value, the better the past behavior of the vehicle, and the more likely it is able to complete the crowdsensing task with high quality. A vehicle can obtain an initial reputation R_i^0 before its first VCS task, and it can apply for the initial reputation only once. Without loss of generality, R_i^0 is set as 0.3 in our system.

Certificate Authority (CA) is responsible for generating system parameters and cryptographic keys for vehicles and fog servers. The CA receives registration requests from vehicles and fog servers. It also assists the initial reputation distribution and the anonymity revoke of misbehaved vehicles. The CA

maintains a list called IR-list, which records the public keys of vehicles who have applied for the initial reputation before. This is to prevent malicious vehicles from applying for initial reputation multiple times when their reputation is lower than the initial value. It does not conflict with the decentralized feature of our system because it stays offline when vehicles are performing crowdsensing tasks and the reputation are updated.

In our model, we assume task initiators, who are interested in some traffic information, have released crowdsensing tasks via the fog servers or the blockchain network. Winning vehicles are selected to upload the data report by some worker selection algorithms in VCS, and they will get rewards that are subject to their reputation values after the task. The details of the above procedure are not our concern. In this paper, we only focus on the decentralized reputation management system with privacy preservation. The workflow is as follows: ① Vehicles and fog servers register with the CA. They get their public keys, and legitimate vehicles get their initial reputation. ② Selected vehicles (*Uploader*s) submit encrypted sensory data along with a ring signature. ③ Local fog server decrypts and verifies the data, and then broadcasts the traffic message to nearby vehicles. ④ *Reporter*s in the proximity submit their feedback reports about this message. ⑤ Local fog server calculates the updating factors. ⑥ Local fog server runs the deployed smart contracts to update the reputation of the vehicle. Consortium fog servers verify all transactions about the reputation updating and build a new block periodically. ⑦ Vehicles update their reputation by accessing the blockchain.

2.2 Threat Model and Assumptions

We assume that the CA is fully trusted, and no adversary can breach it. Fog servers are honest-but-curious, i.e. they follow the protocol, but are also curious about others' privacy by launching passive attacks. Dishonest vehicles may be selfish or malicious, who may forge their reputation values and try to get private information of other vehicles. A dishonest *Uploader* is a legitimate vehicle that is selected to submit data reports, and a dishonest *Reporter* is also a legitimate vehicle that generates correct feedback reports intermittently, with the purpose of maximizing their reputation value via some illegal ways.

Malicious attackers are either compromised vehicles or external adversaries that may eavesdrop to violate vehicles' privacy or intentionally act to cripple the reputation system. A compromised *Reporter* may destroy the system by submitting incorrect feedback of a submitted data generated by an honest *Uploader*, or in collusion with a compromised *Uploader*. In our system, we assume that the number of honest *Reporters* is always higher than the number of compromised *Reporters*, which is also in accord with reality. As for the external adversary, it may forge the legitimate vehicles to perform VCS tasks and identify a vehicle's track from a set of submitted data by observing over time.

2.3 Design Goals

We summarize the design goals of PPDR-VCS under the threat model and assumptions.

- Privacy. ① Data privacy. A submitted data containing in-time traffic information should be hidden from other vehicles, and only revealed to the local fog server, who can verify the data, broadcast the traffic information and wait for the feedback from the *Reporter*s in the vicinity. ② Identity privacy. The *Uploader*'s identity can be protected, i.e. when an *Uploader* performs a VCS task, anyone including the fog server cannot identify its real identity. Furthermore, attackers cannot trace the *Uploader*'s driving trajectory via different reports. However, the anonymity of the *Uploader* can be revoked in case of any misbehavior.
- Security. The reputation system must resist two attacks, i.e., Sybil attacks and collusion attacks. No one can impersonate a legitimate vehicle to submit a data report even when some attackers want to generate a large number of fake vehicles to manipulate the reputation system. Any vehicle can not collude with a fog server to boost its reputation, which can pass the verification of the proof.
- Fairness. The reputation is updated depending on the data quality. The reputation value of *Uploader*s who submit high-quality data increases, and vice versa. Dishonest vehicles can not boost their reputation illegally. Moreover, anyone can not successfully re-upload a sensory data collected by other *Uploader*. Reputation update should be transparent and publicly verifiable to legitimate members of the consortium blockchain.
- Decentralization. Any central point of failure and any single point of control should be avoided. Therefore, the PPDR-VCS can still work well even if some fog server is compromised.

3 Preliminaries

3.1 zk-SNARK

The zero-knowledge succinct non-interactive arguments of knowledge (*zk-SNARK*) [20] is a novel form of zero-knowledge proof. It allows a prover convince a verifier that he knows some secret information without leaking any useful knowledge and the proof can be verified in a few milliseconds. A *zk-SNARK* algorithm $ZS = (Setup, Prove, Verify, Sim)$ is composed by the following probabilistic polynomial time (PPT) algorithms:

- $Setup(R_{el})$: input the constraint relations R_{el}, output the common reference string crs and the trapdoor τ. R_{el} is the non-interactive linear proof output constructed by the quadratic arithmetic program generated by the constraint condition.
- $Prove(crs, x, \omega)$: input a common reference string crs, a statement x and an evidence ω, return an argument π.

- $Verify(crs, x, \pi)$: input a common reference string crs, a statement x and an argument π, return a bit b.
- $Sim(R_{el}, \tau, x)$: input the constraint relations R_{el}, a simulation trapdoor τ and statement x, return an argument π.

3.2 Schnorr Signature Scheme with Re-randomizable Keys

Signatures with re-randomizable keys is a digital signature scheme [19], where the public and private keys can be re-randomized separately. It requires that the distribution of the re-randomized keys is the same as the original keys. So, the signer can sign a message m with a brand new key and prove the relationship between the original key pair (pk, sk) and the new key pair (pk', sk') in zero-knowledge, which guarantees the unlinkability of the signature. In this paper, we use the Schnorr signature scheme with re-randomizable keys $SSS = (KGen, Sig, Ver, RandSK, RandVK)$ to design our traceable ring signature. It mainly consists of the following algorithms:

- $KGen(1^{\lambda})$: given the security parameter λ, select a private key $sk \leftarrow Z_q$, compute a public key $pk = g^{sk}$ and output the key pair (pk, sk).
- $Sig(sk, m)$: given the private key sk and the message to be signed m, select $\alpha \leftarrow Z_q$ randomly, calculate $R = g^{\alpha}$, $c = H(m||R)$, $y = \alpha + sk \cdot c (mod\ q)$, and output the signature $\sigma = (c, y)$.
- $Ver(pk, m, \sigma)$: given the public key pk, message m and the signature σ, parse σ as (c, y), and check whether the equation $c = H(pk^{-c}, g^{y}, m)$ holds and output a bit b.
- $RandSK(sk, \rho)$: given the private key sk and the re-randomizable number ρ which is chosen by signer randomly, calculate the re-randomizable private key $sk' = sk + \rho (mod\ q)$ and output sk'.
- $RandVK(pk, \rho)$: given the public key pk and the re-randomizable number ρ, calculate the re-randomizable public key $pk' = pk \cdot g^{\rho} (mod\ q)$ and output pk'.

3.3 Ring Signature

Ring signatures enable a signer to include his/herself in an ad-hoc group (called a ring) and sign a message as a user in the ring without disclosing which one of them is the signer [21]. Ring signatures are often used to implement anonymous authentication, especially suitable for the ad-hoc network, such as Internet of vehicles [14]. In this paper, we construct a traceable ring signature $ZKTRS = (RSetup, RKGen, RSig, RVerify)$ shown in Fig. 3 for the sake of identity privacy protection of vehicles. The proposed ring signature consists of the output of SSS and a zero-knowledge arguement [18] of the randomization factor of the new public key with respect to original public key in a ring. Furthermore, the traceable ring signature includes an extra tracing algorithm.

$RSetup(1^{\lambda}, R_{el})$	$RSig(r_i, M, S, \Phi)$	$RVerify(S, \Sigma_i, M)$
$(G,q,g,h) \leftarrow \mathcal{G}(1^{\lambda})$ $H: \{0,1\}^* \rightarrow Z_q^*$ $(crs, \tau) \leftarrow Setup(R_{el})$ $Return\ ((G,q,g,h), H, crs, \tau)$	$Parse\ S = (P_1, \ldots, P_n)$ $if\ \nexists i : S_{[i]} = P_i\ return\ \perp$ $\rho \leftarrow \{0,1\}^{\lambda}$ $P_i^{'} \leftarrow RandVK(P_i, \rho), r_i^{'} \leftarrow RandSK(r_i, \rho)$ $r_{tag} \leftarrow Z_q, l = H(S \Vert M \Vert r_{tag})$ $\Delta \leftarrow (ct_1 = h^l, ct_2 = P_i \cdot \Phi^l)$ $x = (S, P_i^{'}, \Phi, ct_2), \omega = ((\rho, i), r_i, l)$ $\pi \leftarrow Prove(crs, x, \omega)$ $\sigma_i \leftarrow Sig(r^{'}, M \Vert S \Vert \Delta)$ $\Sigma_i = (\pi, P_i^{'}, \sigma_i, \Delta)$ $Return\ \Sigma_i$	$Parse\ S = (P_1, \ldots, P_n)$ $Parse\ \Sigma_I = (\pi, P_I^{'}, \sigma_I, \Delta)$ $b_1 \leftarrow Verify(crs, x, \pi)$ $b_2 \leftarrow Ver(P_I^{'}, M \Vert S \Vert \Delta, \sigma_I)$ $Return\ (b_1, b_2)$
$RKGen(1^{\lambda})$		
$\phi \leftarrow Z_q, \Phi \leftarrow h^{\phi}$ $(P_i, r_i) \leftarrow KGen(1^{\lambda})$ $Return\ ((\phi, \Phi), (P_i, r_i))$		

Fig. 3. The proposed traceable ring signature scheme $ZKTRS$

3.4 Rating Feedback Mechanism

The rating feedback mechanism [15] is a trust model which can represent the degree of the trust on the received data. This trust model was originally designed to solve the trust problem of certificates between users. Recently, the rating feedback mechanism is used in many real crowdsensing applications such as Waze, eBay, etc., to provide a rating by feedback (positive, negative and uncertainty) which are received from the consumers of these services. The benefits of using a feedback rating paradigm in VCS are that it is fast, less expensive, and exudes the essence of a vehicular crowdsensing paradigm. In our system, due to the lack of knowledge of an event, we can not assert the uploaded data is true or false. Thus, the rating feedback mechanism is used for measuring the truthfulness of uploaded data. Unlike some existing rating feedback mechanisms, we comprehensively consider both the amount of feedback and the proportion of positive and uncertainty feedback.

4 The Proposed System PPDR-VCS

In this section, we firstly present the privacy-preserving mechanism of PPDR-VCS, and then describe the detailed PPDR-VCS.

4.1 The Privacy-Preserving Mechanism of PPDR-VCS

When a vehicle performs a crowdsensing task, the identity privacy and data privacy are protected in our system. We propose $ZKTRS = (RSetup, RKGen, RSig, RVerify)$ shown in Fig. 3 to protect the identity privacy. We will detail four algorithms of $ZKTRS$ in the VCS scenario. The data privacy is accomplished by encryption algorithms, such as ElGamal.

System Setup *RSetup*

The CA sets the security parameter λ and generates the public parameters of the system. Let G be a group of a prime order $q > 2^{\lambda}$, g and h are two generators of group G. $H : \{0,1\}^* \rightarrow Z_q^*$ is a collision-resistant hash function. The CA also runs the $Setup(R_{el})$ in *zk-SNARK* algorithm to generate the common reference string and the trapdoor (crs, τ). Finally, the system public parameters are $para = \{G, q, g, h, H, crs\}$.

Key Generation *RKGen*

The CA publishes the public key $\Phi = h^{\phi}$, where $\phi \in Z_q$ is its private key. The fog server F selects $sk_f \leftarrow Z_q$ as its private key, and computes the public key $pk_f = g^{sk_f}$. Vehicles generate their public-private key pair $(pk_i = g^{sk_i}, sk_i)$, where $sk_i \in Z_q$. All legitimate entities can get their public key certificates.

Signature Generation *RSig*

Assume a vehicle V_i enters the vicinity of a fog server F, it broadcasts the public key pk_i and the corresponding certificate. The fog server will add the legitimate public key into the PK-list. When the V_i performs a crowdsensing task and uploads a sensory data M, it first selects a ring set, i.e. n ring members, from the PK-list published by the local fog server.

Assume the V_i gets a ring set $S = \{pk_1, pk_2, ..., pk_n\}$ according to the (n, k)-privacy ring selection algorithm [22]. Then, it encrypts the data M as $Enc_{pk_f}(M)$, generates a traceable ring signature $\Sigma_i = (\pi, pk_i', \sigma_i, \Delta)$ on M and sends $(Enc_{pk_f}(M), \Sigma_i)$ to the F as following.

Firstly, the V_i selects a random number $\rho \leftarrow \{0,1\}^{\lambda}$, runs the re-randomizable algorithms $RandVK(pk_i, \rho)$ and $RandSK(sk_i, \rho)$ to generate the new key pair (pk_i', sk_i') where

$$pk_i' = pk_i \cdot g^{\rho}$$
$$sk_i' = sk_i + \rho$$

Secondly, the V_i selects a random number $r_{tag} \in Z_q$, calculates $l = H(S||M||r_{tag})$, generates the tracing tag $\Delta = (ct_1, ct_2)$ where $ct_1 = h^l$ and $ct_2 = pk_i \Phi^l$ and sets the statement $x = (S, pk_i', \Phi, ct_2)$, and the witness $\omega = ((\rho, i), sk_i, l)$. Then the V_i constructs a non-interactive zero-knowledge proof by leveraging the *zk-SNARK* algorithm:

$$\pi \leftarrow Prove\left(\begin{pmatrix} pk_i g^{\rho} = pk_i' \\ ct_2 = pk_i \Phi^l \end{pmatrix} : crs, x, \omega\right)$$

Thirdly, the V_i generates a signature via $Sig(sk_i', M||S||\Delta)$ in SSS, i.e. $\sigma_i \leftarrow (c, y)$, where $c = H(M||S||g^{\alpha}||\Delta)$, $y = \alpha + sk_i' \cdot c(mod q)$, and $\alpha \in Z_q$.

Finally, the V_i generates a ring signature $\Sigma_i = (\pi, pk_i', \sigma_i, \Delta)$.

Signature Verification *RVerify*

Upon receiving $(Enc_{pk_f}(M), \Sigma_i)$, the F firstly decrypts $Enc_{pk_f}(M)$ to get the data M, and then parses S as $\{pk_1, pk_2, ..., pk_n\}$ and Σ_i as the tuple $(\pi, pk_i', \sigma_i, \Delta)$. Then the $Verify$ algorithm in *zk-SNARK* algorithm takes as input the tuple (crs, x, π) and checks whether the proof π holds. The notation b_1 is defined as a result of the above procedure which can be described as:

$$b_1 \leftarrow Verify(crs, x, \pi)$$

Later, the equation $H(M||S||g^y pk_i'^{-c}||\Delta) \stackrel{?}{=} c$ is calculated to verify the validity of the signature. It outputs $b_2 = 1$ when the equation holds, otherwise output $b_2 = 0$.

If all the above verifications are valid, i.e. $b_1 = b_2 = 1$, the M is accepted by the F, who will then broadcast the M to the nearby vehicles. Otherwise, the M will be rejected.

4.2 The Detailed PPDR-VCS

Our proposed PPDR-VCS consists of four phases: system initialization, entities registration, initial reputation distribution, and reputation update.

Algorithm 1. PPDR-VCS

Input: Security parameter

Output: System parameters, public key list, reputation transaction, new block, new reputation and the proof.

/* **System initialization** */

The CA generates the system parameters;

/* **Entities registration** */

Vehicles and fog servers register with the CA and get their certificated public-private key pairs;

/***Initial reputation distribution** */

A vehicle authenticates itself to the local fog server F;

The F generates a reputation transaction;

The vehicle accesses an initial reputation from the blockchain, and generates a proof on the reputation.

/***Reputation update** */

An *Uploader* submits a sensory data M to the F which verifies and broadcasts the M, and receives feedback about the M from the *Reporters* in the vicinity of the F.

The reputation of the *Uploader* is re-calculated according to the data quality, which is quantified by two factors: the truthfulness of the data and the response time.

The F generates a reputation updating transaction;

A new block is created;

The *Uploader*'s reputation is updated and the *Uploader* generates the proof of its current reputation.

System Initialization
The CA initializes the whole reputation system via the algorithm $RSetup$. After that, the system parameters $para$ are public to all entities. The CA also initializes an empty set using bloomfilter $\Omega = \{\varnothing\}$ to maintain the IR-list for the query efficiency.

Entities Registration
All vehicles firstly provide the unique identification (e.g. vehicle identification number (VIN) and driver's ID number) to the CA for the registration. The CA will verify the correctness of the identification and check whether the vehicle has registered before. Each vehicle is only allowed to be registered once. Fog servers also need to register with the CA. All entities get their public-private key pair (pk, sk) via the algorithm $RKGen$.

Initial Reputation Distribution
A registered vehicle can get an initial reputation anonymously on its first task within an area of a fog server F. As shown in Fig. 4, the F will send an identifier query to the CA for the initial reputation identifiers in advance. The CA generates an identifier N_i by encrypting a random number a_i and signing it: $N_i \leftarrow Sig_{\phi}(Enc_{\Phi}(a_i))$. Here, we can take $ECDSA$ as the signature algorithm $Sig(.)$, and ElGamal encryption scheme as the algorithm $Enc(.)$.

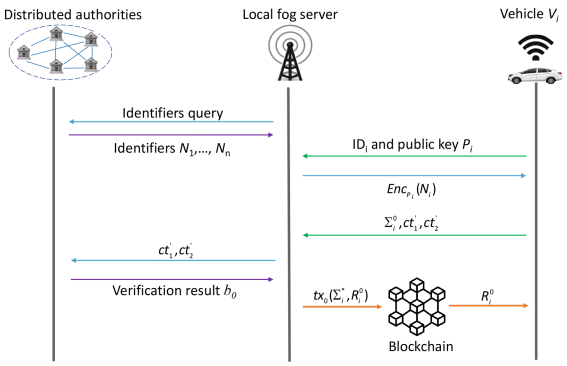

Fig. 4. The distribution of the initial reputation

The F randomly selects and encrypts an identifier N_i by using the V_i's public key pk_i, and sends $Enc_{pk_i}(N_i)$ to the V_i. The V_i decrypts it and gets N_i. Then, the V_i generates a one-time public and private key pair (pk_i'', sk_i''), encrypts N_i and sk_i'' by using pk_i'' and Φ, respectively.

$$\eta_1 \leftarrow Enc_{pk_i''}(N_i), \eta_2 \leftarrow Enc_{\Phi}(sk_i'').$$

The V_i also generates a ring signature Σ_i^0 on η_1 and η_2 via the algorithm $RSig$ and sends it to the F along with η_1 and η_2. The F parses and verifies Σ_i^0, and sends $\eta_1, \eta_2, \Delta = (ct_1, ct_2)$ to the CA, who will firstly check whether the vehicle applies for the initinal reputation before by computing $pk_i = \frac{ct_2}{ct_1{}^{\phi}}$, and searching for the public key pk_i in the IR-List. If pk_i is in the Ω, the CA will inform the F to reject the initial reputation request from the vehicle. Otherwise, the CA gets the one-time private key sk_i'' by decrypting η_2, and then it will get N_i by decrypting η_1 via sk_i''. After that, the CA verifies the signature and decrypts N_i to obtain a_i. If the verification succeeds and a_i is correct, the CA will send a bit $b_0 = 1$ to the F to inform that this vehicle is legitimate and applys for the initial reputation for the first time. Otherwise, $b_0 = 0$ will be sent.

If the F receives $b_0 = 1$, it will generate a reputation transaction tx_0 about (Σ_i^*, R_i^0), where $\Sigma_i^* = H(\Sigma_i^0)$. The tx_0 is then verified, sealed in a block, and appended to the consortium blockchain. Otherwise, the V_i is rejected. Finally, the V_i generates a proof π_0 on (Σ_i^*, R_i^0) by leveraging the *zk-SNARK* algorithm:

$$\pi_0 \leftarrow Prove\left(\begin{pmatrix} \alpha + sk_i^{'} \cdot c(modq) = y \\ \Sigma_i^* = H(\Sigma_i^0) \end{pmatrix} : crs,\, x_0,\, \omega_0\right) \tag{1}$$

where $x_0 = (S, (c, y), \Sigma_i^0, \Sigma_i^*)$ is the statement, and $\omega_0 = (sk_i', \alpha)$ is the witness. π_0 indicates that R_i^0 is indeed the reputation of a legitimate vehicle of which the ring sinature is Σ_i^0, and π_0 also confirms the relationship between R_i^0 and pk_i anonymously.

When the V_i performs a crowdsensing task for the first time, it submits a sensary data M together with its initial reputation and the proof (R_i^0, π_0). Anyone can verify the reputation proof π_0 as follows:

$$b_3 \leftarrow Verify(crs, x_0, \pi_0)$$

It outputs $b_3 = 1$ when the equation holds, otherwise output $b_3 = 0$.

Note that a vehicle's reputation changes all the time since it is constantly engaged in crowdsensing tasks. A vehicle needs to show its current reputation and the proof each time. Only when the reputation verification has passed, the fog server will then broadcast the traffic data M and the reputation of the $Uploader$ will be updated according to the quantified data quality.

Reputation Update

Assume the F broadcasts the M submitted by an authenticated $Uploader$, and $Reporters$ in the vicinity of the F can provide positive, negative or neutral ratings on the M, which is quantified by the truthfulness-based factor Q_r and the time-based factor Q_t (The details of Q_r and Q_t are shown in the next section). Smart contracts are created and deployed on the ledger. Once the updating factors have been calculated, it triggers the *Reputation update algorithm* to execute automatically. In Algorithm 2, the parameter γ controls the degree of importance of Q_r and Q_t. The value of γ is specified according to the requirements of the task initiator. When Q_d is positive, the data has a tendency to be

true and uploaded within survival time, the vehicle's reputation will increase. Otherwise, if Q_d is negative which means $0 < (1 + Q_d) < 1$, the vehicle's reputation will be decreased. And then, the V_i's updated reputation is calculated based on Q_d and its current reputation R_i. Since the range of R_i^* is [0,1], we use the arctan function *atan* for the normalization as $R_i^* = 2atan(R_i^{*'})/\pi$. Besides, if a normalized reputation of some vehicle is less than a threshold (we take 0.05 as an example), which means the vehicle frequently submit low-quality data or engage in improper behavior, the vehicle should be not allowed to participate in the task or be punished.

Algorithm 2. Reputation Update Algorithm

Input: The V_i's ring signature Σ_i, current reputation R_i, the reputation updating factors Q_r and Q_t

Output: The updated reputation R_i^*

if $Q_r > 0$ **then**
    $Q_d = \gamma Q_r + (1 - \gamma)Q_t$;
else
    $Q_d = Q_r$;
end if
$R_i^{*\prime} = (1 + Q_d)R_i$;
$R_i^* = 2atan(R_i^{*'})/\pi$;
if $R_i^* \geq 0.05$ **then**
    $\Sigma_i^* = H(\Sigma_i)$
    return (Σ_i^*, R_i^*);
else
    return (Σ_i, R_i^*);
end if

After executing the reputation update algorithm, the F generates a reputation updating transaction Tx, which includes the transaction number, the V_i's ring signature and the corresponding reputation value (Σ_i^*, R_i^*) or (Σ_i, R_i^*), and the F's public key and signature, as shown in Fig. 5. The transaction Tx is then verified and sealed in a block by the fog validator. Assuming a block has the capacity of v transactions, the fog validator with the highest priority generates a new block, which consists of a blockheader, the validator's updated reputation (vital assets to be authorized as validator), v transactions and a signature from the validator. And the blockheader includes a block number, a hash of previous blockheader, a hash root of Merkle tree constructed from v transactions and a timestamp, etc. Then the validator appends this block to the consortium blockchain and informs the network.

Case I: the V_i accesses the blockchain and gets (Σ_i^*, R_i^*). The V_i computes $H(\Sigma_i)$ and gets his new reputation R_i^* by comparing with Σ_i^*. Then, the V_i updates his reputation by himself and generates a proof by leveraging the *zk-SNARK* algorithm:

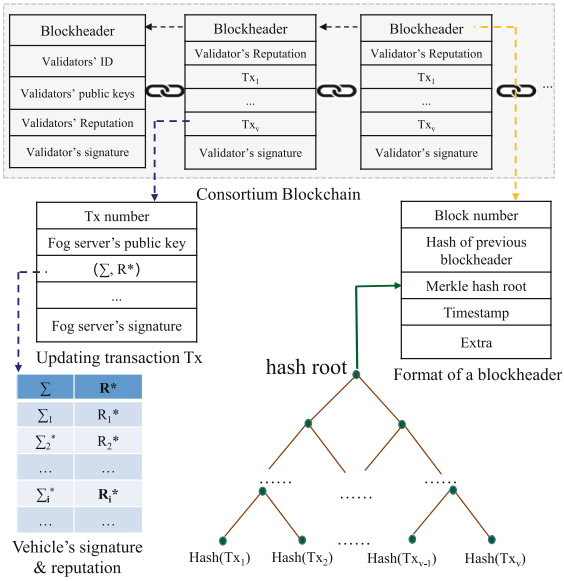

Fig. 5. Construction of the consortium blockchain

$$\pi_1 \leftarrow Prove\left(\begin{pmatrix} \alpha + sk_i^{'} \cdot c(modq) = y \\ \Sigma_i^* = H(\Sigma_i) \end{pmatrix} : crs,\, x_1,\, \omega_1\right) \tag{2}$$

where $x_1 = (S, (c, y), \Sigma_i, \Sigma_i^*)$ and $\omega_1 = (sk_i', \alpha)$. This proof confirms the relationship between R_i^* and pk_i anonymously.

Case II: If a vehicle's R_i^* is lower than the threshold on the blockchain, the CA can unite the fog server F to revoke the anonymity of the V_i by calculating $pk_i = \frac{ct_2}{ct_1{}^{\phi}}$, and mapping the public key pk_i to the V_i. The vehicle will be punished, and the CA will send pk_i to the F, who can remove pk_i from the PK-list. That means the vehicle is not allowed to perform the crowdsensing task for a time.

4.3 Updating Factors Calculation

The truthfulness of the data is calculated based on the feedback from *Reporters* in the proximity. To ensure fairness, the truthfulness-based factor Q_r is calculated by the total number of the feedback, the belief and uncertainty masses. Meanwhile, each task has a lifetime and the submitted data beyond its lifetime will be discarded. So the time-based factor Q_t is modeled as an exponential decay function. The V_i's reputation updating factors are computed as shown in Algorithm 3.

Algorithm 3. Updating Factors Calculation

Input: Sensory data M from the *Uploader* V_i, the V_i's current reputation R_i, the number of three kinds of feedback from repoters N_b, N_f and N_u

Output: V_i's reputation updating factors Q_r and Q_t

```
Calculate posterior probability (b, f, u) and the weights of belief and uncertainty ω_b, ω_u;
τ_r = ω_b · b + ω_u · u;
if τ_r ≥ 0.5 then
    Q_r = (τ_r)^θ;
    if t − t_0 < T then
        Q_t = 1 / e^{ρ(t−t_0)};
    else
        drop;
    end if
else
    Q_r = −(0.5 − τ_r)^μ;
end if
```

① Calculation of Q_r: Let $N = N_b + N_f + N_u$ represents the total number of the feedback. According to the Bayesian theorem based on the feedback, the posterior probabilities of belief, disbelief and uncertainty are given as: $b = \frac{N_b+1}{N+3}$, $f = \frac{N_f+1}{N+3}$, $u = \frac{N_u+1}{N+3}$. We consider that the expected truthfulness of a submitted data depends on the belief and uncertainty masses which is modeled as nonlinear weighted regression model. Then the expected truthfulness can be represented as:

$$\tau_r = \omega_b \cdot b + \omega_u \cdot u \tag{3}$$

where $0 < \omega_b < 1$ is the weight of belief feedback and $0 < \omega_u < 1$ is the weight of uncertainty feedback.

We apply Richard's generalized curve [23] to model ω_b while considering the total number of the feedback. The proportion of belief feedback is also taken into consideration. Less N should have lower ω_b, in other words, ω_b should gradually increase with N, represented as follows:

$$\omega_b = \frac{1}{(1 + \delta e^{-v_b N})^{\frac{1}{\delta}}} \cdot \frac{N_b}{N} \tag{4}$$

where $\delta(\delta > 0$ and $\delta \neq \infty)$ controls the initial value of the weight and the point that ω_b turns to exponential growth. If the F receives a mass of feedback, the value of δ should be set higher, or vice-versa. v_b, the rate of growth, controls the speed of ω_b to reach the maximum value.

Intuitively, most of the *Reporters* are unaware of the event when it just happens. Thus ω_u will increase when the number of feedback is small. However, it will decrease as more feedback is received. Hence, we set N_{thres} denotes the threshold value of the feedback and $\omega_u^{max} = 0.5$ as the maximum value of the weight which means that the belief feedback would make more contributions to the expected truthfulness of the report. After that, ω_u is modeled as a piecewise

function that has a growthing part and a decaying part. The growth part is similar to ω_b while the decayed part is modeled by the Kohlrausch relaxation function [24]. The equation of ω_u is represented as:

$$\omega_u = \begin{cases} \frac{1}{2(1+\delta e^{-v_u N})^{\frac{1}{\delta}}} \cdot \frac{N_u}{N} & N < N_{thres} \\ e^{-(N-N_{thres})^{\epsilon}} & N \geq N_{thres} \end{cases} \tag{5}$$

where ϵ is the Kohlrausch factor, controls the decreasing speed of ω_u after N reaches N_{thres}. A higher value of ϵ can eliminate the effect of uncertainty immediately. The threshold value N_{thres} controls the attenuation point of ω_u where the effect of uncertainty feedback starts to reduce. The concept of other parameters are similar to ω_b.

After achieving the expected truthfulness τ_r, a link function which is treated as the truthfulness-based updating factor Q_r based on Cumulative Prospect theory [25], is described as follows:

$$Q_r = \begin{cases} (\tau_r)^{\theta} & \tau_r \geq 0.5 \\ -(0.5-\tau_r)^{\mu} & \tau_r < 0.5 \end{cases} \tag{6}$$

where Q_r has the value in the interval $[-1, 1]$. θ and μ control the rate of the change of upper and lower parts of Q_r, respectively. $\tau_r = 0.5$ is a reference point. $\tau_r > 0.5$ means that the data tends to be true and the vehicle's reputation should be improved, or vice versa.

② Calculation of Q_t: Another factor that influences the *Uploader*'s reputation is the task response time. When a task is completed within task survival time T (i.e. $t - t_0 < T$), the F will accept the data report and calculate the time-based updating factor Q_t. The contribution of the data M decreases with the increases of the response time, and the time-based updating factor Q_t should be quantified according to the contribution. Therefore Q_t is modeled as an exponential decay function:

$$Q_t = \frac{1}{e^{\rho(t-t_0)}} \quad t - t_0 < T \tag{7}$$

where t is the data uploading time, t_0 is the task release time and ρ is the time factor which controls the decay rate of Q_t. A higher ρ can be chosen if the task is an emergency or time-sensitive. The shorter the response time, the bigger the time-based factor, because a "fresh" report can make more contributions.

5 Privacy and Security Analysis

5.1 Privacy

Data privacy is protected by the encryption algorithm. In our PPDR-VCS, vehicles encrypt their submitted data via ElGamal encryption algorithm. Only the

fog server can decrypt the data before being broadcast. That is, an adversary can get the data submitted by other uploaders only if he can solve the discrete logarithm problem on G. So, the data privacy is preserved in our system.

When a vehicle uploads a message, it will be authenticated by our proposed $ZKTRS$ which has properties of the unforgeability, anonymity and traceability.

Theorem 1. Let *zk-SNARK* be a computationally sound argument of knowledge, SSS be a signature scheme with re-randomizable keys which is unforgeable in the random oracle model. Our $ZKTRS$ is an unforgeable traceable ring signature scheme in the random oracle model.

Proof. Let H be a random oracle hash function. Assume that there exists a PPT adversary $\mathcal{A}$ who can forge a valid traceable ring signature successfully with the probability of $\epsilon(\lambda)$ that is non-negligible. Then the following reduction $\mathcal{R}$ can be constructed to break the unforgeability of the signature scheme with re-randomizable keys. If $\mathcal{A}$ can forge a valid signature successfully, then there exists a forgery in the SSS.

First, the unforgeability of the tracing tag $\Delta = (ct_1, ct_2)$ is guaranteed by the ElGamal [26] of which DDH assumption is hard. The sound of *zk-SNARK* ensures the ciphertext ct_2 as the form $ct_2 = g^{sk_i}\Phi^l$. Next, we construct the reduction $\mathcal{R}$ which breaks the unforgeability of the SSS and $Adv_{vk}(\mathcal{R}^{\mathcal{A}}) \leq \frac{\epsilon(\lambda)}{q}$. The reduction $\mathcal{R}^{\mathcal{A}}(pk)$ is given as follows.

- Choose an index $i \leftarrow \{1, ..., q\}$ uniformly at random, and set $pk_i = P$.
- For all indices $k \neq i$, $\mathcal{R}$ sets $(pk_k, sk_k) \leftarrow KGen(1^\lambda)$, The adversary $\mathcal{A}$ is provided with the public keys $\mathbf{P} = (pk_1, ..., pk_q)$.
- For all indices $k \neq i$, $\mathcal{A}$ is allowed to make the corrupt query and sign query.
  - Corrupt query(k): A corrupt query is in the form of $k \in \{1, ..., q\}$. The challenger sends sk_k to $\mathcal{A}$ and appends pk_k to the corrupted user list $\mathcal{C}$.
  - Sign query (k, S, m): A sign query is in the form of (k, S, m), where m is the uploaded message, S is a set of public keys and k is an index such that $pk_k \in S$, The challenger responds with $RSig(sk_k, m, S, \Phi)$.
- $\mathcal{A}$ makes the form (i, S, m) of the sign query, and $\mathcal{R}$ responds as follows.
  - Select a random ρ.
  - Compute the re-randomized public key $pk_i' \leftarrow RandVK(S_i, \rho)$, where S_i is the i-th member of the S, the tracing tag $\Delta = (ct_1, ct_2)$, and the *zk-SNARK* proof:

$$\pi \leftarrow Prove\left(\begin{pmatrix} pk_i g^\rho = pk_i' \\ ct_2 = pk_i \Phi^l \end{pmatrix} : crs, x, \omega\right)$$

  - On input of $(m||S||\Delta, \rho)$, and query the signing oracle. Assume the challenger responses with σ.
  - Return $(\pi, pk_i', \sigma, \Delta)$.
- $\mathcal{A}$ outputs a forged signature (S^*, Σ^*, m^*).
- $\mathcal{R}$ parses the signature as $\Sigma^* = (\pi^*, pk^*, \sigma^*, \Delta^*)$ and gets $(S^*||pk^*||\Delta^*, \omega^*, \pi^*)$, where ω^* is the form of (ρ^*, i^*, sk^*, l^*) and $i^* = i$, $\mathcal{R}$ aborts if $i^* \neq i$.

- $\mathcal{R}$ returns the tuple of a signature $(m^*||S^*||\Delta^*, \sigma^*, \rho^*)$ and finishes the simulation.

Assume $\mathcal{A}$ successfully forges a valid signature $\Sigma^* = (\pi^*, pk^*, \sigma^*, \Delta^*)$ which satisfies the following conditions: $\mathcal{A}$ doesn't make any corrupt oracle on i, where $pk_i \in S^*$ and $S^* \subseteq \mathbf{P} \setminus \mathcal{C}$; $\mathcal{A}$ doesn't make any signing oracle in the form of $(., S^*, m^*)$; and the equation $RVerify(S^*, \Sigma^*, m^*) = 1$ holds. When $\mathcal{A}$ queries the signing oracle in the form of (i, S, m), the challenger responds $\sigma \leftarrow Sig(RandSK(sk_i, \rho), m||S||\Delta)$ to $\mathcal{R}$. Therefore, we know that $\mathcal{R}$ perfectly simulates the inputs of $\mathcal{A}$. From the above, in the case $i^* = i$, we have the re-randomizable key

$$pk^* = RandVK(pk_{i^*}, \rho^*) = RandVK(pk_i, \rho^*)$$

such that $RVerify(S^*, \Sigma^*, m^*) = 1$, which implies that

$$\begin{aligned} &Verify(crs, (S^*, pk^*, \Phi, ct_2), \pi^*) = 1; \\ &Ver(pk^*, m^*||S^*||\Delta^*, \sigma^*) = 1. \end{aligned}$$

It shows that $(m^*||S^*||\Delta^*, \sigma^*, \rho^*, \Delta^*)$ is the tuple of a valid signature. In other words, if $\mathcal{A}$ successfully forges a valid signature, $\mathcal{R}$ can forge a valid signature with the same probability. That is

$$\begin{aligned} Adv_{vk}(\mathcal{R}^{\mathcal{A}}) &= \sum_{j=1}^{p} Pr[i^* = j] \cdot Pr[\mathcal{R}^{\mathcal{A}} | i^* = j] \\ &+ Pr[\overline{i^* = j}] \cdot Pr[\mathcal{R}^{\mathcal{A}} | \overline{i^* = j}] \\ &\geq \sum_{j=1}^{p} Pr[i^* = j] \cdot Pr[\mathcal{R}^{\mathcal{A}} | i^* = j] \\ &\geq \frac{1}{q} \cdot Pr[\mathcal{R}^{\mathcal{A}} | i^* = j] \geq \frac{\epsilon(\lambda)}{q} \end{aligned}$$

which is non-negligible. This contradicts to the unforgeability of SSS.

Theorem 2. Let *zk-SNARK* be perfect zero-knowledge, SSS be a signature scheme with re-randomizable keys in the random oracle model, then $ZKTRS$ is anonymous in the random oracle model.

Proof. Consider the following games:

- $Game_0$: For all indices $i \in \{1, ..., q\}$, the reduction sets $(pk_i, sk_i) \leftarrow KGen(1^\lambda)$. The public keys $\mathbf{P} = (pk_1, ..., pk_q)$ are provided to the adversary $\mathcal{A}$. $\mathcal{A}$ can make queries in the form of (k, S, m), where m is the uploaded message, S is a set of public keys and k is an index such that $pk_k \in S$. The challenger responds with $RSig(sk_k, m, S, \Phi)$. $\mathcal{A}$ sends the tuple (i_0, i_1, S, m)

to request a challenge, where i_0, i_1 are indices such that $pk_{i_0}, pk_{i_1} \in S$. The challenger chooses a bit $b \leftarrow \{0,1\}$ randomly and sents $RSig(sk_{i_b}, m, S, \Phi)$ to $\mathcal{A}$. Also, random numbers $(\omega_1, ..., \omega_q)$ are given to $\mathcal{A}$. $\mathcal{A}$ outputs b' and makes a success if $b' = b$.

- $Game_1$: The proof π in the challenge step is computed as

$$\pi \leftarrow Sim(\tau, x)$$

where $x = (S^*, pk'_{i_b}, \Phi, ct_2)$. The other part of $Game_1$ is similar to $Game_0$.
- $Game_2$: Is similar to $Game_1$ except that the challenge signature is $(\pi, pk', Sig(sk', m^*||S^*||\Delta), \Delta)$, where $(pk', sk') \leftarrow KGen(1^\lambda)$.
- $Game_3$: Is defined as $Game_2$ except that the tracing tag Δ' is computed by r'_{tag} which is freshly chosen randomly.

We can find that adjacent games are indistinguishable.

$Game_0 \approx Game_1$: In these two games, all the parameters are generated honestly except the proof π. Because of the zero-knowledge property of *zk-SNARK*, the proof generated by $Sim(\tau, x)$ and the proof generated by $(x, (\rho, i))$ are statistically close.

$Game_1 \approx Game_2$: The two games are different only in the sampling step of the key pair (pk', sk') which are used to compute the challenging signature. In $Game_1$, the signature computed by (pk', sk'), while in $Game_2$ the pair is freshly sampled. Since the keys of signature scheme are perfectly re-randomizable, the two games are identical.

$Game_2 \approx Game_3$: The two games differ only in the choosing procedure of random numbers used to compute the tracing tag. In $Game_2$ the signature computed by r'_{tag}, while in $Game_3$ the random number is freshly sampled. Since the pseudo-random property of the tracing tag is derived from pseudo-random numbers, the tracing tags are computationally indistinguishable. Therefore the games are identical.

From the above, the challenge signature computed in $Game_3$ is irrelevant to b which means that $\mathcal{A}$ wins the $Game_3$ with the probability $\frac{1}{2}$. Besides, we have $Game_1 \approx Game_2$ and $Game_0 \approx Game_1$. Thus, we can conclude that any $\mathcal{A}$ cannot win the game with negligible probability greater than guessing.

To sum up, anonymity means the vehicle's identity is hidden in the ring. Anyone including the fog server cannot distinguish which secret key has been used to generate the ring signature. Furthermore, the ring selection algorithm in our system can achieve (n, k)-privacy, thus any attackers cannot trace a vehicle's identity by observing different ring signatures of the same vehicle. Then the identity anonymity and driving trajectory of the vehicles are protected.

Theorem 3. The proposed $ZKTRS$ achieves traceability if *zk-SNARK* is perfectly correct and zero-knowledge, as well as the variant ElGamal encryption is sound.

Proof. The correctness and soundness of *zk-SNARK* guarantee that the tracing tag is generated by the signer. We set (Φ, ct_2) as the statement and (sk_i, l) as the witness. The signer can calculate the proof using its private key. In other words, the proof of *zk-SNARK* guarantees that ct_2 is generated as the form $ct_2 = g^{sk_i}\Phi^l$. If we take the hash function H as a random oracle, we can prove that it is computationally infeasible to find another $l' = H(S'||m'||r'_{tag})$ as a collision, given the output l. Based on the correctness of variant ElGamal, we can ensure that the decryption is correct which means the signer's identity can be revoked. Only the CA can decrypt the ciphertext using the private key ϕ and find out the identity of the poorly behaved signer, whose reputation is lower than the threshold value. The privacy of good behaved vehicles, whose reputation is bigger than the threshold value, is protected unconditionally. Nobody even the CA can revoke the anonymity of them, because (Σ_i^*, R_i^*) are recorded on the blockchain rather than (Σ_i, R_i^*) and ct_2 can not be obtained via Σ_i^*.

In conclusion, the traceable ring signature can be generated by the *Uploader* in a ring S, no one can forge others' signature. Moreover, the CA can revoke the anonymity of a poorly behaved vehicle. Hence, conditional anonymity is preserved in our PPDR-VCS.

5.2 Security

In our PPDR-VCS, all vehicles need to register with the CA firstly. When a vehicle is performing a VCS task in a coverage area of a fog server, the public key of the registered vehicle will be maintained in the PK-list by the local fog server. The public key is certificated by the CA, which prevents an illegal vehicle from generating multiple public keys and launching Sybil attacks. When vehicles perform crowdsensing tasks, all transmitting messages are required to be authenticated to the fog server. Sybil devices cannot damage our reputation system through misbehaviors without successful registration and authentication.

An external adversary can launch Sybil attacks in the following two ways: The first one is to generate a large number of Sybil vehicles [12] to manipulate the reputation system. Since the fake vehicle has no legal filing information with DMV, it can not register with the CA successfully. Second, it tries to compromise a legitimate vehicle and forge a signature of its submitted message. Since the unforgeability of the signature is guaranteed by the proposed $ZKTRS$, the adversary can not forge a valid signature which can pass the verification process. In case an attacker creates a ring signature by using his secret key, generated by running $KGen(1^\lambda)$ algorithm, along with a ring set. Since its public key is not in the PK-list, the signature cannot pass the verification $RVerify$. So the proposed system can resist Sybil attacks.

In our system, there is no such reputation center, which is responsible for evaluating and updating the reputation. Although the fog server is responsible for the calculation of updating factors, it follows the protocol and computes Q_r and Q_t honestly based on the feedback responded by *Reporter*s. The reputation is updated by the smart contracts on the blockchain, which is tamper-resistant. So, there is no opportunity for collusion between the vehicle and the fog server.

5.3 Fairness

Firstly, the crowdsensing data is encrypted by the *Uploader*, which will prevent some lazy *Uploader* from re-uploading the data report to earn profit. Secondly, fairness is guaranteed by the reputation update algorithm, which is influenced by the data quality quantification. Since we quantify the data quality by using the truthfulness-based and time-based updating factors detailed as Q_r and Q_t in Algorithm 3. So our analysis will focus on how the calculation of Q_r and Q_t can ensure the fairness.

Q_r is calculated based on τ_r (i.e. $\tau_r = \omega_b \cdot b + \omega_u \cdot u$), where the weight of belief feedback ω_b and the weight of uncertainty feedback ω_u are modeled as Richard's generalized curve and kohlrausch relaxation function, respectively. Since the Richard's generalized curve can control the initial lower asymptote, inflection point, and the rate of change, so fog servers can set appropriate parameters to control ω_b for different tasks. This makes the calculation of τ_r objective and authentic. Besides, the proportion of the belief feedback is taken into consideration. It is obviously that the data supported by more feedback is more reliable. What's more, the amount of all received feedback is also added into our expression (Eq. 2) in order to compute ω_b fairly. Consider the following situation, there is an *Uploader* uploading a wrong data and manipulating several *Reporters* to give wrong feedback. If we leave out the amount of feedback, the proportion will be high when the amount of the received feedback is small. Then ω_b is calculated to be a bigger value than its normal one. And this malicious *Uploader*'s reputation will be increased illegally. Similarly, the growth part of ω_u is also modeled as Richard's generalized curve. Hence, the truthfulness-based updating factor Q_r can be calculated via ω_b and ω_u fairly.

As for Q_t, it is modeled as an exponential decay function on account of the contribution of the data. This is fair because the data which is uploaded earlier has better accuracy and does more contributions to the task initiator. Moreover, the updating procedure is accomplished by the smart contract so that no one can change it, and the reputation can be checked by all authority nodes.

5.4 Decentralization

In this work, when a vehicle firstly registers with the CA, the initial reputation is distributed via the CA and the local fog server instead of a reputation center. The reputation update is realized by smart contracts automatically, and there is no central reputation center, which might be compromised allowing illegal reputation updating or even impairing the reputation system. The local fog server only assists in distributing the initial reputation and calculating the updating factors. Vehicles access the blockchain to update their reputation and generate the corresponding proof via *zk-SNARK* algorithm after a task. Any central party is not needed to maintain the updated reputation of vehicles.

In addition, the local fog server also acts as a validator candidate in the PoA consensus mechanism. These validators are responsible for verifying and signing the blockchain and one of them will package all transactions into the block. If

a fog server is compromised, other validators will vote out the malicious fog server and another fog server in this area will be selected to take its place. Thus the compromised fog server can not damage the system which means that the PPDR-VCS can still work well even there exist some malicious nodes.

6 Performance Evaluation

6.1 Implementation Overview

We present a proof-of-concept implementation of our system based on Parity Ethereum, and accomplish extensive experiments to evaluate the performance of PPDR-VCS.

We implement our PPDR-VCS on a notebook with AMD Core R7-5800H CPU@3.20 GHz and 16.00 GB memory. The operating system is Ubuntu 20.04.3 LTS AMD64. We use libsnark to implement the *zk-SNARK* [18]. The hash function is SHA-256. The security-parameter of the Schnorr&ECDSA signature scheme and the ElGamal encryption scheme are 256 bits. We construct a blockchain testing network based on Proof of Authority (PoA) [27], which consists of Authority nodes and User nodes. In particular, fog servers play the role of Authority nodes in Parity PoA network, and they can be selected as validators to verify the transactions and issue blocks. While local fog servers can also perform the function of the User nodes that send reputation update transactions to the blockchain.

We deployed a few fog nodes in our experiments. Eclipse as the JAVA client communicates with the Parity blockchain via web3j to fulfill the interaction with smart contracts. We specify the validator list as configurations in the blockchain file, and we also encode the public parameters of the system in Java clients. The off-chain and on-chain performance are tested to reveal the system efficiency.

6.2 Off-Chain Performance

We evaluate the off-chain performance of the initial reputation distribution, the anonymous authentication based on ring signatures and updating factors calculation.

In the initial reputation distribution, the main computational costs come from the identifiers generation and verification as well as the ring signature. As shown in Fig. 6, we set the number of identifiers as $\{10, 30, 50, 70, 100\}$. The computation cost of identifier generation and verification is linearly increasing with the number of identifiers. From Fig. 6, it costs less than 1400ms to generate 100 identifiers. As for the verification process, the cost of verifying 100 identifiers at the same time is less than 2500ms. The procedure of ring signatures is similar to the message authentication in the Reputation update phase, we will discuss below.

We set a PK-list of the local fog server containing 1000 vehicles. The ring selection algorithm chooses n vehicles to generate a traceable ring signature of η_1

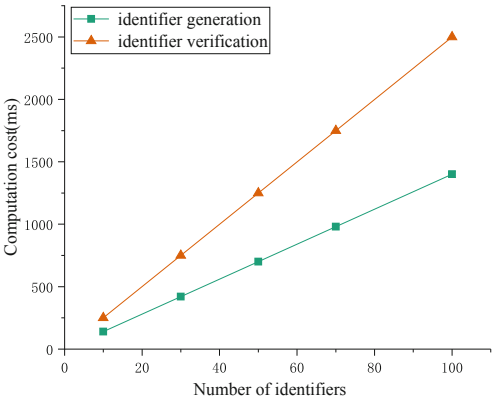

Fig. 6. Computation costs of identifier generation and verification

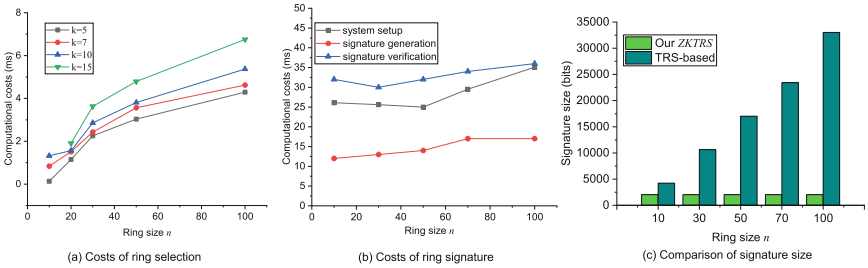

Fig. 7. Performance of the proposed $ZKTRS$

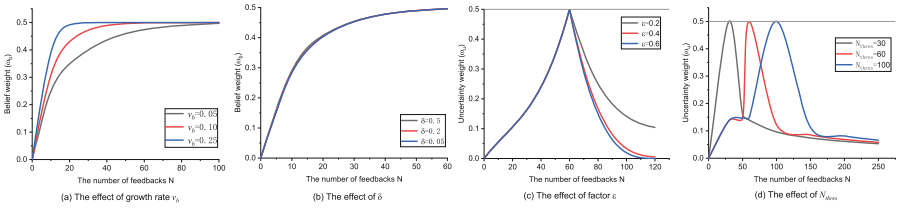

Fig. 8. The effects of parameter choices on ω_b and ω_u

and η_2 or the sensory data M. The computational cost varies with the ring size n. In each set of experiments with different n, which is set as $\{10, 20, 30, 50, 100\}$, we adopted an average result of 100-times round.

As shown in Fig. 7(a), we can see that the time cost for ring selection is increasing as the privacy level k increases. Because keeping k ring members of S_1 takes more time than randomly selecting k members from the PK-list. Given a fixed privacy level k, the computational cost of ring selection algorithm also increases as the ring size n grows. Because it also takes some time to randomly select n-k new ring members from the PK-list. Nevertheless, the ring selection

algorithm takes very little or negligible time, which is efficient while improving privacy protection.

When the privacy level of the system is fixed, the execution time of the ring signature is shown in Fig. 7(b). As the ring size increases, the time cost on the ring generation and verification are both maintains a fixed value because the compact ring signature algorithm has fixed calculation procedure. The cost of setup algorithm will increase with the ring size, however, it is still tens of milliseconds. So the computational cost of ring signature is acceptable.

Moreover, our traceable ring signature scheme yields small signature size. The Fig. 7(c) shows the comparison of our $ZKTRS$ with the baseline TRS-based scheme [28]. The signature size of TRS-based scheme [28] is multiple times larger than ours and grows with the ring size. The signature size of the $ZKTRS$ is only 2040 bits and doesn't grow with the ring size.

We test the performance of updating factors calculation by considering the truthfulness-based factor and time-based factor. We obtain the parameters from the Waze data set [29] as our default system parameters to make simulation environment. Figure 8(a) shows the effect of v_b on the belief weight ω_b. It is obvious that v_b controls the number of feedback N which is required to reach the maximum of ω_b. If the number of feedback in the task is small, v_b can be set lower. Otherwise, a higher v_b should be selected. Meanwhile, Fig. 8(b) shows the change of ω_b with the number of feedback N based on different δ. The number of feedback required to reach the maximum value is smaller while δ is higher. However, under the situation that all other parameters are unchanged, the change of δ has little effect on ω_b.

Figure 8(c) shows how ϵ affects the uncertainty weight ω_u. We can see that the higher the ϵ, the faster the decrease of ω_u. If the task initiator wants to decrease the effect of uncertainty feedback on the truthfulness, the parameter ϵ should be set higher, or vice versa. In Fig. 8(d), we can see three curves represent the trend of ω_u based on $N_{thres} = 30$, $N_{thres} = 60$ and $N_{thres} = 100$, respectively. All of them reach $\omega_u^{max} = 0.5$ at the threshold value and will decrease immediately after that $N = N_{thres}$ satisfies. Therefore, N_{thres} controls the number of feedback required to obtain the maximum value of ω_u and it can be chosen based on the task time and task area.

Figure 9(a) shows the trend of truthfulness-based factor Q_r is based on different θ and μ which controls the growth rate above and below zero, respectively. When $\tau_r > 0.5$, Q_r is positive and that the one based on a higher θ grows faster than the one based on a lower θ. On the contrary, when $\tau_r < 0.5$, Q_r is negative. As τ_r gradually decreases from 0.5, Q_r will decreases faster if it has a lower μ. The growth curve of time-based factor Q_t is described in Fig. 9(b) which shows that Q_t with a higher ρ decreases faster. When the task is time-sensitive or the task initiator needs data urgently, a higher value can be given to the parameter ρ, otherwise a lower value is a good choice.

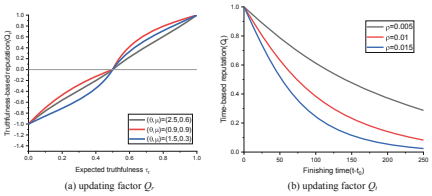

Fig. 9. The effects of different factors on Q_r and Q_t

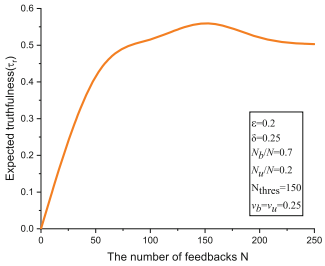

Fig. 10. The truthfulness variation with the number of feedback

Figure 10 describes the trend of τ_r with the number of feedback N where we assume that the ratio of belief feedback, the ratio of uncertainty feedback, v_b, v_u, δ, ϵ and N_{thres} are fixed. When $N < N_{thres}$, τ_r is increasing with the growth of N and obtains the maximum value at $N = N_{thres}$. On the contrary, τ_r is decreasing with N after the point $N = N_{thres}$. It is reasonable that the data which has more belief feedback has a higher truthfulness and it will decrease if more uncertainty feedback are received. We also evaluate the data quality Q_d based on Q_r and Q_t as shown in Fig. 11. Apparently, Q_d will increase with Q_r and Q_t. However, the weight of two parameters is controlled by γ. If a task acquires more truthful data, the γ can be set higher(> 0.5), and if a task is time-sensitive, γ can be set lower.

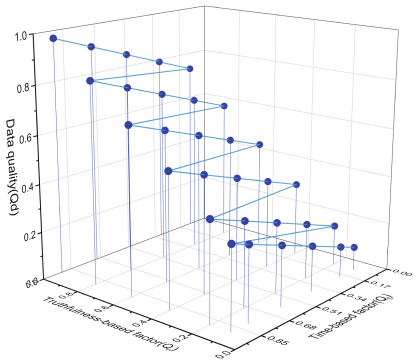

Fig. 11. The variation of Q_d based on Q_r and Q_t

6.3 On-Chain Performance

We evaluate the on-chain performance of smart reputation update through the confirmation time of the update transaction and the gas costs of deploying the contract and executing its functions on Parity Ethereum. Local fog server sends an updating transaction Tx by calling the reputation update algorithm in the update contract. Then, the fog validator verifies the correctness of the transaction. We conduct 20 sets of experiments to evaluate the on-chain performance, the average transaction confirmation time of the updating transaction Tx is 129 ms, which is efficient in the experiment environment.

Table 1. The comparison of PPDR-VCS with existing reputation systems

Proposal	Scenarios	Architecture	Technologies for identity privacy	Transparency	Fairness
Blomer [30]	Not given	Centralized	Group signatures	No	Yes
Zhai [31]	Online services	Decentralized	Linkable ring signatures	No	Yes
Soska [8]	E-commerce	Blockchain	Ring signatures	Yes	No
ARS-PS [9]	Retail marketing	Blockchain	Anonymous credentials	Yes	No
RepChain [11]	E-commerce	Blockchain	Blind signatures	Yes	Yes
BC-DRS [10]	E-commerce	Blockchain	None	Yes	Yes
PPDR-VCS	Vehicular crowdsensing	Decentralized	Traceable ring signatures	Yes	Yes

7 Related Work

In this section, we compare the existing works with our system from five aspects in Table 1.

7.1 Anonymous Reputation Systems

Privacy concerns and the prevention of different attacks for reputation systems are frequently discussed in the rencent literature. Extensive research have made efforts to design anonymous reputation systems to protect users' privacy. Blomer et al. [30] pointed out that the security properties for reputation systems are anonymity, traceability, linkability, and non-frameability, and they proposed an anonymous reputation system through group signatures and Σ-protocol. Zhai et al. [31] utilized verifiable shuffles [32] and linkable ring signatures [33] to propose a tracking-resistant anonymous reputation system. Liu et al. [9] proposed an anonymous reputation system based on PS signatures in retail marketing. Almost all proposed systems are combined with modern e-commerce services, such as eBay, Amazon, Yelp, etc., and the goal of these systems is to protect raters' privacy.

There exist a few works addressing the issues on privacy-preserving reputation system in mobile crowdsensing applications. Wang et al. [34] proposed an anonymous reputation management scheme. In their scheme, honest participants are vulnerable to tracking attacks, while malicious participants can keep large reputation values for some time even when they provide false data. To conquer the drawbacks of the work [34], Ma et al. [35] presented a privacy-preserving reputation management scheme for edge computing enhanced mobile crowdsensing. The scheme in [35] updates the reputation values based on the deviations of the sensing data to the final aggregating result. However, the scheme is based on centralized model, which can not meet our decentralized requirement. Different from the existing works, we focus on the privacy presevation of the *Uploader* and provide a scheme which can combine with a specific VCS application.

7.2 Reputation Calculation Model

Jøsang [15] and Yu et al. [36] proposed trust models to calculate reputation scores based on rating feedback, which leverage the ratio of positive feedback to the all. However, there exist threats such as ballot and obfuscation stuffing in Jøsang's belief models [15], and Dempster-Shafer model [36] does not consider the degree of participation and data quality when computing the reputation score. To address these issues, Bhattacharjee et al. [16] proposed a quality and quantity-unified QoI metric for published information in a mobile crowdsensing system. In order to model the expected truthfulness of the published information, they use generalized Richard's curve and Kohlsrausch relaxation function to calculate the weights to belief and uncertainty masses, respectively. They pointed out that their approach outperforms Jøsang's belief and Dempster-Shafer based reputation models in some aspects of classification, incentivization, and scalability.

We presented our data quality quantification for vehicular crowdsensing scenario. Different from QnQ [16], we modify the design of the belief and uncertainty coefficient to satisfy our scenario. Furthermore, QnQ only considers the truthfulness of an event to calculate the aggregate reputation score. In our system, another time-based factor, which affects the reputation value in time-sensitive crowdsensing tasks, is also taken into account.

7.3 Blockchain-Enabled Reputation Systems

To build a decentralized system, the blockchain has been taken into consideration to construct reputation systems [8–11] in E-commerce environment. Soska et al. [8] proposed an anonymous reputation system based on ring signature and the robust transaction chain property. Different from our system, the ring signature in [8] is used to protect the identity privacy of the customer, but the size of the ring signature is not constant. Liu et al. [9] proposed an anonymous reputation system based on the PoS Blockchain architecture [37]. And they focus on the efficiency and scalability issues of a blockchain-based architecture. Zhou et al. [10] gave a blockchain-based decentralized reputation system in the E-commerce

environment. They did not consider the privacy pretection of users, whose reputation scores are stored on the blockchain and can be accessed by others. Li et al. [11] presented RepChain, a privacy preserving reputation system for E-commerce platforms based on the blockchain. Different from the existing work, our system is proposed under the VCS scenario. And we aim to accomplish privacy-preserving trusted reputation update on the premise of minimizing the number of times accessing the blockchain.

8 Conclusion

In this paper, we have investigated the security and privacy issues of reputation systems in VCS scenarios. We propose a privacy-preserving decentralized reputation system by utilizing *zk-SNARK*, traceable ring signature and the blockchain technology. The privacy of vehicles is protected unconditionally for their good behaviors. Any dishonest vehicle cannot boost their reputation arbitrarily. Sybil attacks and collusion attacks are resisted in our system. The reputation of vehicles is updated trustworthily depending on the quantified data quality. We have also implemented a prototype system based on Ethereum to verify its performance and feasibility. For future work, we will study how to enrich our current design and improve the decentralized feature [38] of our system. Besides, multiple updating factors should also be taken into account to meet the demands of different tasks since there are different factors that affect the performance.

Acknowledgments. This work was supported by National Natural Science Foundation of China under Grant 61802217, in part by the Natural Science Foundation of Shandong Province under Grant ZR2023MF082, in part by Qingdao Science and Technology Plan Key Research and Development Project under Grant 22-3-4-xxgg-10-gx, and in part by Original Exploration Project of Qingdao Natural Science Foundation under Grant 23-2-1-164-zyyd-jch.

References

1. Ni, J., Zhang, A., Lin, X., Shen, X.: Security, privacy, and fairness in fog-based vehicular crowdsensing. IEEE Commun. Mag. **55**(6), 146–152 (2017)
2. Li, M., Chen, Y., Zheng, S., Hu, D., Lal, C., Conti, M.: Privacy-preserving navigation supporting similar queries in vehicular networks. IEEE Trans. Dependable Secure Comput. **19**(2), 1133–1148 (2020)
3. Work-Brows by the Type of Client. https://www.pentagram.com/work/waze
4. Wang, L., Cao, Z., Zhou, P., Zhao, X.: Towards a smart privacy-preserving incentive mechanism for vehicular crowd sensing. Secur. Commun. Netw. **2021**, 1–16 (2021)
5. Wang, X., Cheng, W., Mohapatra, P., Abdelzaher, T.F.: Enabling reputation and trust in privacy-preserving mobile sensing. IEEE Trans. Mob. Comput. **13**(12), 2777–2790 (2013)
6. Singh, S.K., Rathore, S., Park, J.H.: BlockIoTIntelligence: a blockchain-enabled intelligent IoT architecture with artificial intelligence. Futur. Gener. Comput. Syst. **110**, 721–743 (2020)

7. Xu, Y., Ren, J., Wang, G., Zhang, C., Yang, J., Zhang, Y.: A blockchain-based nonrepudiation network computing service scheme for industrial IoT. IEEE Trans. Industr. Inf. **15**(6), 3632–3641 (2019)
8. Soska, K., Kwon, A., Christin, N., Devadas, S.: Beaver: a decentralized anonymous marketplace with secure reputation. Cryptology ePrint Archive (2016)
9. Liu, D., Alahmadi, A., Ni, J., Lin, X., Shen, X.: Anonymous reputation system for IIoT-enabled retail marketing atop POS blockchain. IEEE Trans. Industr. Inf. **15**(6), 3527–3537 (2019)
10. Zhou, Z., Wang, M., Yang, C.N., Fu, Z., Sun, X., Wu, Q.J.: Blockchain-based decentralized reputation system in e-commerce environment. Futur. Gener. Comput. Syst. **124**, 155–167 (2021)
11. Li, M., Zhu, L., Zhang, Z., Lal, C., Conti, M., Alazab, M.: Anonymous and verifiable reputation system for e-commerce platforms based on blockchain. IEEE Trans. Netw. Serv. Manage. **18**(4), 4434–4449 (2021)
12. Wang, G., Wang, B., Wang, T., Nika, A., Zheng, H., Zhao, B.: Defending against sybil devices in crowdsourced mapping services. In: Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services, pp. 179–191 (2016)
13. Li, H., Chen, Q., Zhu, H., Ma, D., Wen, H., Shen, X.: Privacy leakage via de-anonymization and aggregation in heterogeneous social networks. IEEE Trans. Dependable Secure Comput. **17**(2), 350–362 (2017)
14. Wang, L., Lin, X., Zima, E., Ma, C.: Towards airbnb-like privacy-enhanced private parking spot sharing based on blockchain. IEEE Trans. Veh. Technol. **69**(3), 2411–2423 (2020)
15. Jsang, A.: An algebra for assessing trust in certification chains (1999)
16. Bhattacharjee, S., Ghosh, N., Shah, V., Das, S.: Q n Q: quality and quantity based unified approach for secure and trustworthy mobile rowdsensing. IEEE Trans. Mob. Comput. **19**(1), 200–216 (2018)
17. Xu, Z., Yang, W., Xiong, Z., Wang, J., Liu, G.: TPSense: a framework for event-reports trustworthiness evaluation in privacy-preserving vehicular crowdsensing systems. J. Signal Process. Syst. **93**(2–3), 209–219 (2021)
18. Groth, J.: On the size of pairing-based non-interactive arguments. In: Fischlin, M., Coron, J.-S. (eds.) EUROCRYPT 2016, Part II. LNCS, vol. 9666, pp. 305–326. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49896-5_11
19. Fleischhacker, N., Krupp, J., Malavolta, G., Schneider, J., Schröder, D., Simkin, M.: Efficient unlinkable sanitizable signatures from signatures with re-randomizable keys. In: Cheng, C.-M., Chung, K.-M., Persiano, G., Yang, B.-Y. (eds.) PKC 2016, Part I. LNCS, vol. 9614, pp. 301–330. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49384-7_12
20. Gennaro, R., Gentry, C., Parno, B., Raykova, M.: Quadratic span programs and succinct NIZKs without PCPs. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 626–645. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38348-9_37
21. Wang, L., Zhang, G., Ma, C.: A survey of ring signature. Front. Electr. Electron. Eng. China **3**, 10–19 (2008)
22. Wang, L., Lin, X., Qu, L., Ma, C.: Ring selection for ring signature-based privacy protection in VANETs. In: ICC 2020–2020 IEEE International Conference on Communications (ICC), pp. 1–6 (2020)
23. Ram, N., Grimm, K.: Growth curve modeling and longitudinal factor analysis (2015)

24. Anderssen, R., Husain, S., Loy, R.: The Kohlrausch function: properties and applications. Anziam J. **45**, 800–816 (2003)
25. Kahneman, D., Tversky, A.: Prospect theory: an analysis of decision under risk. In: Handbook of the Fundamentals of Financial Decision Making: Part I, pp. 99–127 (2013)
26. Fujisaki, E., Okamoto, T.: Secure integration of asymmetric and symmetric encryption schemes. In: Wiener, M. (ed.) CRYPTO 1999. LNCS, vol. 1666, pp. 537–554. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48405-1_34
27. Alofs, M.: Blockchain: proof of authority (2018)
28. Fujisaki, E., Suzuki, K.: Traceable ring signature. In: Okamoto, T., Wang, X. (eds.) PKC 2007. LNCS, vol. 4450, pp. 181–200. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-71677-8_13
29. Barnwal, R., Ghosh, N., Ghosh, S., Das, S.: Enhancing reliability of vehicular participatory sensing network: a Bayesian approach. In: 2016 IEEE International Conference on Smart Computing (SMARTCOMP), pp. 1–8 (2016)
30. Blömer, J., Eidens, F., Juhnke, J.: Practical, anonymous, and publicly linkable universally-composable reputation systems. In: Smart, N.P. (ed.) CT-RSA 2018. LNCS, vol. 10808, pp. 470–490. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-76953-0_25
31. Zhai, E., Wolinsky, D.I., Chen, R., Syta, E., Teng, C., Ford, B.: AnonRep: towards tracking-resistant anonymous reputation. In: 13th $\{USENIX\}$ Symposium on Networked Systems Design and Implementation ($\{NSDI\}$ 2016), pp. 583–596 (2016)
32. Neff, C.A.: A verifiable secret shuffle and its application to e-voting. In: Proceedings of the 8th ACM Conference on Computer and Communications Security, pp. 116–125 (2001)
33. Liu, J.K., Wong, D.S.: Linkable ring signatures: security models and new schemes. In: Gervasi, O., Gavrilova, M.L., Kumar, V., Laganà, A., Lee, H.P., Mun, Y., Taniar, D., Tan, C.J.K. (eds.) ICCSA 2005. LNCS, vol. 3481, pp. 614–623. Springer, Heidelberg (2005). https://doi.org/10.1007/11424826_65
34. Wang, X., Cheng, W., Mohapatra, P., Abdelzaher, T.: Enabling reputation and trust in privacy-preserving mobile sensing. IEEE Trans. Mob. Comput. **13**(12), 2777–2790 (2013)
35. Ma, L., Liu, X., Pei, Q., Xiang, Y.: Privacy-preserving reputation management for edge computing enhanced mobile crowdsensing. IEEE Trans. Serv. Comput. **12**(5), 786–799 (2018)
36. Yu, B., Singh, M.: An evidential model of distributed reputation management. In: Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems: Part 1, pp. 294–301 (2002)
37. Kiayias, A., Russell, A., David, B., Oliynykov, R.: Ouroboros: a provably secure proof-of-stake blockchain protocol. In: Katz, J., Shacham, H. (eds.) CRYPTO 2017. LNCS, vol. 10401, pp. 357–388. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-63688-7_12
38. Wang, L., Zhao, X., Lu, Z., Wang, L., Zhang, S.: Enhancing privacy preservation and trustworthiness for decentralized federated learning. Inf. Sci. **628**, 449–468 (2023)