Truthful Crowdsensed Data Trading Based on Reverse Auction and Blockchain

Baoyi An[1], Mingjun Xiao[1(✉)], An Liu[2], Guoju Gao[1], and Hui Zhao[1]

[1] School of Computer Science and Technology, Suzhou Institute for Advanced Study, University of Science and Technology of China, Hefei, China
xiaomj@ustc.edu.cn

[2] School of Computer Science and Technology, Soochow University, Suzhou, China

Abstract. Crowdsensed Data Trading (CDT) is a novel data trading paradigm, in which each data consumer can publicize its demand as some crowdsensing tasks, and some mobile users can compete for these tasks, collect the corresponding data, and sell the results to the consumers. Existing CDT systems either depend on a trusted data trading broker or cannot ensure sellers to report costs honestly. To address this problem, we propose a Reverse-Auction-and-blockchain-based crowdsensed Data Trading (RADT) system, mainly containing a smart contract, called RADToken. We adopt a greedy strategy to determine winners, and prove the truthfulness and individual rationality of the whole reverse auction process. Moreover, we exploit the smart contract with a series of devises to enforce mutually untrusted parties to participate in the data trading honestly. Additionally, we also deploy RADToken on an Ethereum test network to demonstrate its significant performances. To the best of our knowledge, this is the first CDT work that exploits both auction and blockchain to ensure the truthfulness of the whole data trading process.

1 Introduction

Owing to the huge potential economic value of data resources, many online data trading systems [12] have emerged in recent years, such as CitizenMe, DataExchange, Datacoup, Factual, and Terbine, etc., whereby data consumers can search and purchase their interested data. However, most data in the real world are preserved by few research institutions or companies only for their own analysis purposes rather than sharing them with others who have data needs but cannot afford to collect data by themselves, which causes trouble to the availability of data. Consequently, the volumes of data in trading systems are still very limited, which has significantly suppressed the increasing market demand for data [24]. To tackle this problem, a novel data trading paradigm, called Crowdsensed Data Trading (CDT), is proposed, in which the mobile crowdsensing technology is adopted to provide data resources for trading, i.e., a large crowd of mobile users are leveraged to collect data with their smart phones [17,24].

G. Li et al. (Eds.): DASFAA 2019, LNCS 11446, pp. 292–309, 2019.
https://doi.org/10.1007/978-3-030-18576-3_18

In general, a typical CDT system (e.g., Thingful [1], Thingspeak [2]) includes a data trading broker, some data consumers, and data sellers (a.k.a., crowdsensing workers). The broker employs a large amount of sellers to collect data according to some requirements, and then sells the sensed data to the consumers who are interested in. So far, there have been a few works focusing on the CDT system design. For example, [24] proposes a profit-driven data collection framework for crowd-sensed data markets, called VENUS, in which a data procurement auction is adopted to determine the minimum payment for each data collection. A data sharing market is introduced in [11], where the sensed data is saved and processed in user devices locally and shared among users in a P2P manner. A brokerage-based market is launched in [23], where sellers and consumers propose their selling and buying quantities, respectively, to match the market supply and demand in the trading platform. However, these CDT systems have to depend on a data trading broker, making those data consumers worry about the truthfulness and even unwilling to use the systems.

On the other hand, blockchain [16], a newly-emerging decentralized transaction recording technology, shows a glimpse of solutions to fairness and transparency issues which is resistant to modification of the data. In addition, blockchains allow mutually distrusted users to complete data exchange or transaction securely without a centralized truthful intermediary, avoiding high legal and transactional costs [13]. Smart contracts [5] are some complex programs deployed on blockchains which can automatically execute operations according to treaty conditions. An important advantage of smart contracts is that they can enforce the participants, who might not trust each other, to fulfill their obligations. Due to this characteristic, smart contracts are introduced into CDT systems as a truthful broker to conduct the data trading between sellers and consumers. For instance, [6] proposes a CDT framework which enables efficient truth discovery over encrypted crowdsensed data streams and knowledge monetization on smart contract. A reliable mechanism that allows consumers to search directly over encrypted data is also implemented on blockchain [10].

Although blockchain-based CDT systems can create trust between sellers and consumers of data trading, they cannot guarantee the truthfulness of individuals, i.e., sellers might report fake data collection costs to achieve more rewards. As we know, auction mechanisms can ensure the participants to report their bids honestly, which have been widely used in crowdsensing systems [8,21]. Hence, to construct fully truthful data trading systems on blockchain, we propose a Reverse-Auction-and-blockchain-based crowdsensed Data Trading (RADT) system in this paper, which mainly contains a smart contract, called RADToken. Each consumer can start a RADT by issuing the data demand via RADToken (e.g., report traffic conditions of multiple locations), and sellers who have registered in RADToken can bid for their interested tasks. RADToken will automatically execute to determine the winners and payments. After sellers complete the data collection and submit the results, the consumer will pay some rewards to sellers via RADToken. Since bids on blockchain is transparent and public visible, RADToken takes a two-stage bidding strategy to ensure the security of bids.

Moreover, both consumer and sellers are asked for a deposit which keeps them comply with the prescribed rules. Once they finish their obligations, RADToken will automatically and transparently transfer their deposits and payments. We summarize the contributions of this paper as follows:

1. We propose a CDT system based on reverse auction and blockchain, i.e., RADT. It does not need any truthful broker but can provide truthful data trading between mutually untrusted consumers and sellers. To the best of our knowledge, this is the first CDT system that exploits both auction theory and blockchain technology to ensure the truthfulness of the whole data trading.
2. We design a reverse-auction-based smart contract for the RADT system, i.e., RADToken, where the reliability of each seller is taken into consideration and a greedy strategy is used to determine winners. Moreover, we prove the truthfulness and individual rationality of the whole auction process, which implies that all sellers will report their data collection costs honestly.
3. To ensure the truthfulness of the data trading process, we devise some modifiers and set life span limitations for each procedure to filter the illegal invokes which can resist certificate forgery and tamper attacks effectively. Meanwhile, we adopt a two-stage bid strategy to protect bid privacy in auction procedure and use symmetric and asymmetric encryptions for data delivery procedure which can protect the confidentiality of the sensed data.
4. We implement a prototype of the RADT system and deploy RADToken to an Ethereum test network. Extensive simulations are conducted to demonstrate the significant performances and the practicability of RADToken.

The paper is organized as follows. In Sect. 3, we present a system model for RADT and analyze the security in Sect. 4. The reverse auction is elaborated in Sect. 5. We carry on the theoretical analysis in Sect. 6. We present simulations and evaluations in Sect. 7 and review related works in Sect. 8. Finally, we conclude in Sect. 9.

2 Preliminaries

We first introduce the background of Ethereum before the system overview.

2.1 Account Types

- **Externally Owned Accounts (EOAs).** An EOA only has a balance which can send transactions either to transfer ether or to trigger contract code.
- **Smart Contract Accounts.** A smart contract has a balance and associated code. Code execution is triggered by transactions from EOAs.

A contract is invoked by a transaction and is run by Ethereum Virtual Machine on each node participating in the network as part of their verification of new blocks. Contracts like *autonomous agents* have direct control over their balances and dictionary (key/value-datatype) storage.

2.2 Transaction

The term *transaction* is used in Ethereum to refer to the signed data package. Its *VALUE* field is the amount of **wei** to be transferred from the sender to the recipient. All values are denominated in units of wei: 1 **ether** is 10^{18} **wei**. Once a smart contract receives a transaction (*msg*), it can obtain two parameters:

- **msg.sender** is sender's account.
- **msg.value** is the amount of **ether** that sender transfer to it.

2.3 Gas System

Ethereum charges a fee (gas) per computational step to prevent deliberate attacks and abuse on Ethereum. Each transaction is required to include a gas limit and a fee that it is willing to pay per gas. If the total gas used for the computational steps spawned by the transaction is not greater than the gas limit, the transaction will be processed. Otherwise, all modifications are reverted. The excess gas is reimbursed to the sender. The total cost involves two aspects:

- **gasUsed** is the total gas which is consumed by the transaction.
- **gasPrice** is the price of one unit gas that is specified in the transaction.

Hence, the total cost = gasUsed × gasPrice.

3 System Overview

3.1 The RADT System

Figure 1 illustrates the RADT system including 4 major entities, i.e., *consumer*, RADToken, *Secure Cloud Server (SCS)* and *seller*. The consumer sends data collection job requirements to RADToken, a smart contract on Ethereum. The job includes a set of Points of Interest (PoIs) which correspond to the location-sensitive sensing tasks, denoted as $\mathcal{T} = \{t_1, t_2, \cdots, t_l\}$. Sellers who participate in the job, denoted by $\mathcal{W} = \{w_1, w_2, \cdots, w_n\}$. Each seller w_i can submit a bid β_i for their interested tasks $\mathcal{T}_i$ ($\subseteq \mathcal{T}$). Moreover, we denote the sellers who can execute t_j by $\mathcal{W}_j$ ($\subseteq \mathcal{W}$). RADToken as a broker then executes a reverse auction to select winners $\mathcal{S}(\subseteq \mathcal{W})$ and determine payments $\mathcal{P} = \{p_i | s_i \in \mathcal{S}\}$ for the winners. The winners upload the encrypted sensed data ($EnData$) to SCS where the consumer can download and decrypt $EnData$ to obtain the real sensed data ($Data$). The consumer and sellers can get the refund and payments respectively after the job ends. We specify two key features of a smart contract [13]:

- *Timing.* A smart contract has a time clock which is modeled as a continuously increasing variable *now*. *now* is an alias for a timestamp of the blockchain.
- *Function Modifier.* Modifiers are inheritable properties of contracts which are used to automatically check a condition prior to executing the function.

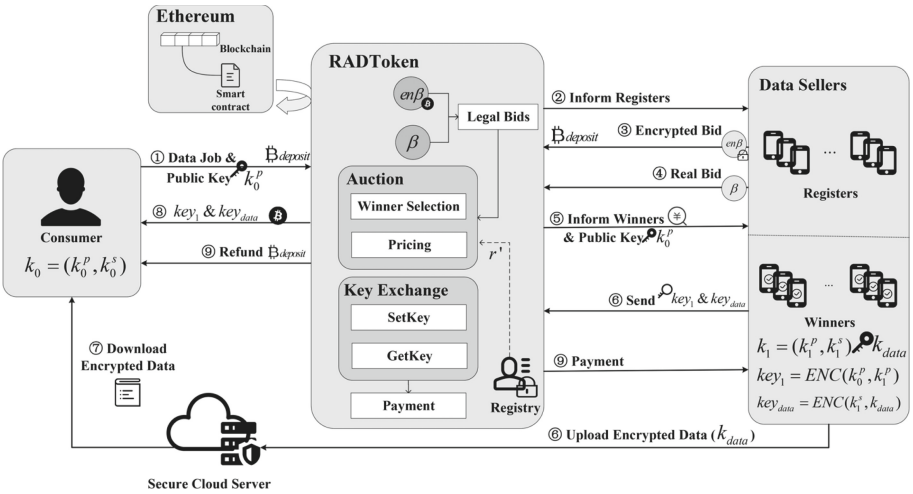

Fig. 1. The RADT system

3.2 The Workflow of the RADT System

Sellers register in RADToken to be informed once a data collection job appears. RADToken maintains **Registry** which contains a dictionary $\mathcal{R}$ to record sellers' reliabilities and $\mathcal{R}(w_i) = r_i$. The system works as follows:

Initiate($\mathcal{T}, \tau_{bid}, \tau_{reveal}, \tau_{exec}, k_0^p$):
1. require \$deposit($msg.value$) $\geq guaranty$.
2. set $\mathcal{Q} = msg.sender$ and store k_0^p.
3. set $\mathcal{T} = \mathcal{T}$, $t_{bid} = now + \tau_{bid}$, $t_{reveal} = t_{bid} + \tau_{reveal}$, $t_{exec} = t_{reveal} + \tau_{exec}$.
4. trigger $Notify$ event to inform the registered data sellers.

Fig. 2. The initiate function

Step 1: Job Initialization. The consumer describes a data collection job with requirements: $\mathcal{T}$, bid commitment duration τ_{bid}, bid reveal duration τ_{reveal} and job execution duration τ_{exec}. The consumer generates a pair of keys $k_0 = (k_0^p, k_0^s)$, where k_0^p is his public key and k_0^s is his private key. Then, he sends a transaction which contains the job description and the public key k_0^p to RADToken. The detailed function is given in Fig. 2.

Step 2: Notify Data Providers. *Initiate()* notifies all registered sellers of the new job. A seller provides a deposit ($\$deposit_i$) for participating in the job.

To prevent the bids from being intercepted, a two-stage bidding strategy is adopted in Sect. 4.2. Each seller has an encrypted bid $enB_i = (\mathcal{T}_i, en\beta_i)$ and a real

bid $B_i = (\mathcal{T}_i, \beta_i)$. RADToken uses two dictionaries $EnBids$ and $Bids$ to record them respectively, where $EnBids(w_i) = en\beta_i$ and $Bids(w_i) = (T_i, \beta_i, \$deposit_i)$.

CommitBid($\mathcal{T}_i, en\beta_i$) payable:
1. require now $\leq t_{bid}$ and $msg.value \geq guaranty$.
2. add $(msg.sender, \mathcal{T}_i, msg.value)$ to $Bids$. ▷$w_i = msg.sender$, $deposit_i = msg.value$
3. add $(msg.sender, en\beta_i)$ to $EnBids$.

RevealBid($\mathcal{T}_i, \beta_i, nonce_i$):
1. require $t_{bid} \leq$ now $\leq t_{reveal}$.
2. require $EnBids(msg.sender)$=sha3$(\mathcal{T}_i, \beta_i, nonce_i)$.
3. require $Bids(msg.sender).T = \mathcal{T}_i$.
4. add β_i to $Bids(msg.sender)$.

Fig. 3. The two-stage bidding procedure

Step 3: Commit Encrypted Bid. A seller computes his encrypted bid $en\beta_i$ using Secure Hash Algorithm-3 (SHA-3) [4]. SHA-3 takes as input his account w_i, his bid β_i and a randomly selected $nonce_i$, i.e., $en\beta_i = sha3(w_i, \beta_i, nonce_i)$. *CommitBid()* in RADToken takes as input a binary tuple enB_i and records it.
Step 4: Reveal Real Bid. RADToken is invoked by a seller w_i to send $\mathcal{T}_i$, β_i and $nonce_i$. *RevealBid()* will check the bid's authenticity and record the legal bid. If w_i sends illegal bid, he is untruthful and his $\$deposit_i$ will be forfeited. The detailed functions of *CommitBid()* and *RevealBid()* are given in Fig. 3.

WinnerSelection():
1. require now $\geq t_{reveal}$ and $msg.sender = consumer$.
2. compute $(\mathcal{S}, \mathcal{C})$ according to Winner Selection algorithm in Fig. 9.

Pricing(s_i):
1. require $msg.sender = consumer$.
2. compute $\mathcal{P}(s_i)$ according to Pricing algorithm in Fig. 10.
3. trigger *AuctionEnd* event to inform all winners($\mathcal{S}$) and send them k_0^p.

Fig. 4. The reverse auction procedure

Step 5: Inform Winners. RADToken executes a reverse auction in Fig. 4 to select winners $\mathcal{S}$ and determine payments for all winners. All winners will get informed the auction results and public key k_0^p. The detailed reverse auction process is in Sect. 5.
Step 6: Send Keys. To ensure the confidentiality of the sensed data, w_i generates a symmetric key $k_{data,i}$ and encrypts $Data_i$ with $k_{data,i}$. After uploading the encrypted data ($Endata_i$) to SCS, w_i generates a pair of asymmetric keys $k_{1,i} = (k_{1,i}^p, k_{1,i}^s)$. And he obtains $key_{data,i}$ and $key_{1,i}$ by encrypting $k_{data,i}$ and $k_{1,i}^p$ as described in Sect. 4.3. Then the seller invokes $SetKey()$ to delivery

$key_{data,i}$ and $key_{1,i}$. RADToken maintains two dictionaries KDs and $K1s$ to store $key_{data,i}$ and $key_{1,i}$ respectively, as shown in Fig. 5.

SetKey($key_{data,i}, key_{1,i}$):
1. add $(msg.sender, key_{data,i})$ to KDs.
2. add $(msg.sender, key_{1,i})$ to $K1s$.

GetKey(s_i):
1. require $msg.sender = consumer$.
2. require $\$rewards \geq \sum_{s_i \in \mathcal{S}} \mathcal{P}(s_i)$. ▷ consumer transferred $rewards before
3. send $(KDs(s_i), K1s(s_i), \mathcal{R}(s_i))$ to $consumer$.

Fig. 5. The key exchange procedure

Step 7: Download $EnData$**.** The consumer downloads $EnData$ from SCS.
Step 8: Get Keys. In Fig. 5, the consumer can invoke $GetKey()$ to get keys for decrypting $EnData$. Before the invocation, he should transfer some ethers ($\$rewards$) to RADToken which are no less than the total payments.

Refund():
1. require now $\geq t_{exec}$.
2. compute untruthful sellers' total deposits which is $fines.
3. compute the remaining rewards $\$rewards = \$rewards - \sum_{s_i \in \mathcal{S}} \mathcal{P}(s_i)$.
4. transfer $deposit+$fines+$rewards to consumer.

Payment():
1. require $Bids(msg.sender).\beta \neq 0$. ▷ untruthful according to step 4 in Fig. 3
2. if $msg.sender \in \mathcal{S}$, require KDs(msg.sender)$\neq$ 0 and K1s(msg.sender)$\neq$ 0.
3. transfer the final payment $\$P = \mathcal{P}(msg.sender) + \$deposit_i$+$fine to $msg.sender$

Fig. 6. The refund and payment procedure

Step 9: Payment and Refund. Upon receiving the total rewards, RADToken will return the deposit to the consumer. Each winner can get his payment only when he had sent his keys. The functions are detailed in Fig. 6.

4 Security Analysis

4.1 Robustness of RADToken

We use some modifiers introduced at the beginning of Sect. 4 in Fig. 7 to ensure that RADToken can run steadily. The keyword *require* can roll back all states without deducting gas when encountering some invalid codes. The robustness of RADT is guaranteed by the following points:

```
Modifiers:
1. modifier onlyBefore(uint time)  require(now < time);
2. modifier onlyAfter (uint time)  require(now > time);
3. modifier onlyTrue (uint flag)  require(flag == true);
4. modifier onlyFalse (uint flag)  require(flag == false);
```

Fig. 7. Some modifiers

1. **Only one job in one round.** Once a consumer invokes RADToken to launch a job like in Fig. 2, others cannot invoke it until the job ends. If there exists an active job, the value of flag *jobEnded* whose default is *true* would be $false$, which cannot pass the check of $onlyTrue(jobEnded)$ in *Initiate*, so that other consumers will be rejected.
2. **Each participant should offer deposit.** We set *Initiate()* and *CommitBid() payable*, a keyword of smart contract, which requires invokers to transfer ethers to RADToken. We assume that the consumer will transfer rewards and will not quit the RADT system midway. Otherwise his deposit offered in *Initiate()* will be fined as a compensation for sellers and he even cannot get keys to decrypt *EnData*. Each seller should invoke *CommitBid()* with his deposit, which urges him to reveal his bid truthfully.

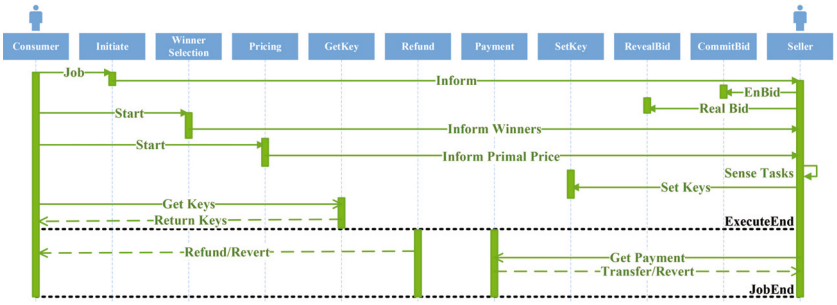

Fig. 8. The sequence diagram of RADT system

3. **Each procedure only be executed orderly.** Every function in our RADToken is an independent entry through separate calls. We also set some time modifiers to ensure the safety. We illustrate the sequence diagram of RADT in Fig. 8. For example, *RevealBid()* should be invoked after t_{bid} and before t_{reveal} where the auction only can be executed after *RevealBid()*.

4.2 Security and Truthfulness of Bid

Everything on blockchain is transparent, which enables a potential adversary to reconstruct the entire transaction history and find out the meanings and logic

behind it. That means the adversary can grab other bids and make a bid in his favour. Thus, we should consider how to guarantee the security of each bid.

Security. Since smart contracts cannot handle complex encryption and decryption, we adopt a two-stage bidding strategy to protect bids. At the first stage, a seller w_i can encrypt his bid β_i using SHA-3 as follows:

$$en\beta_i = \text{“0}x\text{”} + ethereumjs.ABI.\ solidity SHA3([\text{“}address\text{”}, \text{“}uint256\text{”}, \text{“}uint256\text{”}], [w_i, \beta_i, nonce_i]).toString(\text{“}hex\text{”})$$

Then, he sends enB_i to RADToken but other users can only read a hash value $en\beta_i$ rather than the real bid β_i. Moreover, a wise adversary will not try to crack the hash value which is impossible to succeed. So β_i is secure until t_{bid}.

Truthfulness. Even though an adversary can obtain other bids which will be revealed at the second stage, he cannot adjust his bid which will make him untruthful for the bid check of *RevelBid()* in Fig. 3. First, it recomputes the hash value $sha3(msg.sender, \beta_i, nonce_i)$ of w_i and compares it with his recorded $en\beta_i$ (step 2). RADToken also checks if his new submitted tasks are same with $\mathcal{T}_i$ (step 3). w_i will be accepted as a truthful seller only when he passes the check.

4.3 Confidentiality of Data

Symmetric encryption has more efficiency and less computation overhead. So we use it to encrypt data and asymmetric encryption to protect keys as below:

1. A winner s_i encrypts his sensed data with a symmetric key k_{data}, i.e., $EnData_i = ENC(Data_i, k_{data,i})$. Then he uploads the $EnData_i$ to SCS.
2. s_i uses his private key to encrypt k_{data} and get $key_{data,i} = ENC(k^s_{1,i}, k_{data,i})$.
3. s_i encrypts $k^p_{1,i}$ with k^p_0, i.e., $key_{1,i} = ENC(k^p_0, k^p_{1,i})$.
4. s_i invokes $SetKey()$ to send $key_{data,i}$ and $key_{1,i}$ to RADToken.
5. The consumer downloads $EnData_i$ and invokes $GetKey()$ to get $(key_{data,i}, key_{1,i})$.
6. The consumer decrypts $key_{1,i}$ with k^s_0 to get $k^p_{1,i} = DEC(key_{1,i}, k^s_0)$.
7. The consumer then decrypts key_{data} with $k^p_{1,i}$ to get $k_{data,i} = DEC(key_{data,i}, k^p_{1,i})$.
8. Finally, the consumer can obtain the real data $Data_i = DEC(EnData_i, k_{data,i})$.

5 The Reverse Auction Mechanism of RADT

5.1 Problem Formulation

As more sellers take part in the job, the actual sensed data would exceed the reliability requirements $\boldsymbol{\epsilon} = \{\epsilon_j | t_j \in \mathcal{T}\}$, whereas it also increases the total costs $\mathcal{C}$. Furthermore, we adopt $\sigma^{\mathcal{S}}_j$ to denote the overall reliability that all winners

contribute to $t_j \in \mathcal{T}$. In order to estimate $\sigma_j^{\mathcal{S}}$, we compute the sum of the reliabilities of the winners who process t_j as follows:

$$\sigma_j^{\mathcal{S}} = \sum\nolimits_{s_i \in \mathcal{S} \cap \mathcal{W}_j} r_i \tag{1}$$

Due to the truthfulness of sellers, we regard a seller w_i's bid β_i as his cost (i.e., $c_i = \beta_i$). The goal of the auction is to find a subset of sellers that minimize the overall cost while satisfying the reliability requirements of data. Hence, the RADT problem can be formulated as follows:

$$Minimize: C(S) = \sum\nolimits_{s_i \in \mathcal{S}} \beta_i, \ //c_i = \beta_i \tag{2}$$

$$Subject\ to: \quad \mathcal{S} \subseteq \mathcal{W} \tag{3}$$

$$\sigma_j^{\mathcal{S}} \geq \epsilon_j, \ 1 \leq j \leq l \tag{4}$$

Here, Eq. 4 indicates that the total reliability of task t_j is no less than ϵ_j.

The auction first selects winners who minimize the total costs under reliability constraints. Then, it determines payments for winners so that the whole auction satisfies truthfulness and individual rationality which are defined as follows:

Definition 1 ***(Truthfulness).*** *Let B_i be the truthful bid and B_i' be the untruthful bid where the payments are $p_i(B_i)$ and $p_i(B_i')$ respectively. Then, if*

$$p_i(B_i) - c_i \geq p_i(B_i') - c_i, \tag{5}$$

we say that the auction mechanism is truthful.

Definition 2 ***(Individual Rationality).*** *The payoff for B_i is non-negative,*

$$p_i(B_i) - c_i \geq 0. \tag{6}$$

5.2 The Winner Selection Algorithm of RADT

WinnerSelection():

1. repeat until $G(\mathcal{S}) = \sum_{j=1}^{l} \epsilon_j$:
   (a) for $\forall B_i$, compute $\rho_i = \frac{v_i(\mathcal{S})}{\beta_i}$;
   (b) record the index of the maximum ρ_i as i^*;
   (c) add s_{i^*} to $\mathcal{S}$, set $\mathcal{C} = \mathcal{C} + \beta_{i^*}$;
2. return $(\mathcal{S}, \mathcal{C})$.

Fig. 9. The reliability-aware winner selection algorithm

The RADT problem is NP-hard, because the minimum weight set cover problem is to find a subset that minimize the total weight which can be polynomial-time reducible to the RADT problem according to [8]. So we propose a greedy

algorithm to solve RADT. A seller who has the largest reliability to execute the most tasks with the least cost will be selected and added into the set $\mathcal{S}$ first.

To design an appropriate approximation algorithm, we first define a reliability contribution function $G(\mathcal{S})$. $G(\mathcal{S})$ indicates the current total reliability of winners who process $\mathcal{T}$ under the constraint of $\boldsymbol{\epsilon}$, defined as follows:

$$G(S) = \sum_{j=1}^{l} \min\{\sigma_j^{\mathcal{S}}, \epsilon_j\} \tag{7}$$

Based on $G(S)$, the marginal reliability contribution $v_i(\mathcal{S})$ is the marginal reliability that $w_i \in \mathcal{W} - \mathcal{S}$ can contribute to the whole job, defined as follows:

$$v_i(\mathcal{S}) = G(\mathcal{S} \cup \{w_i\}) - G(\mathcal{S}) \tag{8}$$

Based on Eq. 8, we illustrate the winner selection algorithm in Fig. 9. The algorithm begins from an empty set $\mathcal{S}$. In each iteration, it adds the winner who has the maximum weight $\rho_i = \frac{v_i(\mathcal{S})}{\beta_i}$ into $\mathcal{S}$. The algorithm terminates when $G(\mathcal{S}) = \sum_{j=1}^{l} \epsilon_j$. The computation overhead of the algorithm is $O(n^2 l)$, where n is the number of sellers and l is the number of tasks.

5.3 The Critical Pricing Algorithm of RADT

Pricing(s_i):

1. create a empty winner set $\mathcal{S}'$.
2. record $bid = Bids(s_i)$ and set $Bids(s_i) = 0$. ▷ remove B_i from $\mathcal{B}$
3. repeat until $G(\mathcal{S}') = \sum_{j=1}^{l} \epsilon_j$:
   (a) compute $\rho_k = \frac{v_i(\mathcal{S}')}{\beta_k}$, for $\forall s_k$ is not in $\mathcal{S}'$ and satisfies $Bids(s_k) \neq 0$;
   (b) record the index of the maximum ρ_k as k^*;
   (c) if $\mathcal{P}(s_i) < \frac{\beta_{k^*} v_i(\mathcal{S}')}{v_{k^*}(\mathcal{S}')}$, set $\mathcal{P}(s_i) = \frac{\beta_{k^*} v_i(\mathcal{S}')}{v_{k^*}(\mathcal{S}')}$;
   (d) add s_{k^*} to $\mathcal{S}'$.
4. set $Bids(s_i) = bid$, delete $\mathcal{S}'$.

Fig. 10. The reliability-aware pricing algorithm

The pricing algorithm in Fig. 10 is to determine payments for winners. We consider that each winner is priced at p_i for his winning bid B_i. Let $\mathcal{B}_{-i}$ denote all bids except B_i. Then, we conduct the greedy winner selection over $\mathcal{B}_{-i}$ to get a solution, denoted by $\mathcal{S}'$. We assume that the bid B_k is the winning bid in the k_{th} iteration, where $G(\mathcal{S}')$ is the utility before adding w_k into $\mathcal{S}'$. So the payment of B_i must be no more than $\frac{\beta_k v_i(\mathcal{S}')}{v_k(\mathcal{S}')}$. Otherwise, the weight ρ_i is not the largest. So the critical payment of winner s_i is the maximum critical value:

$$p_i = \max\{\frac{\beta_k\, v_i(\mathcal{S}')}{v_k(\mathcal{S}')} | k = 1, 2, \cdots\} \tag{9}$$

which terminates at $G(\mathcal{S}') = \sum_{j=1}^{l} \epsilon_j$. The total time complexity of pricing is $O(n^3 l)$, where n is the number of sellers and l is the number of tasks.

6 Theoretical Analysis

To prove the truthfulness of our RADT system, we should ensure some properties of the RADT auction. First, we simply define a notation:

$$G(j|\mathcal{S}) = \min\{\sigma_j^{\mathcal{S}}, \epsilon_j\}.$$

Lemma 1. *$G(\mathcal{S})$ is an increasing function.*

Proof. Considering two arbitrary winner sets $\mathcal{S}_1$ and $\mathcal{S}_2$, $\mathcal{S}_1 \subseteq \mathcal{S}_2 \subseteq \mathcal{S}$. According to Eq. 1, we have $\sigma_i^{\mathcal{S}_1} \leq \sigma_i^{\mathcal{S}_2}$ and $G(j|\mathcal{S}_1) \leq G(j|\mathcal{S}_2)$ for $\forall t_j \in \mathcal{T}$. Then $G(\mathcal{S}_1) \leq G(\mathcal{S}_2)$ when $\mathcal{S}_1 \subseteq \mathcal{S}_2$, which implies $G(\mathcal{S})$ is increasing. ■

Theorem 1. *$G(\mathcal{S})$ is submodular.*

Proof. Without loss of generality, we assume that for two arbitrary winner sets $A, B \subseteq \mathcal{S}$. For $\forall t_i \in \mathcal{T}$, we have the conclusion that $\sigma_j^A + \sigma_j^B = \sigma_j^{A \cap B} + \sigma_j^{A \cup B}$ which indicates $\sigma_j^{\mathcal{S}}$ is submodular. Since $G(j|\mathcal{S})$ is the cut-off function of $\sigma_j^{\mathcal{S}}$, we can prove that $G(j|\mathcal{S})$ is submodular according to [8]. Hence, $G(\mathcal{S})$ is submodular because of the fact that $G(\mathcal{S}) = \sum_{j=1}^{l} G(j|\mathcal{S})$. ■

Lemma 2 ***(Bid monotonicity).*** *Each seller w_i who wins by bidding $(\mathcal{T}_i, \beta_i)$ still wins by biding any $\beta_i' < \beta_i$ and any $\mathcal{T}_i' \supset \mathcal{T}_i$ given that other bids are fixed.*

Proof. Let ρ_i, $v_i(\mathcal{S})$ denote the weight and marginal reliability of seller w_i who bids $(\mathcal{T}_i, \beta_i)$, where $\rho_i = \frac{v_i(\mathcal{S})}{\beta_i}$. Let ρ_i', $v_i(\mathcal{S})'$ denote the weight and marginal reliability respectively if w_i bids $(\mathcal{T}_i', \beta_i)$ or $(\mathcal{T}_i, \beta_i')$. Either in $(\mathcal{T}_i', \beta_i)$ or in $(\mathcal{T}_i, \beta_i')$, it is clear that $v_i(\mathcal{S}) \geq v_i(\mathcal{S})'$ and $\rho_i \geq \rho_i'$ because of the submodularity of $G(\mathcal{S})$ according to Theorem 1. Moreover, if w_i has not been selected by bidding $(\mathcal{T}_i, \beta_i)$, he will not be selected by bidding $(\mathcal{T}_i, \beta_i')$ or $(\mathcal{T}_i', \beta_i)$. ■

Lemma 3 ***(Critical payment).*** *Each seller s_i is paid a critical value p_i.*

Proof. We assume s_i wins in the k_{th} iteration, so the set $\mathcal{S}$ of winner selection and $\mathcal{S}'$ of pricing is same from the 0_{th} to the $(k-1)_{th}$ iterations. If s_i reports a bid β_i' instead of β_i. We need to prove that β_i' will fail if $\beta_i' > p_i$, otherwise he still wins when $\beta_i' \leq p_i$. Then we consider these two cases in the k_{th} iteration:

Case 1: $\beta_i' > p_i$. According to Fig. 9, we can derive that $\frac{v_i(\mathcal{S}_{k-1})}{\beta_i'} = \frac{v_i(\mathcal{S}_{k-1}')}{\beta_i'} < \frac{v_i(\mathcal{S}_{k-1}')}{p_i} \leq \frac{v_i(\mathcal{S}_{k-1}')}{\beta_k}$, where s_k is a winner, so that $\mathcal{S}' = \mathcal{S}' \cup \{s_k\}$. The first equation holds because $\mathcal{S}_{k-1} = \mathcal{S}_{k-1}'$. And the last inequation makes sense due to $p_i \geq \frac{\beta_k v_i(\mathcal{S}_{k-1}')}{v_k(\mathcal{S}_{k-1}')}$ according to Eq. 9. Hence, s_k is selected as a winner instead of s_i in the k_{th} iteration. So, $\mathcal{S}_k' = \mathcal{S}_{k-1} \cup \{s_k\} = \mathcal{S}_k$. Based on the above analysis, we can conclude that s_i will fail in all iterations of the winner selection in Fig. 9.

Case 2: $\beta_i' \leq p_i$. Assume that the winner selection runs over $\mathcal{B}_{-i}$ which is the process for pricing s_i. According to Eq. 9, we assume that $p_i = \frac{\beta_k v_i(\mathcal{S}_{k'-1})}{v_k(\mathcal{S}_{k'-1})}$, where s_k is the winner in the k_{th}' iteration. Now we run the winner selection

again with the input set $\mathcal{B}$. We discuss two subcases of this process: (1) s_i wins before the k'_{th} iteration; (2) s_i does not win before the k'_{th} iteration. In the k'_{th} iteration: $\frac{v_i(\mathcal{S}_{k'-1})}{\beta'_i} \geq \frac{v_i(\mathcal{S}_{k'-1})}{p_i} \geq \frac{v_k(\mathcal{S}_{k'-1})}{\beta_k}$. Therefore, s_i wins in this iteration.

In Summary, the payments for all winners are critical. ■

Theorem 2. *The RADT auction is truthful.*

Proof. Lemmas 2 and 3 prove that the winner selection is monotonic and all the payments are critical respectively. So the auction is truthful according to [15]. ■

Theorem 3. *The RADT auction is individually rational.*

Proof. We consider that a seller w_i probably encounters these two situations, $w_i \in \mathcal{S}$ and $w_i \notin \mathcal{S}$. If $w_i \notin \mathcal{S}$, his payment will be zero. Otherwise, he wins the auction and his payment is p_i. According to Lemma 3, w_i will always be paid with the critical value p_i when he bids any $\beta_i < p_i$. Each seller bids his truthful cost due to the truthfulness in Theorem 2. Apparently, $p_i - c_i \geq 0$ holds. ■

Theorem 4. *The RADT system is truthful.*

Proof. We guarantee the truthfulness of RADT from three aspects.

1. **Deposit.** The consumer and sellers are asked for deposits which enforce them not to deviate from RADT, i.e., quit midway.
2. **Truthful auction.** The two-stage bidding strategy in Sect. 4.2 requires sellers to reveal bids truthfully. And the reverse auction can make sellers bid their truthful cost according to Theorem 2.
3. **Modifier.** The consumer must offer rewards to sellers to get keys for decryption, guaranteed by the modifiers in $GetKey()$. Moreover, some time modifiers are used to ensure that each procedure is invoked orderly.

Hence, the whole data trading process of RADT is truthful. ■

7 Implementation and Evaluations

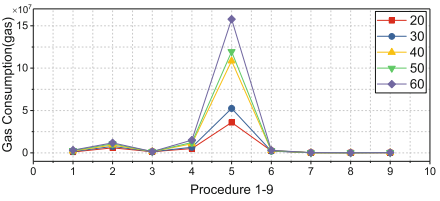

Fig. 11. Gas consumption of each procedure in Ganache Cli (with 20 sellers)

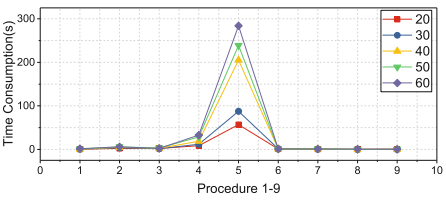

Fig. 12. Time consumption of each procedure in Ganache Cli (with 20 sellers)

We implement a prototype of RADT including the RADToken, the consumers and sellers. RADToken is deployed to a local simulated network TestRPC using Ethereum development tool Ganache Cli which is realized in the programming language Solidity with JavaScript (JS) as the intermediate interactive language. The consumer side is written in Python who needs to complete data decryption. And seller side is written in JS and Python who should finish the bid encryption and data encryption.

Due to difference of SHA-3 between JS and Solidity, we implement a custom SHA-3 in JS to make $en\beta_i$ have the identical value with sha3 $(w_i, \beta_i, nonce_i)$ in Solidity. We employ the standard cryptographic toolkit in Python, where we use AES for symmetric encryption and RSA for asymmetric encryption. Before the evaluation, we set some major parameters of RADT. The number of tasks l varies in [20, 30, 40, 50, 60] while the number of sellers n is fixed at 20. For each seller, his reliability is randomly generated from 0.6 to 1 and his bid is from 10 to 20. The reliability requirements ranges from 1 to 2.

7.1 Evaluations on Simulated Network at System Level

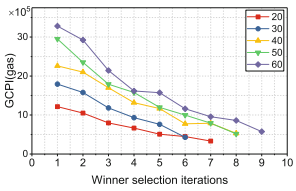

Fig. 13. GCPI vs. iterations

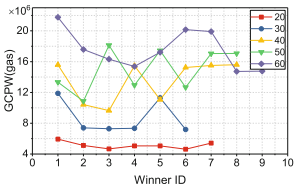

Fig. 14. GCPW vs. ID

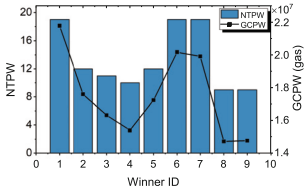

Fig. 15. NTPW vs. ID

To demonstrate the practicality of our system, we first use Ganache Cli to construct a simulated network which is much like the real Ethereum environment except for its automatically mining mechanism in the background. This allows us to focus on the performance of RADToken, irrespective of time-consuming mining process and complex network circumstances in Ethereum. Our RADToken consists of nine main functions which correspond to **Procedures 1–9** that reflect the functionalities of **Initiate**, **CommitBid**, **RevealBid**, **WinnerSelection**, **Pricing**, **SetKey**, **GetKey**, **Refund** and **Payment** in Sect. 3.2.

To evaluate the unique performance of RADToken at the system level, we use two metrics for each procedure: *gas consumption* and *time consumption* which are depicted in Figs. 11 and 12 respectively. Since each computational step will be charged some gas, the more complicated the procedure is, the more gas and time it will consume. The operations to create and write storage data are relatively expensive [20], as we can see, Procedure 2, 4 and 5 use more gas and time. Procedure 4 need execute a nontrivial set of add, subtract, multiply, divide, compare and write operations, and there is a positive correlation between

the number of winners and gas consumption. Procedure 5 is roughly equivalent to execute Procedure 4 $|\mathcal{S}|$ times. On the other hand, we may traverse more iterations than the entire Procedure 4 to find the critical payment for a winner, which uses more gas and time accordingly. Procedure 3 uses much gas because each encrypted bid is a 32-bytes hash value which will take up more storage. Other procedures use less gas due to their most read operations and smaller data length. Notice that here we use gas in **wei** where 1 ether $= 10^{18}$ wei.

7.2 Evaluations of Auction Mechanism

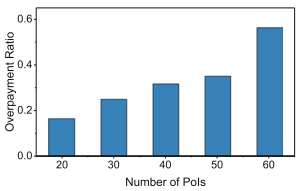

Fig. 16. Overpayment ratio

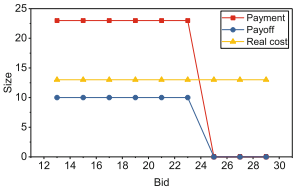

Fig. 17. Payoff of a bid

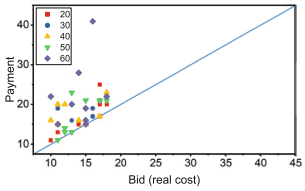

Fig. 18. Payments vs. bids

Since the **auction** including **WinnerSelection** and **Pricing** is the core mechanism of RADToken, we explicitly evaluate the performance of **WinnerSelection** and **Pricing** respectively. To avoid exceeding **gasLimit**, we divide the winner selection and pricing procedures into multiple iterations and repeatedly send a transaction to RADToken to select a winner and price the winner.

WinnerSelection. We give an example in Fig. 13 to compare the *gas consumption per iteration* (GCPI) under different number of tasks from 20 to 60. We notice a gradual decline of GCPI. We explain that by the constant cost of loading past mined blocks from storage into memory before each selection [10].

Pricing. We use *gas consumption per winner* (GCPW) and *number of transactions per winner* (NTPW) as two metrics in Figs. 14 and 15 respectively. We figure out that determining the payment for each winner will consume how much gas and need how many transactions. Figure 14 shows that the total gas consumption increases as the increasing number of tasks from an overall perspective. The GCPW has nothing to do with the iteration sequence which is only related to the number of PoIs and the number of traverse times to obtain its critical payment, which we can see in Fig. 15. The NTPW represents traverse times needed to price a winner and the corresponding GCPW shows that more gas will be used if more transactions are needed when the number of PoIs is 60.

Beyond valuating the performance on blockchain, we should ensure that the properties of our auction mechanism holds. we use the following metrics: overpay ratio, truthfulness and individual rationality. The overpay ratio is defined as:

$$\lambda = (\mathcal{P} - \mathcal{C}(\mathcal{S}))/\mathcal{C}(\mathcal{S}) \tag{10}$$

where $\mathcal{P}$ is the total payment and $\mathcal{C}(\mathcal{S})$ is the total cost. It measures the cost paid by the consumer to induce the truthfulness overall. Ensuring truthfulness means that no sellers can improve his payment by committing a different bid from the real one. Individual rationality ensures that each payoff is non-negative.

Overpayment Ratio: Figure 16 plots the overpayment ratio λ when l changes from 20 to 60. The results show that λ is always less than 0.6, which means that the consumer does not have to pay much extra money to induce truthfulness. λ increases monotonously with increasing l because more sellers will be selected and the increments of the payments are greater than those of the costs.

Truthfulness: To verify the truthfulness, we randomly pick a winner and change its claimed bid, then recalculate the payments as well as the payoffs. The results illustrated in Fig. 17 show that when the truthful bid (real cost) is 13, the payment is 23 and the payoff is 10. The payoff remains unchanged when the bid is no more than 23. However, if the bid is larger than the critical payment 23, the payoff becomes zero which means that the winner loses the auction.

Individual Rationality: We demonstrate individual rationality in Fig. 18. Each payment is greater than the related bid when l varies from 20 to 60.

8 Related Works

We review related works from the following two aspects:

Trading on Blockchain: Blockchain offers users new options for managing their holdings and their trading intentions which can ensure the data integrity. Due to the honest-but-curious property of secure third party, a few works resort to blockchain to build trading systems. [3] implements a decentralized energy trading system using blockchain to address the problem of providing transaction security. [12] proposes AccountTrade for big data trading which can achieve book-keeping ability and accountability against dishonest consumers. In addition to P2P trading, blockchain is fit for the crowd trading. For instance, [14] conceptualizes a blockchain-based decentralized framework named CrowdBC, which does not depend on any central third party to accomplish crowdsourcing process. However, the high storage requirement prevents the wide usage of blockchains on mobile phones. A novel concept, Consensus Unit (CU) [22], organizes different nodes into one unit and lets them to store at least one copy of blockchain data, which can be applied in more application scenarios. Blockchains conduct trading will consume resources, so BLOCKBENCH is designed in [7] to understand the performance of blockchains against data processing workloads.

Incentive Data Trading Mechanism: In order to improve the repetitive use rate of data, [18] designs a DataMart to determine the pricing and consumer-seller matching in distributed fashion which is suited for highly dynamic and heterogeneous market environment and ad-hoc setting. It adopts double auction for pricing to ensure the truthfulness. However this cannot satisfy some consumers' needs who want buy large volumes of data which is not easy to

collect. [24] designs a practical data collection scheme leveraging mobile crowdsensing and proposes VENUS-PRO for profit maximization and VENUS-PAY for payment minimization which is a data procurement auction in Bayesian setting. The above incentive data trading which adopts auction is executed by a third-party which may disclose privacy or data information. A novel distributed agent-based privacy-preserving framework DADP proposed in [19] enables real-time crowd-sourced statistical data publishing with strong privacy protection under an untrusted server. [9] considers the introduction of homomorphic cryptography to allow the auctions to be processed using only encrypted bids. Moreover, it uses the digital signature to ensure that data has not been manipulated in transmission or by a compromised entity in network.

9 Conclusion

In this paper, we first propose a Reverse-Auction-and-blockchain-based crowdsensed Data Trading system. Different from the existing CDT, we use a meticulous designed smart contract to replace a third-party data broker which can ensure the truthfulness of data consumers and data sellers. In order to incentivize more sellers to participate in the crowdsensing data collection, we propose a reverse auction mechanism to prompt sellers to provide high quality sensing data and claim truthful bids. Meanwhile, we protect sellers' bids by leveraging a two-stage bidding strategy which can blame untruthful sellers and ensure the immutability of bids. The confidentiality of data is preserved by the introduction of symmetric and asymmetric cryptography where the keys cannot be grabbed in transmission. Finally, we implement a prototype on an Ethereum test network and the evaluations demonstrate its practicability.

Acknowledgment. This research was supported in part by National Natural Science Foundation of China (NSFC) (Grant No. 61872330, 61572336, 61572457, 61632016, 61379132, U1709217), Natural Science Foundation of Jiangsu Province in China (Grant No. BK20131174, BK2009150), Anhui Initiative in Quantum Information Technologies (Grant No. AHY150300), and Natural Science Research Project of Jiangsu Higher Education Institution (No. 18KJA520010).

References

1. Thingful. https://www.thingful.net/
2. ThingSpeak. https://thingspeak.com/
3. Aitzhan, N.Z., Svetinovic, D.: Security and privacy in decentralized energy trading through multi-signatures, blockchain and anonymous messaging streams. IEEE Trans. Dependable Secure Comput. **15**(5), 840–852 (2018)
4. Aumasson, J.P., Henzen, L., Meier, W., Phan, R.C.W.: SHA-3 proposal BLAKE. Submission to NIST (2008)
5. Buterin, V.: A next-generation smart contract and decentralized application platform. White paper (2014)

6. Cai, C., Zheng, Y., Wang, C.: Leveraging crowdsensed data streams to discover and sell knowledge: a secure and efficient realization. In: IEEE ICDCS (2018)
7. Dinh, T.T.A., Liu, R., Zhang, M., Chen, G., Ooi, B.C., Wang, J.: Untangling blockchain: a data processing view of blockchain systems. IEEE Trans. Knowl. Data Eng. **30**(7), 1366–1385 (2018)
8. Gao, G., Xiao, M., Wu, J., Huang, L., Hu, C.: Truthful incentive mechanism for nondeterministic crowdsensing with vehicles. IEEE Trans. Mob. Comput. **17**(12), 2982–2997 (2018)
9. Gao, W., Yu, W., Liang, F., Hatcher, W.G., Lu, C.: Privacy-preserving auction for big data trading using homomorphic encryption. IEEE Trans. Netw. Sci. Eng. (2018)
10. Hu, S., Cai, C., Wang, Q., Wang, C., Luo, X., Ren, K.: Searching an encrypted cloud meets blockchain: a decentralized, reliable and fair realization. In: IEEE INFOCOM (2018)
11. Jiang, C., Gao, L., Duan, L., Huang, J.: Scalable mobile crowdsensing via peer-to-peer data sharing. IEEE Trans. Mob. Comput. **17**(4), 898–912 (2018)
12. Jung, T., et al.: AccountTrade: accountable protocols for big data trading against dishonest consumers. In: IEEE INFOCOM (2017)
13. Kosba, A., Miller, A., Shi, E., Wen, Z., Papamanthou, C.: Hawk: the blockchain model of cryptography and privacy-preserving smart contracts. In: IEEE S&P (2016)
14. Li, M., et al.: CrowdBC: a blockchain-based decentralized framework for crowdsourcing. IEEE Trans. Parallel Distrib. Syst. (2018)
15. Myerson, R.B.: Optimal auction design. Math. Oper. Res. **6**(1), 58–73 (1981)
16. Nakamoto, S.: Bitcoin: a peer-to-peer electronic cash system. https://bitcoin.org/bitcoin.pdf
17. Niu, C., Zheng, Z., Wu, F., Gao, X., Chen, G.: Trading data in good faith: integrating truthfulness and privacy preservation in data markets. In: ICDE (2017)
18. Susanto, H., Zhang, H., Ho, S., Liu, B.: Effective mobile data trading in secondary ad-hoc market with heterogeneous and dynamic environment. In: IEEE ICDCS (2017)
19. Wang, Z., et al.: Privacy-preserving crowd-sourced statistical data publishing with an untrusted server. IEEE Trans. Mob. Comput. (2018)
20. Wood, G.: Ethereum: a secure decentralised generalised transaction ledger (2014). https://gavwood.com/paper.pdf
21. Xiao, M., Wu, J., Huang, L., Cheng, R., Wang, Y.: Online task assignment for crowdsensing in predictable mobile social networks. IEEE Trans. Mob. Comput. **16**(8), 2306–2320 (2017)
22. Xu, Z., Han, S., Chen, L.: CUB, a consensus unit-based storage scheme for blockchain system. In: ICDE (2018)
23. Yu, J., Cheung, M.H., Huang, J., Poor, H.V.: Mobile data trading: behavioral economics analysis and algorithm design. IEEE J. Sel. Areas Commun. **35**(4), 994–1005 (2017)
24. Zheng, Z., Peng, Y., Wu, F., Tang, S., Chen, G.: Trading data in the crowd: profit-driven data acquisition for mobile crowdsensing. IEEE J. Sel. Areas Commun. **35**(2), 486–501 (2017)