Blockchain-Based Reliable and Privacy-Aware Crowdsourcing with Truth and Fairness Assurance

Wu, Haiqin; Dudder, Boris; Wang, Liangmin; Sun, Shipu; Xue, Guoliang

Published in:
IEEE Internet of Things Journal

DOI:
10.1109/JIOT.2021.3097950

Publication date:
2022

Document version
Peer reviewed version

Document license:
Unspecified

Citation for published version (APA):
Blockchain-based Reliable and Privacy-aware Crowdsourcing with Truth and Fairness Assurance

Haiqin Wu, Member, IEEE, Boris Düdder, Member, IEEE, Liangmin Wang, Member, ACM, Guoliang Xue, Fellow, IEEE, and Shipu Sun

Abstract—The ubiquity of crowdsourcing has reshaped the static-sensor-enabled data sensing paradigm with cost efficiency and flexibility. Still, most existing triangular crowdsourcing systems only work under the centralized trust assumption and suffer from various attacks mounted by malicious users. Although incorporating the emerging blockchain technology into crowdsourcing provides a possibility to mitigate some of the issues, how to concretely implement the crucial components and their functionalities in a verifiable and privacy-aware manner remains unaddressed. In this paper, we construct BRPC, a blockchain-based decentralized system for general crowdsourcing. BRPC integrates the confident-aware truth discovery algorithm to provide task requesters with reliable task truths while evaluating each worker’s data quality. To mitigate incorrect evaluation of malicious requesters, we propose a privacy-aware verification protocol leveraging the threshold Paillier Cryptosystem, with which a certain number of workers can collaboratively verify the evaluation results without knowing any sensory data. Furthermore, we define the user’s three roles and elaborate a comprehensive reputation evaluation model enforced by smart contracts for its trustworthy running. Particularly, financial and social incentives are both offered to motivate users honest involvement. Finally, we implement a prototype of BRPC and deploy it on Ethereum. Theoretical analyses and experiment results show its security and practicality.

Index Terms—Crowdsourcing, blockchain, truth discovery, verifiability, privacy.

1 INTRODUCTION

The past few years have witnessed the rapid development and huge commercial value of crowdsourcing especially in the smartphone proliferation era. As a significant part of sharing economy, crowdsourcing has become a leading paradigm leveraging the crowds, e.g., smartphone users, to solve some tough problems that cannot be addressed by individuals or machines alone, such as large-scale data sensing/collection and more specialized services. After successful completion, the task requesters will obtain what they want. Meanwhile, the crowds (i.e., workers) can earn rewards as compensation for resource consumption. Apparently, crowdsourcing brings a win-win situation for both task requesters and the workers. Due to these remarkable benefits, numerous crowdsourcing applications emerged and gained considerable attention in different fields such as environment monitoring, healthcare assistance [6], transportation, and business.

Traditionally, the crowdsourcing model involves three roles: task requesters, workers, and a centralized crowdsourcing platform. The platform acts as the intermediary between the task requesters and the workers, where the task requesters can post tasks and the workers are able to search available tasks and submit data. Moreover, the crowdsourcing platform also provides worker selection and reward allocation service for participants. Such a centralized crowdsourcing model is easy for system management and utilization of crowdsourcing applications. However, due to its trust-based centralization, many issues cannot be circumvented, including single point of failure, vulnerable to Sybil attacks, DDoS attacks, data privacy disclosure [10], and the typical “false-reporting” and “free-riding” behaviors of malicious users. These weaknesses were also pointed out by Li et. al [18]. Although some cryptography-based privacy preservation methods ([21], [27], [28], [39]) and anonymous reputation management model [32], [34] were proposed to solve part of these problems, they are still limited to the centralized setting and would fail to work once the centralized trust is broken.

Recently, blockchain technology has exhibited promise in constructing a decentralized crowdsourcing model, due to its inherent advantages like decentralization, transparency, and tamper-resistance. Leveraging the emerging blockchain technology, some researches have attempted to build decentralized crowdsourcing systems ([11], [12], [18], [20]), which mostly only provide a general blockchain-assisted framework, with no fine-grained implementation...
of some modules or having not fully addressed some crucial concerns in crowdsourcing, e.g., how to extract truthful knowledge from the unreliable sensory data contributed by diverse workers, how to reliably and privately evaluate the quality of sensory data, how to anonymously and efficiently assess and manage the participant’s reputation in a distributed manner, and how to ensure the fairness between the workers and the task requesters, e.g., fair reward allocation. With these in mind, a series of researches like blockchain-based quality and reputation evaluation ([1], [2], [9], [42], [43], [46]), truth discovery/selling ([3], [4], [26]), truthful incentive mechanism ([13], [30]), anonymous/privacy-aware decentralized crowdsourcing systems ([22], [23], [37], [38], [40]) have been investigated to address some of the above key issues. Despite their nontrivial contributions, these prior solutions still suffer from the following limitations.

First, those relying on miners or workers to evaluate/verify data quality [2], [14], [30] are at the expense of data privacy, as all sensory data is exposed to miners/workers. Those using smart contracts for quality evaluation or verification are either limited to simple data integrity evaluation [18] or prone to tolerate costly cryptographic tools like zero-knowledge proof ([22], [23]). On-chain computations are prohibitively expensive especially when cryptographic operations such as modular exponentiation/multiplication are included. Moreover, solutions assigning evaluation to the task requester either fail to consider the malicious behaviors of requesters [43] or the verification method has technical flaws [42]. Second, the above resolutions mostly adopt simple methods for data quality evaluation and lack quantified fine-grained evaluations, e.g., using majority voting or averaging to decide the truthful information, and regard the same data as truth and in good quality. In real applications, it is far from sufficient and reliable to capture the quality difference in sensory data contributed by multiple workers, especially for the numeric sensory data besides the categorical data. Hence, there is a great need for a method to discover the truths of sensing tasks and evaluate the worker-contributed data quality based on the inferred truths. Third, existing blockchain-based reputation assessment and management are mainly based on the user’s behaviors as a worker, whereas fail to consider user’s other roles such as task requester or verifier. Additionally, the reputation is updated as long as a task is completed, which would inevitably bring considerable burden for blockchain with increasing tasks. Essentially, it is not necessary to perform frequent updates since a user usually exhibits similar behaviors during a short period of time. In other words, there is no remarkable change in the reputation after he completes a task. Last but not least, most incentive mechanisms adopted quality-aware reward strategies. Their coarse-grained and inaccurate quality evaluation methods make it unfair for workers who make efforts and contributions to a different extent. Fine-grained quality evaluation and reward allocation are both required to achieve true fairness. Essentially, considering the variety of malicious behaviors, it is more complicated to fairly and correctly evaluate the data quality and user reputation in the blockchain-based crowdsourcing system, especially when there is no ground truth for the data quality evaluation. To the best of our knowledge, there is no work addressing these limitations thoroughly, still leaving it an open and tough challenge.

Motivated by this, in this paper, we leverage the Blockchain technology to build a Reliable, Privacy-aware, and fair decentralized Crowdsourcing system, called BRPC. Apart from the generic blockchain-based crowdsourcing framework, we focus on elaborating how data quality and user reputation can be evaluated in a distributed, private, verifiable, and efficient manner. On this basis, we provide an incentive strategy to reward or penalize users, which achieves both financial and social fairness between the requesters and the workers. The main contributions of our work are summarized as follows.

- We construct a general decentralized crowdsourcing system, named BRPC, via blockchain. In contrast to existing blockchain-based frameworks, BRPC can efficiently suit more realistic crowdsourcing scenarios where each task requester publishes multiple tasks and each worker taking multiple tasks in different time slots. Additionally, BRPC achieves system reliability and security, user privacy, accurate truths, and financial-social fairness of all users.
- Based on BRPC, we employ state-of-the-art confident-aware truth discovery algorithm to estimate the task truths meanwhile evaluating the data quality of each worker over the involved tasks. Task requesters are entitled to evaluate the reliability of the workers performing multiple tasks with the inferred truths. Meanwhile, leveraging the threshold Paillier Cryptosystem, a privacy-aware computation verification protocol is devised to prevent malicious requesters from reporting incorrect evaluations, only requiring cooperation among a set of participating workers with no reveal of any sensory data. Furthermore, we propose a quality-aware rewards allocation strategy to motivate workers’ high-quality data contribution and active participation in the verification procedure, which achieves financial fairness among the users. Particularly, we comprehensively capture user’s different roles and formulate a novel reputation evaluation model for BRPC. The reputation evaluation is enforced by smart contracts to achieve reliability and social fairness among the users.
- We implement a prototype of our BRPC and evaluate it’s performance over Ethereum. The experiment results demonstrate its validity and effectiveness with an affordable overhead.

The remainder of this paper is organized as follows. In Section 2, we introduce the background knowledge. Section 3 states the research problem. Section 4 describes our proposed BRPC system in detail. The security analysis and experimental evaluations are presented in Section 5 and Section 6, respectively. Section 7 reviews some related work. Finally, we conclude this paper in Section 8.

2 Background

Blockchain & smart contract. The emergence of cryptocurrencies has unveiled the underlying blockchain technology. Conceptually, a blockchain is essentially a replicated,
immutable, and distributed ledger maintained by a P2P network with a specific consensus mechanism. Each block aggregates a sequence of transactions and is chained to the previous block via a cryptographic hash function. The salient characteristics of blockchain include decentralization and anonymity, transparency and immutability, and distributed consensus. There are some well known consensus protocols such as Proof of Work (PoW), Proof of Stake (PoS), and Practical Byzantine Fault Tolerance (PBFT) [5].

Known as blockchain 2.0, a smart contract is a kind of self-executing and self-verified digital contract programmed on the blockchain securely, which can be verified by blockchain nodes. In the Ethereum blockchain, smart contract is regarded as a special account with associated codes. Once a smart contract is deployed, users can interact with the contract through the contract address and the Application Binary Interface (ABI). Moreover, it can be triggered by certain events to carry out predefined functions in the smart contract. In this paper, we design and implement two smart contracts: user smart contract (USC) and task smart contract (TSC) for our BRPC system (to be detailed in Section 4.2).

**Confidence-aware truth discovery.** In crowdsourcing applications, truth discovery algorithms effectively solve conflicts among the sensory data submitted by multiple workers, and provide truthful information about sensing tasks. Unlike some simple methods such as taking the majority or the average as the “truth”, truth discovery innovatively captures the different reliability levels of workers, which works by iteratively estimating the truths via weighted aggregation over sensory data among workers and updating the reliability degrees of different workers in the form of weights, until some convergence criterion is reached.

In this paper, aiming at the generic and real-world crowdsourcing scenarios, we resort to the typical confidence-aware truth discovery algorithm (CATD) [19] which exhibits better accuracy performance and considers the ubiquitous long-tail phenomenon in crowdsourcing tasks, i.e., for resource constraints, most workers take a few tasks, and only a few workers take plenty of tasks. The rationale behind CATD lies in that if a worker is more reliable, the probability that he gives trustworthy information is higher, and the worker who provides trustworthy information is more reliable. Additionally, if a worker takes more tasks, it is more likely that the worker weight estimation is closer to his true reliability degree, i.e., the confidence of weight estimation is higher. In contrast, if a user only takes a few tasks, the weight estimation confidence is low.

W.l.o.g., let \( T \) and \( U \) be the set of sensing tasks and workers, respectively. The sensory data submitted by a worker \( U_i \in U \) for a task \( T_j \in T \) is represented by \( x_{ji} \), and the estimated truth for task \( T_j \) is \( x_j^* \). Next, we describe the detailed iterative process in CATD, which mainly consists of two sub-steps: truth estimation and weight estimation.

1) Truth estimation: Given all workers’ sensory data and the estimated truth \( x_j^* \), the weight for each worker \( U_i \) is estimated as follows.

\[
\omega_i = \frac{\chi^2_{\alpha/2,|T_{U_i}|} \sum_{\tau_j \in T_{U_i}} (x_j^* - x_j)^2}{\sum_{\tau_j \in T_{U_i}} \omega_i},
\]

where \( \chi^2_{\alpha/2,|T_{U_i}|} \) is a coefficient used to adjust the worker’s weight, and can be directly computed regardless of the sensory data. Specifically, \( \chi \) denotes the Chi-squared distribution and \( \alpha \) is the significance level (a small constant such as 0.05). Moreover, \( T_{U_i} \) represents the set of tasks chosen by worker \( U_i \) in the current time slot.

The aforementioned iterative procedure does not terminate until some predefined criterion is met, such as reaching a maximum number of iterations or the truths difference between two adjacent iterations is less than a preset threshold. In this paper, we preset a maximum iteration \( I_{\text{max}} \) as adopted in most related work. Note that, in our scheme, we use CATD algorithm as a building block as it provides a fine-grained and more reliable data quality evaluation and truth finding solution supporting both numerical and categorical data, based on the workers’ participation behaviors in a time slot. Since a worker’s weight indicates the reliability of data he submitted for a set of tasks, we refer to it to quantify the data quality, and use worker weight and data quality (corresponds to this worker) interchangeably in the latter concrete scheme description (see Section 4.2).

2) weight estimation: Given all workers’ sensory data and the estimated truth \( x_j^* \), the weight for each worker \( U_i \) is estimated as follows.

\[
\omega_i = \frac{\chi^2_{\alpha/2,|T_{U_i}|} \sum_{\tau_j \in T_{U_i}} (x_j^* - x_j)^2}{\sum_{\tau_j \in T_{U_i}} \omega_i},
\]

where \( \chi^2_{\alpha/2,|T_{U_i}|} \) is a coefficient used to adjust the worker’s weight, and can be directly computed regardless of the sensory data. Specifically, \( \chi \) denotes the Chi-squared distribution and \( \alpha \) is the significance level (a small constant such as 0.05). Moreover, \( T_{U_i} \) represents the set of tasks chosen by worker \( U_i \) in the current time slot.

The aforementioned iterative procedure does not terminate until some predefined criterion is met, such as reaching a maximum number of iterations or the truths difference between two adjacent iterations is less than a preset threshold. In this paper, we preset a maximum iteration \( I_{\text{max}} \) as adopted in most related work. Note that, in our scheme, we use CATD algorithm as a building block as it provides a fine-grained and more reliable data quality evaluation and truth finding solution supporting both numerical and categorical data, based on the workers’ participation behaviors in a time slot. Since a worker’s weight indicates the reliability of data he submitted for a set of tasks, we refer to it to quantify the data quality, and use worker weight and data quality (corresponds to this worker) interchangeably in the latter concrete scheme description (see Section 4.2).

**Threshold variant of Paillier Cryptosystem.** In [25], Paillier proposed a new probabilistic asymmetric encryption scheme for public key cryptography. The scheme is known as its attractive additive homomorphic property. Given a public key \( pk = (g, n) \) (where \( g \) is a random integer selected from \( Z_{n^2}^* \) ) and ciphertexts of two messages \( m_1, m_2 \in Z_n \), we are able to derive the ciphertext of \( m_1 + m_2 \). Concretely, the homomorphic properties are described as follows.

\[
E_{pk}(m_1 + m_2) = E_{pk}(m_1) \cdot E_{pk}(m_2),
\]

\[
E_{pk}(a \cdot m_1) = E_{pk}(m_1)^a,
\]

where \( E_{pk}(m_1) = g^{m_1} r_1^a \mod n^2, r_1 \in Z_{n^2} \) and \( a \) is constant from set \( N \).

In this paper, we use the threshold variant of the Paillier Cryptosystem to cater for our scenario requirements, with which the ciphertext of worker weight can be jointly decrypted by a sufficient number of workers (elaborated in Section 4.2.4). With this idea, the correctness of the quality evaluation can be ensured by the joint cooperation among some workers. Take \( (t, N) \)-threshold Paillier Cryptosystem as an example, the private key \( sk \) is first divided into \( N \) pieces, denoted as \( sk^1, sk^2, \ldots, sk^N \), and these pieces are distributed to \( N \) entities. Each entity only holds one of the \( N \) pieces (i.e., holds one partial private key) and at least \( t \) pieces held by \( t \) entities are required to collaboratively recover the plaintext encrypted by \( pk \). Specifically, the decryption process contains two steps: share decryption at each entity and share combination for recovering plaintext. For the former, the \( i \)-th entity computes the partially decrypted ciphertext \( c_i = e^{2\Delta sk^i} \), where \( e \) is the ciphertext and \( \Delta = N! \). For the latter, due to space limitations, we refer the reader to [7] for how to recover the plaintext by combining \( \{c_i\}_{i=1}^{N}, t \leq t' \leq N \).
3 Problem statement

In this section, we provide a formal definition of our problem. Before that, we first illustrate the system model of our BRPC system, and then describe the potential security threats and the corresponding security goals.

3.1 System model

As illustrated in Fig. 1, our BRPC system consists of four kinds of roles: task requesters, workers, blockchain nodes, and a decentralized database. The detailed description is as follows.

![System Architecture](image)

**Task requesters**, identified by $\mathcal{R} = \{R_1, R_2, \ldots, R_M\}$, are task owners who create tasks, recruit a certain number of workers, and give workers rewards according to their contributed data quality. To efficiently find the qualified workers and stimulate worker’s participation, each task with a task identifier contains a task specification, including task time/location, task category, sensing object, data evaluation algorithm, worker’s minimum reputation, number of workers, and task budget. Note that, considering the scalability bottleneck of blockchain, as in prior researches [2], [18], [22], [23], we choose to store the task details in the decentralized storage, and only record the task hash on the blockchain. Moreover, they may be curious about the sensory data of users and contributed sensory data. Besides, task requesters can act as miners if they join in the mining work. In this paper, we consider a public blockchain like Ethereum.

**Decentralized database** is a decentralized storage system which relies on service peers to provide storage or computation services by renting out their resources, e.g., consisting of distributed nodes in IPFS network.

In BRPC system, w.l.o.g., we assume that time is divided into time slots $T_1$, $T_2$, $\ldots$, and there are $C$ task categories, denoted by $\mathcal{C} = \{1, 2, \ldots, C\}$, with different difficulty degrees (indicated by different weights in reputation evaluation). In a time slot $T_i$, each task requester $R_i \in \mathcal{R}$ posts a set of tasks $\mathcal{T}_{R_i} = \{\tau_1, \tau_2, \ldots, \tau_m\}$ with task category $c \in \mathcal{C}$ and each task $\tau_j \in \mathcal{T}_{R_i}$ specifies a budget $B_j$ to collect data from $n_j$ workers whose global reputation and task expertise reputation values are no lower than $\varepsilon^1_j$ and $\varepsilon^2_j$, respectively. For better illustration, we assume that $R_i$ tends to post same category of tasks in each time slot $T_i$, which indicates that workers have the similar expertise/reliability in completing tasks posted by the same requester within a period of time. The situation where requesters posts tasks with distinct types in the same time slot can be easily addressed by grouping the same type of tasks and performing weighted aggregation on each group data. Besides, the fine-grained truth discovery algorithm [24] can be also employed and we will leave the detailed implementation for our future work.

Let $\mathcal{U}_i$ be the set of users performing $R_i$’s tasks in time slot $T_i$. We denote the subset of tasks chosen by worker $U_j \in \mathcal{U}_i$ as $\mathcal{T}_{U_i}^j$, where $\mathcal{T}_{U_j} \subset \mathcal{T}_{R_i}$. Each worker $U_j \in \mathcal{U}_i$ submits the encrypted sensory data (numerical or categorical) and $R_i$ obtains the original data after decryption. Due to the diverse reliability of workers and the uncertainty of task truths, our first issue is to let $R_i$ detect the most trustworthy information from multiple sensory data and evaluate the reliability of each worker in completing this kind of tasks, i.e., data quality. Considering the possibly unreliable evaluation from malicious requesters, the next issue is how to provide a privacy-aware verifiable scheme to ensure the evaluation correctness and the fairness of quality-based incentives. Last, since each user may act in multiple roles in different phases of BRPC, it is a crucial problem to comprehensively model these roles and develop a reliable decentralized reputation assessment method.

3.2 Threat model

In this paper, the security threats are mainly from the internal attackers who may be curious about the real identities of users and contributed sensory data. Besides, task requesters and workers may behave in a malicious manner and attempt to cheat on the other side due to their self-interests.

Specifically, task requesters may be curious about the true identity of the participating workers, and vice versa. Moreover, they may be curious about the sensory data of tasks submitted by others. Similarly, the blockchain nodes...
may want to know the sensory data of specific tasks or data submitted by specific workers.

More seriously, task requesters and workers may behave maliciously to maximize their profits. On one hand, malicious task requesters may want to collect sensory data without giving payments to workers, or giving unfair rewards to workers. For example, for two workers submitting data with the same quality, a malicious requester may give an unknown worker lower rewards while giving an acquainted worker higher rewards. Additionally, they may misreport high-quality data as low-quality data, or even repudiate the fact of obtaining the sensory data (false-reporting). On the other hand, malicious workers may attempt to earn rewards with no/less effort (free-riding). They may forge data or submit low-quality data. Moreover, they may create multiple fake identities (Sybil attack) to request tasks to earn more rewards, or purposely do not submit data on time after choosing many tasks (DDoS), which results in the insufficient sensory data and discourages requesters’ participation. Additionally, some malicious workers may collusively submit low-quality data. Since each participating worker is assigned a piece of the requester’s secret key and is also involved in verifying the correctness of requester’s data quality evaluation, malicious workers may collude with each other to reveal the private key of the task requester or misreport the verification results.

In this paper, We assume that tasks submitted by a task requester are executed by at least \( t \) workers, in which there are fewer than \( t \) colluding workers. These are the basic security requirements in \((t, N)\)-threshold Paillier Cryptosystem. In addition, we assume that task requesters and workers have limited money and the amount of deposit is more than the rewards. For blockchain security, we assume that the majority of blockchain nodes are honest, indicating that attackers do not have the power to control the blockchain.

### 3.3 Design goals

Considering the aforementioned security threats, we elicit the following security requirements for our privacy-aware, reliable, and decentralized crowdsourcing system.

1) **Privacy protection.** For identity privacy, this property requires that the real identities of task requesters and workers are not revealed to any other system entities. For data privacy, we aim to protect the confidentiality of sensory data. Only the task requester is able to obtain the real sensory data and the estimated ground truths of his posted tasks.

2) **Reliability.** From the system perspective, intuitively, single point of failure should be mitigated. From the user’s perspective, on one hand, we should guarantee that the task requesters are able to obtain reliable truths for their tasks (in case of unreliable workers). On the other hand, in case of malicious requesters who may deliberately misreport the data quality, we need to offer a verifiable computation solution for requester’s data quality evaluation to validate the correctness. Moreover, for the reputation evaluation of users, we should ensure that the whole evaluation process is public and the correctness of reputation evaluation is publicly verifiable.

3) **Fairness.** We consider two kinds of fairness: transaction (i.e., financial) fairness and evaluation (i.e., social) fairness. For the former, on one hand, task requesters should be able to obtain the sensory data and get their deposit back only if the corresponding rewards are given to the workers and it is proved that their data quality evaluation is correct. On the other hand, the rewards earned by the workers should depend on the submitted data quality (higher data quality means more rewards) and the workers can get their deposit back only if they follow the designated protocol such as submitting data on time. For the evaluation fairness, the evaluation methods of data quality and user reputation are disclosed to the public. All joined workers can verify the data quality in their joined tasks. Moreover, the trust evaluation should comprehensively consider user’s different roles, and anyone can verify the evaluation results.

We list the notations used throughout this paper in Table 1.

<table>
<thead>
<tr>
<th>Notations</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( R, U )</td>
<td>Set of task requesters and workers</td>
</tr>
<tr>
<td>( T_i )</td>
<td>Set of tasks posted by ( R_i ) in ( T )</td>
</tr>
<tr>
<td>( C )</td>
<td>Set of task categories</td>
</tr>
<tr>
<td>( T_1, T_2, \ldots )</td>
<td>Time slots</td>
</tr>
<tr>
<td>( PK_i, SK_i, A_i )</td>
<td>Key pair and account address of ( U_i/R_i )</td>
</tr>
<tr>
<td>( \epsilon_i, \tau_i )</td>
<td>Encryption/decryption key pair of ( U_i/R_i )</td>
</tr>
<tr>
<td>( E(), \mathcal{D}() )</td>
<td>Encryption and decryption algorithm</td>
</tr>
<tr>
<td>( U_i^+ )</td>
<td>Set of workers executing ( R_i )'s tasks in ( T )</td>
</tr>
<tr>
<td>( U_i^- )</td>
<td>Set of workers executing task ( \tau_j )</td>
</tr>
<tr>
<td>( B_i )</td>
<td>Budget of task ( \tau_j )</td>
</tr>
<tr>
<td>( D_i )</td>
<td>Deposit provided by ( U_i/R_i ) for task ( \tau_j )</td>
</tr>
<tr>
<td>( n_j )</td>
<td>Number of required workers in task ( \tau_j )</td>
</tr>
<tr>
<td>( c )</td>
<td>Weight of task category ( c )</td>
</tr>
<tr>
<td>( s_k )</td>
<td>The ( k )-th private key share of ( U_i/R_i )</td>
</tr>
<tr>
<td>( \omega_{ij} )</td>
<td>Category of tasks posted by ( R_i ) in the current slot</td>
</tr>
<tr>
<td>( \gamma_i )</td>
<td>Global reputation of user ( i ) in ( T )</td>
</tr>
<tr>
<td>( \gamma_{ij} )</td>
<td>Expertise reputation of user ( i ) on ( c ) in ( T )</td>
</tr>
<tr>
<td>( x_{ij}^+, x_{ij}^- )</td>
<td>Sensory data from ( U_i ) and estimated truth of ( \tau_j )</td>
</tr>
</tbody>
</table>

4 **Blockchain-based reliable and privacy-aware crowdsourcing protocol**

In this section, we formalize the BRPC system and show how it works to address the reliability, privacy, and fairness challenges in existing crowdsourcing systems, with a guarantee of deriving the task truths and achieving fair interactions. First, we present an overview of our BRPC system. Next, we focus on illustrating the concrete workflow of BRPC and the corresponding protocols among involved parties. Specifically, we will provide an accurate and efficient method for data quality evaluation in a set of tasks. Moreover, the solution of workers’ joint quality verification is to detect the requester’s possibly unreliable evaluation without compromising data privacy. Based on this, we will quantify the rewards and reputation allocated to each user.
from a holistic perspective so that both financial and social fairness among the users are truly fulfilled.

4.1 Overview

At a high level, the entire crowdsourcing process of our BRPC system consists of five phases, as shown in Fig 2. In Phase 1, the system parameters are initialized and each user (worker or task requester) needs to be registered in the system for later participation (users’ information is sent to the blockchain, see Section 4.2.1). In the task publishing phase (Phase 2), any task requester with task requirements is able to post tasks to the BRPC system via a smart contract in the form of transactions. Each task specifies task details including worker selection rules and data quality evaluation method as mentioned in Section 3.1. Moreover, we define a set of task states and each task is initiated with a “Pending” state and then switches into an “Available” state after being successfully recorded on the blockchain. Along with the tasks, task requesters also need to commit budget and deposit to the contract address for rewarding workers/verifiers and in case of malicious behaviors, respectively. With the on-chain task state information, workers can query available/valid tasks and are eligible to participate in the tasks if their reputation values satisfy a qualification check function (Phase 3). Similarly, some money from workers is locked as deposit once the task execution is confirmed (Section 4.2.2).

To enable the following privacy-aware verification on the worker side, the task requester is responsible for distributing his private key shares to the participating workers before data submission (Section 4.2.3). Once all encrypted sensory data is submitted, the task requester will turn to the CATD algorithm to infer the task truths and evaluate the data quality of each worker after decryption in Phase 4. Meanwhile, workers are incentivized to cooperatively verify the correctness of the evaluation results. As long as $t$ workers are involved in which the majority of the workers are honest, the verification can proceed with correctness assurance. For remuneration, each worker is first rewarded according to the quality of contributed data. Moreover, workers and task requesters will earn extra rewards or lose the deposit, depending on their benign/malicious behaviors (Section 4.2.4). Finally, in Phase 5, BRPC uses the smart contract to enforce reliable reputation evaluation and update, based on a holistic reputation evaluation model which includes users’ different roles and the time implication. The global and expertise reputation values of users are finally recorded on the blockchain (Section 4.2.5). Note that the on-chain user reputation is not updated after each task but only after the completion of a set of tasks, so as to mitigate the storage cost of blockchain to some extent.

4.2 Protocol details

Now we are ready to present the decentralized crowdsourcing protocols in BRPC. Since our protocol design is mainly based on the open blockchain, in the following description, we omit some basic and detailed processes of the underlying blockchain infrastructure such as signing/verifying a transaction and paying the transaction fees. Specifically, our protocol details are illustrated in Fig. 2.

4.2.1 System initialization

This phase consists of two steps: setup and user registration.

In the setup step, we assume that parameters such as the duration of each time slot, $C$ and the corresponding weight of each category $c \in C$ are set by a bootstrap program in our BRPC system. Moreover, the system also sets the user’s initial global reputation and expertise reputation regarding different task categories, based on the average reputation level of humans. To depict the task process, a set of task states $S = \{\text{Pending}, \text{Available}, \text{Unavailable}, \text{QEvaluating}, \text{Completed}, \text{Canceled}\}$ is also predefined in a task smart contract (TSC). Once a task is posted by the task requester, it is initiated with Pending and is waiting for confirmation of miners. Workers can query the current task state whenever they want to participate in some tasks. If it is available, i.e., tasks are recorded on the blockchain, workers have a chance to receive their interested tasks if their global and expertise reputation values meet the task requirements. The state of a task switches into Unavailable when there are sufficient workers. QEvaluating state dominates once the CATD algorithm and worker-side verification start, after which, the budget is rewarded to the workers and the deposit is returned to the honest parties. Finally, the task is completed.

In the whole process, if the task requester cancels the task or the task is expired, the task state would be Canceled. It is worth noting that reputation evaluation is not included in the task state as it is conducted when tasks are completed.

In the registration step, both task requesters and workers register in our BRPC system as users. Specifically, each user $i$ will obtain a key pair $PK_i, SK_i$ and a public address $A_i$, i.e., pseudonym. The initial global and expertise reputation values of $i$ are $\gamma^g_{i,1}$ and $\gamma^e_{i,1}, c \in C$. When $i$ registers, a user smart contract (USC) is invoked which records $A_i$ and his corresponding information including global and expertise reputation values. This information will be updated at the end of each time slot.

4.2.2 Task publishing

In this phase, each $R_i \in R$ posts a series of tasks $T^{R_i}_T = \{\tau_1, \tau_2, \ldots, \tau_m\}$ to BRPC in $T_j$ via transactions. Specifically,
each \( \tau_j \in T_{R_i} \), \( j \in [1, m] \) contains the following information:

\[
\tau_j = \{ T_{id}, c_i, A_i, S_{deadline}, P_{deadline}, T_{deadline}, pk_j, \\
\varepsilon_j, l_j, n_j, B_j + D_{R_i}, \text{checkQ}, \text{QEvaluate} \},
\]

where \( T_{id} \) denotes the task identifier, i.e., the hash of task details. \( c_i \) denotes the category of current tasks posted by \( R_i \). Moreover, \( S_{deadline} \) is the submission deadline before which sensory data should be submitted. \( P_{deadline} \) is the proving deadline, i.e., the time before which the correctness of \( R_i \)'s data quality evaluation should be verified. \( T_{deadline} \) is the task deadline before which task \( \tau_j \) should be completed. \( pk_j \) is the public key for data encryption and can be updated with \( T_i \). The deposit \( D_{R_i} \), committed by \( R_i \) for task \( \tau_j \) consists of two parts: the first part is used to prevent false-reporting attacks and the second part is used to reward workers (we term them verifiers) who participate in the correctness verification process and submit correct verification results. If the evaluation of \( R_i \) is proved correct before \( P_{deadline} \), \( B_j \) and the second part deposit are automatically assigned to the workers and verifiers, respectively. After \( P_{deadline} \), the first part deposit is refunded to \( R_i \). Otherwise, the second part deposit would be assigned to the corresponding workers (see Section 4.2.4). Function checkQ: \( \varepsilon \times \gamma \rightarrow \text{bool} \) checks if the worker is qualified to perform task \( \tau_j \) based on his reputation, and QEvaluate: \( \omega \times x \rightarrow x^\omega \) specifies the CATD algorithm. Note that, the TSC is updated when new tasks are posted and some functions are performed.

### 4.2.4 Data quality evaluation and reward allocation

With all sensory data \( \{ x_j^k \}_{k=1}^{T\ell} \) submitted by each \( U_k \in U \) for task \( \tau_j \in T_{R_i} \), \( R_i \) performs the CATD algorithm and iteratively updates the weight of each worker as well as the estimated task truths until \( I_{max} \) is satisfied. After that, \( R_i \) derives the estimated truth \( x_j^\ast \) for each task \( \tau_j \in T_{R_i} \) and the estimated weight/reliability \( \omega_j \) of each worker \( U_k \in U \) in time slot \( T_i \) according to Eqs. (1) and (2). Moreover, \( \omega_k \) is broadcasted to the network.

#### Fig. 3. An example of workers’ joint verification.

Fig. 3 illustrates an example of workers’ joint verification of requester-side data quality evaluation. After obtaining the weights broadcasted in Step 1, each worker \( U_k \in U \) finds the encrypted sensory data from the distributed database. Being aware of \( \{ E_{pk_k}(x_j^k) \}_{k=1}^{U \ell} \), the workers can jointly run Algorithm 1 to perform privacy-aware verification (Step 2 in Fig. 3), in which the each derived ciphertext \( E_{pk_k}(d_j^k) \) (Line 12 in Algorithm 1) is partially decrypted using each key share and the corresponding result is sent to the other participating workers. For example, as depicted in Fig. 3, \( U_1 \) sends the partially decrypted ciphertext \( C_{i_1} \) to the other workers and meanwhile he also receives information from the others.

Specifically, before iterations, each worker’s weight is initialized with his latest expertise reputation with respect to task category \( c_i \) (Line 2, set it as \( \gamma_i \cdot 1 \)). Next, the estimated truth and weight updates are iteratively performed on encrypted data for \( I_{max} \) times. Specifically, the encrypted estimated truth for task \( \tau_j \) is derived as \( E_{pk_i}(\tilde{x}_j) \) where \( \tilde{x}_j = L \cdot x_j^\ast \) and \( L \) is a scaling factor to tackle fractional data (Line 8). In Section 6, we will show that the accuracy of our result is not compromised if a proper \( L \) is chosen. In the following weight update phase, the encrypted difference between \( L \cdot x_j^\ast \) and \( \tilde{x}_j \) is first calculated,
denoted by $E_{pk_i}(d^k_{ij})$ (Line 11). Since Paillier Cryptosystem does not have multiplicative homomorphic property, it is hard to derive $E_{pk_i}((d^k_{ij})^2)$ by a single worker based on $E_{pk_i}(d^k_{ij})$. Instead, we turn to let each worker $U_k$ decrypt $E_{pk_i}(d^k_{ij})$ using his partial private key $sk^k_i$, and the partial decrypted ciphertext $C_k$ is broadcast to other workers in $U_i^j$. Leveraging $(t,|U_i^j|)$-threshold Paillier Cryptosystem, $U_k$ can recover $d_{ij}$ and further computes each worker’s weight $w^k_{ij}$ (Line 15), which will be utilized as input of the next iteration. After reaching $I_{\text{max}}, U_k$ obtains the final weights and compares them with $w_k$ (broadcasted by $R_i$), if they are equivalent, the verification result $f_k$ is set to be 1, otherwise, it is set to be 0 (Lines 16-20). Before $P_{\text{deadline}}$, $U_k \in U_i^j$ signs $f_j, j \in [1, |U_i^j|]$ using $SK_k$. Concretely, if $f_j = 1$, $U_k$ broadcasts $(A_k, A_j, 1, \text{Sig}_{SK_k}(h(1||A_i||A_j)))$. Otherwise, $(A_k, A_j, 0, \text{Sig}_{SK_k}(h(0||A_i||A_j||w^k_{ij}), w^k_{ij})$ is broadcasted which indicates that $R_i$ incorrectly evaluates the data quality of $U_j$ (Step 3 in Fig. 3). Moreover, the correct estimated weight of $U_j$ (i.e., $w^\tau_{ij}$) is included. All blockchain nodes are able to verify the integrity of the verification messages and miners can upload the correct weight to the blockchain. Although some workers may collude with each other and report incorrect verification results either to conceal $R_i$’s misbehavior or slander honest task requesters, such misbehaviors can be easily identified since we assume that the majority of the verification workers are honest. Solutions dealing with massive malicious workers are out of the scope of this paper and will be investigated in our future work.

When $P_{\text{deadline}}$ arrives, if all signatures are valid and there are no zeros with respect to $R_k$, each task budget $B_{j}, j \in [1, m]$ is redeemed and the rewards assigned to each worker $U_k \in U_i^j$ for each task $\tau_j \in T_k^i$ is:

$$\xi_k^i = \begin{cases} \frac{w_k}{\sum_{U_k \in U_i^j} w_k} \cdot B_j & \tau_j \in T_k^i, \\ 0 & \tau_j \notin T_k^i. \end{cases}$$

From Eq. (6), we can observe that if $U_k$ contributes data to task $\tau_j$, he would earn rewards proportionally to his data quality $w_k$. Otherwise, nothing is rewarded. Therefore, the total reward given by $R_i$ to $U_k$ is $\sum_{j=1}^{m} \xi_k^i$.

Moreover, the first part deposit of $D_R^i$ is returned to $R_i$ while another part is evenly distributed among the verifiers as rewards. In this case, the number of correct quality evaluation is $|U_i^j|$ for $R_k$, which is regarded as a reputation evaluation factor (elaborated in Section 4.2.5). In contrast, if there exist messages containing zero regarding $R_i$, the rewards allocation is based on the correct evaluation results broadcasted by most verifiers. Once $R_i$ gives the wrong quality evaluation, $D_R^i$ will not be refunded and to be used as verification rewards. In addition, as a task requester, the reputation of $R_i$ would be affected. Similarly, for the verifiers, if they report incorrect verification results, they would get no rewards. Moreover, their reputation would degrade as a verifier.

### 4.2.5 Reputation evaluation and update

Unlike prior work which evaluates users from a single perspective, we take different roles of users into consideration and assign different weights to aggregate a user’s global reputation. Moreover, we also examine the reliability of a user as a worker executing tasks with different categories and build an expertise reputation model. W.o.l.g., in $T_k$, we assume that as a worker, $U_k$ receives tasks posted by a set of task requesters $\mathcal{R}^i \subseteq \mathcal{R}$. Meanwhile, as a task requester, $U_k$ also publishes a set of tasks that are assigned to $\eta$ workers.

During the data quality evaluation phase, let $w^c_{k,ij}$ be the estimated weight of $U_k$ who performs tasks of for $R_i \in \mathcal{R}^i$ in $T_i$, and the global weight of $U_k$ as a worker is:

$$\bar{\gamma}_k^w = \sum_{R_i \in \mathcal{R}^i} \alpha_{c_i} \cdot w^c_{k,ij},$$

where $\alpha_{c_i}$ is the weight of $c_i$ concerning $R_i$ currently.

When $U_k$ acts as a task requester, we assume that $U_k$ correctly evaluates and broadcasts the estimated weights of $\eta_k$ workers$^1$. Hence, the reliability of $U_k$ as a task requester is $\gamma_k^w = \eta_k/\eta$. Similarly, as a verifier, if $U_k$ joins

1. Here $\eta_k$ has been recorded on the blockchain in the prior phase, so does the following number.
in $\lambda$ verifications, out of which $\lambda_k$ verifications are correct, the reliability of $U_k$ as a verifier is $\gamma_k = \lambda_k / \lambda$. From a comprehensive perspective, the global reputation value of $U_k$ can be modeled using the following equation.

$$\gamma_{k,i}^t = (1 - \beta)(\rho_1 \gamma_k^t + \rho_2 \gamma_k^{t-1} + \rho_3 \gamma_k^t) + \beta \gamma_{k,i-1}^t,$$

where $\rho_1$, $\rho_2$ and $\rho_3$ denote the proportion of $U_k$’s reliability as a worker, a task requester, and a verifier, respectively. Moreover, $\rho_1 + \rho_2 + \rho_3 = 1$. $\beta$ denotes the proportion of historical global reputation.

As for $U_k$’s expertise reputation regarding task category $c$, it is modeled as the user’s weight in executing such type of tasks and can be updated as follow.

$$\gamma_{k,c}^t = (1 - \beta)w_{k,c}^t + \beta w_{k,c,i-1}^t.$$

The aforementioned reputation evaluation and update will be conducted automatically by the smart contract when $T_{\text{deadline}}$ arrives.

5 SECURITY ANALYSIS

In this section, we will show that our proposed BRPC system can achieve the security goals defined in Section 3.3.

5.1 Privacy protection

**Theorem 1.** BRPC ensures the privacy of requesters and workers throughout the entire crowdsourcing phase, under the assumption that less than $t$ workers are corrupted out of $|U_t^t|$ participating workers.

**Proof.** Recall that task requesters and workers do not need to provide identity-related information when registering in the system. The real identities of users are protected by the use of pseudonyms, i.e., blockchain addresses. For data privacy, we observe that each sensory data $x_i^t$ is encrypted with the requester’s public key $pk_i$ and then stored in the distributed system. Although everyone can retrieve $E_{pk_i}(x_i^t)$ based on the on-chain metadata, it is infeasible to infer the corresponding plaintext based on the semantic security of the public key encryption, i.e., the difficulty in solving the discrete logarithm. It is a fact that the participating workers are able to jointly decrypt $E_{pk_i}(x_i^t)$ with their own key shares $\{sk_i^k\}_{k=1}^{t}$. However, the number of compromised/collusive workers is assumed less than $t$, which is a basic security requirement in the $(t, |U_t^t|)$ threshold Paillier Cryptosystem. Hence, corrupted workers cannot recover the plaintext as at least $t$ shares are needed for successful recovery. Clearly, only $R_i$ can decrypt it with $sk_i$. Similarly, the confidentiality of $x_i^t$ is also not revealed to anyone except $R_i$. □

5.2 Reliability

**Theorem 2.** BRPC achieves both system and user reliabilities. For a participating worker set $U_t^t$, as long as $t$ honest workers (i.e., verifiers) exist and fewer than $t$ workers collude with each other in the verification process, our protocol guarantees the correctness of data quality and user reputation evaluations without relying on a trusted third party.

**Proof.** Due to the decentralized characteristic of blockchain, the single point of failure can be effectively prevented, which offers system reliability. For data quality evaluation of requesters, QEvaluate is included in each task and is open to all entities. By running QEvaluate, $R_i$ can derive $\omega_k$ of each worker $U_k \in U_t^t$ and obtains the task truths. $\omega_k$ is then broadcasted for worker verification. Based on Algorithm 1, it is observed that $\omega_k$ will be derived without revealing each sensory data as long as $t$ honest workers contribute their decrypted shares and there are fewer than $t$ collusive workers in the remaining verifiers. Furthermore, the comparison between $\omega_k$ and $\omega_k$ would validate the reliability of data quality evaluated by $R_i$. There may exist some malicious workers individually or collusively reporting incorrect verification results, but the final verification result is determined by the majority of honest verifiers. In other words, fewer than half of the verifiers cannot deviate the results even if they are all collusive. Hence the number of collusive verifiers $n_{cv}$ should satisfy (10). Since when $t \geq \|U_i^t\|$, the number of collusive verifiers is bound to be less than $\|U_i^t\|$ even if $\|U_i^t\|$ workers join the verification process. The correctness of data quality verification can be guaranteed with fewer than $t$ collusive workers.

$$n_{cv} < \begin{cases} t & t < \frac{|U_i^t|}{2}, \\ \frac{|U_i^t|}{2} & t \geq \frac{|U_i^t|}{2}. \end{cases}$$

Second, for reputation evaluation, we use smart contracts to conduct the evaluation algorithm, after which the reputation is anchored to the blockchain. Obviously, its reliability depends on the blockchain security and immutability. Therefore, anyone in BRPC cannot successfully tamper with the evaluation results. From another perspective, the reliability of reputation evaluation further improves the reliability of workers selected in the task assignment phase.

5.3 Financial and social fairness

**Theorem 3.** Financial fairness between the task requester and the worker as well as social fairness among the users are both realized, as long as each user is rational in participating crowdsourcing tasks.

**Proof.** From the financial perspective, task requesters and workers need to deposit money on the blockchain before participation. On one hand, for a task requester $R_i$ posting task $\tau_j$, he needs to deposit $B_j$ which is allocated to the participating workers via smart contracts as long as $R_i$ obtain the original sensory data. Additionally, he needs to deposit another deposit $D_j^r_i$ consisting of two parts. For the first part, it can only be redeemed when the quality evaluation of $R_i$ is verified to be correct, or it will be allocated to the honest verifiers. Besides, such honest verifiers will also earn extra rewards from the remaining money of $D_j^r_i$. On the other hand, for a worker $U_k$ performing task $\tau_j$, he can obtain rewards proportional to the data quality as long as the contributed sensory data is submitted before deadline. Otherwise, $U_k$ will get nothing and also lose the first part of his deposit $D_j^r_i$. As such a deposit is much more expensive than the earning rewards and each worker has limited money, a rational worker will not take the risk to launch free-riding, Sybil, and DDos attacks.
We implement the protocol of BRPC atop Ethereum and realized the functionalities of the key phases in crowdsourcing with smart contracts. Specifically, the USC and TSC were written in Solidity and were deployed to a local simulated network Ganache on a laptop with Intel Core i5-3210M CPU (2.50GHz) and 8G RAM. We used Truffle development environment to compile and deploy contracts, and enabled interactions between the front-end and the contracts using Web3.py library. Instead of adding data to the Ethereum blockchain one by one with Truffle console, we created a .py file to provide batch uploading, e.g., upload multiple task metadata at one time. For distributed storage, we adopted IPFS network, to which the users need to upload the task details or the encrypted data before writing into the blockchain. Additionally, for user-side computations, we implemented the CATD algorithm, either in plaintext at the data requester or in encrypted form at the workers, using Java programming language.

In our crowdsourcing scenario, we assumed a series of data sensing tasks that require workers to submit numerical data, e.g., health data monitoring, and peer grading. Initially, 20 workers and 1 requester were registered with initial reputation 0.5, and each of them is assigned 100 ETH coins. The requester had 10 tasks in default during each time slot. Since the difference of datasets has few impacts on the performance evaluation, and it is hard to obtain the sensitive real dataset in practice, we used a synthetic dataset in our experiment. Concretely, all sensory data was generated from a normal distribution with a random mean $\mu \in [20, 40]$ (represents the ground truth) and variance $\sigma^2 \in [1, 2]$. We require that each task is executed by at least one worker, and each worker performs at least one task. Under this premise, each worker randomly determines whether to contribute data to the tasks. We set the significance level $\alpha$ to be 0.05 in CATD. For cryptographic parameters, we set the public key $n$ to be 512 bits to achieve sufficient security. The threshold $t$ was set to be half of the number of the participating workers. We integrated the Paillier Threshold Encryption package\(^2\) into CATD to implement worker-side verification. By referring to the existing plaintext CATD implementations, we set $I_{max}$ to 3 in default.

Since gas consumption is a crucial consideration in Ethereum, we tested the gas consumption and ETH cost incurred in writing data into the blockchain to show the practicality of BRPC. The gas price was set to be the average market price of $2 \times 10^{10}$ Wei. Apart from the contract deployment cost, we respectively varied the number of registered users, the number of posted tasks, and the number of submitted data to show their impacts on gas consumption. In addition, we evaluated the accuracy of BRPC in truth discovery by taking the commonly used standard root of mean square error RMSE as the accuracy metric. On the one hand, we compared RMSE in a different number of iterations and observed the convergence performance. On the other hand, we set the scaling factor $L$ in the range from 10 to $10^5$ to show its impact on the accuracy of the worker-side verification. Meanwhile, the corresponding verification cost was also captured to demonstrate its efficiency.

\(\sigma\) and variance $\sigma^2 \in [1, 2]$. We require that each task is executed by at least one worker, and each worker performs at least one task. Under this premise, each worker randomly determines whether to contribute data to the tasks. We set the significance level $\alpha$ to be 0.05 in CATD. For cryptographic parameters, we set the public key $n$ to be 512 bits to achieve sufficient security. The threshold $t$ was set to be half of the number of the participating workers. We integrated the Paillier Threshold Encryption package\(^2\) into CATD to implement worker-side verification. By referring to the existing plaintext CATD implementations, we set $I_{max}$ to 3 in default.

Since gas consumption is a crucial consideration in Ethereum, we tested the gas consumption and ETH cost incurred in writing data into the blockchain to show the practicality of BRPC. The gas price was set to be the average market price of $2 \times 10^{10}$ Wei. Apart from the contract deployment cost, we respectively varied the number of registered users, the number of posted tasks, and the number of submitted data to show their impacts on gas consumption. In addition, we evaluated the accuracy of BRPC in truth discovery by taking the commonly used standard root of mean square error RMSE as the accuracy metric. On the one hand, we compared RMSE in a different number of iterations and observed the convergence performance. On the other hand, we set the scaling factor $L$ in the range from 10 to $10^5$ to show its impact on the accuracy of the worker-side verification. Meanwhile, the corresponding verification cost was also captured to demonstrate its efficiency.

6.2 Simulation results

Gas consumption. As mentioned, we deployed USC and TSC in Ganache, which consumes 431180 and 1530840 gas, respectively. According to our preset gas price, the corresponding cost is 0.0086236ETH and 0.0306168ETH, leading to a total of $6.13$ transaction fee when referring to the average ETH price $\$156.45$ in Jan. 2020 (when we conducted the experiment). Such cost is not a big concern as the smart contracts were deployed only once.

Fig. 4 depicts the gas consumption in the primary phases of BRPC versus the number of data records written into a blockchain. Our result shows a linear increase with more users registered, more tasks posted/accepted, and more data submitted. This is because gas consumption is closely

related to the size of data recorded on the blockchain and the type of computation operations. For the same phase with the same operations, the gas will be increased when more data records are written via smart contracts. However, we notice that task publishing requires more gas, about 10700000, i.e., $33.48$ consumption when compared to other phases for 50 written data records. Because, besides the task identifier, the task requester also needs to store the deadline, reputation thresholds, and budget information on-chain for later task matching and reward allocation. Moreover, the budget and deposit are also transferred to a third-party address, resulting in additional gas cost. For other phases, user registration and data submission have a similar number of written data, whereas accepting a task only needs to determine whether to add the requested workers into the task’s worker list and charge the worker’s deposit. Hence the former two phases present comparable gas cost, while task acceptance is the most gas-efficient phase with only around 2500000, i.e., $7.8$. The cost is acceptable as crowdsourcing tasks generally come with enough reward budget for incentive. According to Amazon Mechanical Turk AMT policy, the requester needs to pay a 20% fee on the reward and bonus amount he pays workers and will be charged an additional 20% fee on the reward more if he has more than 10 tasks. Moreover, there is an additional fee for using the masters or premium qualifications. For more than 10 tasks with $100$ rewards, at least $40$ service fee is charged. In contrast, requesters in BRPC only needs to pay the transaction fee independent of the reward amount, which saves the service cost and caters for practical large-scale crowdsourcing scenarios.

Similarly, we evaluate the gas consumption generated for reward payment and user reputation update, respectively. As illustrated in Fig. 5, the gas consumption linearly scales up when the requester gives rewards to more workers (including reward verifiers and deposit refund), and the reputation of more workers are updated. This is in accord with that in Fig. 4 for similar reasons. When all workers are paid, the corresponding gas is $633776$, which is about $1.98$. In contrast to money transfer, it is more expensive to update the worker’s reputation ($7.3$). Note that ETH price fluctuates a lot depending on the marketplace. The transaction fee reported above may be different over time. Nevertheless, we can see that the transaction fee is acceptable as the cost to ensure privacy, security, and fairness in our distributed and reliable crowdsourcing systems. Clearly, these requirements cannot be achieved in traditional centralized crowdsourcing platforms.

**Accuracy & convergence.** We report the impact of $L$ on the accuracy of our encrypted CATD, which also affects the correctness of the verification result. As shown in Fig. 5, we observe that the difference in $L$ does not affect the accuracy of the plaintext CATD at the requester (refer to R-plaintext in the figure). In contrast, a small $L$ results in a slightly higher RMSE on the encrypted data. This is because most fractional parts of sensory data and intermediate results are removed for computations on integers. However, when $L$ is 1000 or higher, the worker-side encrypted CATD (refer to W-Encrypted) achieves nearly the same RMSE as that at the requester. We can guarantee the accuracy of the worker-side verification as long as $L$ is large enough. Hence, unless otherwise specified, we set $L$ as $10^4$ for Algorithm 1.

For convergence performance, in Fig. 6, we present the evolution of RMSE with the number of iterations in different CATD settings. As observed, the encrypted CATD compromises the accuracy when $L = 10$, which is consistent with the result in Fig. 5. In contrast, when $L = 10^4$, the result reveals that the convergence of encrypted CATD matches that in the plaintext form. Moreover, the algorithm can reach convergence with the least RMSE when the number of iterations is 3. Therefore, we can quickly derive each estimated truth and worker’s weight in a few interactions without degrading the estimation accuracy.

**Worker-side verification cost.** Since computations on encrypted data are more costly than that in the plaintext domain, it is a great concern, especially for the resource-constrained users. We sample five sets of data with different number of workers and tasks, i.e., $20 \times 10, 20 \times 30, 20 \times 40, 20 \times 50, 50 \times 20$. In Fig. 7, we report the worker-side verification time in three iterations with respect to the number of submitted data. It is observed that the number of submitted data ($112, 223, 309, 440, 528$) in our random participation is more than half of the quantity of all-worker participation. Recall that each iteration consists of truth update and weight update. We denote the two key components share decryption and share combination in weight update as SD and SC, respectively. From the figure, we can see that the SD operation linearly scales as more data is involved and is much more expensive than others. This is reasonable as each worker needs to partially decrypt $E_{pk}(d_i^2)$, $\tau_j \in T_{U_k}^i$, $U_k \in U_k^i$, which is exactly the number of submitted data. Moreover, each partial decryption includes one expensive modular exponentiation operation. In contrast, the truth update cost scales with $m$ and SC only needs to perform a one-time share combination with one modular multiplication and one modular exponentiation operations. As expected, Fig. 7 shows a much lower and stable cost for truth update and SC when more data is submitted, only requiring 100ms for 528 data. For the entire privacy-aware verification, the time overhead at the worker is nearly 2600ms when 50 workers and 20 tasks are involved. This result is acceptable for such a privacy-aware and reliable system.

### 7 RELATED WORK

In this section, we thoroughly review related researches on crowdsourcing systems, including centralized and decentralized crowdsourcing services. Mainly, we focus on discussing the state-of-the-art blockchain-based solutions.
Many efforts have been made to develop crowdsourcing systems that serve diverse applications in a centralized manner, such as Amazon Mechanical Turk and Gigwalk.3

For successful deployment, these crowdsourcing systems commonly rely on the centralized crowdsourcing platform/server to provide the essential crowdsourcing services such as worker selection, task allocation, data evaluation, truth discovery [16], incentive provision [29], and reputation management. Dang et al. proposed a general worker quality evaluation algorithm for various types of tasks and used MapReduce parallel programming for high computing performance. Considering the requirements of both participants and systems, a series of privacy-aware task allocation methods [35], [41], privacy-preserving and truthful incentive mechanisms [33], secure truth discovery protocols [44], and anonymous reputation assessment models [32] were separately investigated in existing works. In [36], the authors systematically addressed user privacy, data trustworthiness, and incentive provision issues. However, due to the centralized system architecture, the above solutions only work in a fully trusted or honest-but-curious server model, and inherently suffer from a single-point-of-failure and DDoS/Sybil attacks. Once the trust assumption of centralized platforms is removed in practical applications, the system security and functionalities will be compromised.

7.2 Decentralized crowdsourcing services

Recently, blockchain technology has been tentatively integrated into various crowdsourcing systems, with a focus on overcoming the shortcomings in centralized crowdsourcing services. In [15], a blockchain-based system was built for the specific crowdfunding applications. In the context of vehicular edge computing and networks, Kang et al. [17] investigated a secure and efficient data sharing mechanism by leveraging the consortium blockchain and smart contract technologies. Some researches [8], [31] explored blockchain-assisted crowdsourced energy ecosystems.

Aiming at constructing generalized crowdsourcing systems, Li et al. [18] proposed a blockchain-based decentralized crowdsourcing framework named CrowdBC, which removes the crowdsourcing platforms and designs three smart contracts to implement each phase in crowdsourcing. By cooperating blockchain with the traditional three-party crowdsourcing structure, a framework FLUID was designed in [12] to provide transparent incentive mechanisms and enable workers to share their profiles among different platforms. To mitigate the security threats from malicious participants, Gu et al. [11] proposed a blockchain-based participant management framework CrowdChain for fog-assisted crowdsensing. However, these aforementioned works only offer fundamental frameworks for blockchain-based crowdsourcing while not specifying some key components such as reliable data quality evaluation, accurate reputation assessment, truth discovery, incentive provision, and fair interactions. It is nontrivial to implement these functionalities in such novel blockchain-assisted systems, especially when user’s privacy requirements are considered.

For data quality and user reliability issues, An et al. [1] modeled a crowdsensing quality control method via blockchain, in which the authors presented a credit-based verifier selection strategy and a two-consensus approach. The data quality is controlled with consideration of matching degree and quality grading evaluation. WorkerRep [2] was developed to build trust on a crowdsourcing platform using blockchain and prevent fake reputations. However, the sensed data of workers are revealed to the evaluators for worker trust evaluation. Edge computing was integrated with blockchain into the crowdsourcing in [43]. They presented a privacy-preserving reputation management scheme to resist malicious users, in which the requester was considered trusted for local reputation computation (fail to prevent the requester’s misbehaviors). Although Zhang et al. [42] described that any worker could verify the correctness of evaluation and reward allocation without knowing the sensory data, their verification method via comparing two ciphertexts had flaws since two ciphertexts may be different even if they have the same plaintext. Recently, a truthful incentive mechanism [13] was proposed for distributed P2P applications on the basis of blockchain. From the perspective of privacy and incentive, Wang et al. [30] designed a blockchain-based secure incentive mechanism for crowdsourcing, relying on the miners to verify the data quality and further achieves k-anonymity from the miners when verifying transactions. Aside from that, [40] focused on solving the location privacy in crowdsensing. To overcome two fundamental challenges: data disclosure and identity breach in decentralized crowdsourcing, a private and anonymous decentralized crowdsourcing system ZebraLancer [22], [23] was proposed. Moreover, a hybrid blockchain crowdsourcing platform zkCrowd [45] was improved in order to relieve the performance bottleneck further. It elaborates on an innovative blockchain structure by integrating a public blockchain with Delegated Proof of Stake (DPOS) and multiple sub-chains with BPFT, which are in charge of public and private tasks, respectively. Although providing customized privacy protection with low cost, the fundamental functions of smart contracts and the zero-knowledge proof are still far away from successful implementations.

Another line of research, orthogonal to our work concerns, is to improve the consensus mechanism used in existing blockchain-based crowdsourcing. Feng et al. [9] proposed a blockchain-based mobile system MCS-Chain.

and designed a novel consensus mechanism for new block generation by referring to the total amount of payments waiting to be stored in the next block. A novel consensus protocol called Proof-of-Trust (PoT) [46] was devised to suit the crowdsourcing and the general online service industry. PoT selects transaction validators based on the users’ trust, which avoids the intensive resource consumption of PoW and the scalability issue in traditional BFT algorithms.

8 Conclusion

In this paper, we present a blockchain-based reliable and privacy-aware system, named BRPC, for universal crowdsourcing. In contrast to prior work, BRPC considers a more practical crowdsourcing scenario where multi-worker join a set of tasks during different time slots. We enable the requesters to efficiently and accurately infer the task truths from unreliable data. Meanwhile, we provide an efficient data quality evaluation method based on CATD and avoid the inefficient evaluation of every single data. Particularly, privacy-aware computation verification is proposed to detect the incorrect evaluation of malicious requesters. Moreover, we consider a user’s different roles and establish a multi-dimensional reputation model with a reliable computation and efficient update, easing the burden of blockchain storage. Based on that, both social and financial fairness are achieved with our rewarding/punishment solution. Security analysis and experimental results over Ethereum validate the predefined requirements and application potential of BRPC.

References


Liangmin Wang received his B.S. degree (1999) in Computational Mathematics in Jilin University, Changchun, China, and the Ph.D degree (2007) in Cryptology from Xidian University, Xi’an, China. He is a full professor in the School of Computer Science and Communication Engineering, Jiangsu University, Zhenjiang, China. He has been honored as a “Wan-Jiang Scholar” of Anhui Province since Nov. 2013. His research interests include data security & privacy, and blockchain. He has published over 60 technical papers at premium international journals and conferences, like IEEE/ACM Transactions on Networking, IEEE Transactions on Intelligent Transportation Systems, IEEE Transactions on Vehicular Technology, IEEE Global Communications Conference, IEEE Wireless Communications and Networking Conference. Dr WANG has served as a TPC member of many IEEE conferences, such as IEEE ICC, IEEE HPCC, IEEE TrustCOM. Now he is an associate editor of Security and Communication Networks, a member of IEEE, ACM, and a senior member of Chinese Computer Federation.

Guoliang Xue is a professor of Computer Science and Engineering at Arizona State University. He received the Ph.D degree in Computer Science from the University of Minnesota in 1991. His research interests span the areas of Quality of Service provisioning, network security and privacy, crowdsourcing and network economics, RFID systems and Internet of Things, smart city and smart grids. He has published over 280 papers in these areas, many of which in top conferences such as INFOCOM, MOBI-COM, NDSS and top journals such as IEEE/ACM Transactions on Networking, IEEE Journal on Selected Areas in Communications, and IEEE Transactions on Mobile Computing. He has received the IEEE Communications Society William R. Bennett Prize in 2019 (best paper award for IEEE/ACM TON and IEEE TNSM in the previous three years). He was a keynote speaker at IEEE LCN’2011 and ICNC’2014. He was a TPC Co-Chair of IEEE INFOCOM’2010 and a General Co-Chair of IEEE ONS’2014. He has served on the editorial board of IEEE/ACM Transactions on Networking. He served as the Area Editor of IEEE Transactions on Wireless Communications, overseeing 13 editors in the Wireless Networking area. He is the Steering Committee Chair of IEEE INFOCOM.

Haiqin Wu received her B.S. degree in Computer Science and Ph.D degree in Computer Application Technology from Jiangsu University in June 2014 and September 2019, respectively. She is currently a postdoctoral researcher in the Department of Computer Science, University of Copenhagen, Denmark. During her Ph.D study, she was a visiting student in the School of Computing,Informatics, and Decision Systems Engineering at Arizona State University, US. Her research interests include data security and privacy protection, mobile crowdsensing, and blockchain application.

Boris Düdder received his Diploma degree (2008) and the Ph.D. degree (2014) in Computer Science at the Technical University of Dortmund, Dortmund, Germany. He is associate professor in software engineering in the Department of Computer Science at the University of Copenhagen, Copenhagen, Denmark, where he is head of the research group Formal Methods & Software Engineering. His research interests are in formal methods in software engineering and secure and reliable distributed systems. He has published over 30 technical papers at premium international journals and conferences. Dr. Düdder has served as TPC and PC member of many ACM conferences. His background is both academic and industry, e.g., Fraunhofer, Germany, and Microsoft, USA.

Shipu Sun received the B.S. degree in electrical engineering from Zhengzhou University, Zhengzhou, China, in 2015. He is currently pursuing the M.S. degree in the School of Computer Science and Communication Engineering, Jiangsu University, Zhenjiang, China. His current research interests include cryptography, network and data security, and blockchain.