Research Article

Blockchain-Based Crowdsourcing Framework with Distributed Task Assignment and Solution Verification

Shasha Li, Xiaodong Bai, and Songjie Wei

School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing 210094, China

Correspondence should be addressed to Songjie Wei; swei@njust.edu.cn

Received 17 November 2021; Accepted 3 February 2022; Published 19 March 2022

Academic Editor: Zhili Zhou

Copyright © 2022 Shasha Li et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Internet backboned crowdsourcing utilizes network-wide resources to solve complicated and large-scale tasks, which are not accomplishable for independent individuals. Existing crowdsourcing platforms are mostly centralized solutions with reliability and trustworthiness fragile to single-point failures on the central servers. The innovation of distributed ledgers as blockchain inspires us to optimize the traditional crowdsourcing procedure with distributed sustainability. We propose a blockchain-based design of the distributed secure crowdsourcing scheme for task distribution and result verification without relying on any third trusted institution. A preference-based task distribution (PTD) mechanism is presented which guarantees the percentage of task distribution and the satisfaction of the chosen workers. Task works are continuously assessed for reputations based on their historical behaviors. Task completion correctness is verified by blockchain consensus in two different scenarios after workers submit their results with reputations. We implement a prototype system based on the Ethereum chain with PTD and solution verification components. With various tasks and scenarios evaluated in the system, the proposed distributed crowdsourcing framework shows system reliability, data security, and scenario feasibility.

1. Introduction

As the most successful data and social communication network globally, the Internet system interconnects computer and data sources, making collaboration possible in private, public, academic, business, and government networks. Crowdsourcing formalizes and systemizes such distributed collaboration with tasks spread to multiples and result integrated and verified as one [1]. While the concept has been continuously developed since 2006, various crowdsourcing systems have succeeded in both academic research and industrial applications. Regarding as a practical task distribution mechanism, crowdsourcing usually integrates collective intelligence to solve enormous tasks with high complexity [2]. Nowadays, many crowdsourcing platforms are offering business services publicly. Most of them are centralized, such as Amazon Mechanical Trunk (Mtrunk), Upwork, and Wave.

With the extensive usage of crowdsourcing, the drawbacks of centralized infrastructure also concern users heavily. Traditional crowdsourcing systems are generally server-centric with distributed workers and users, highly depending on a trusted third party to maintain the whole service process and the fair exchange between task solutions and rewards [3, 4]. This centralized operation way makes a single-point failure extremely easy to happen and prone to vulnerabilities for malicious attacks such as “false-reporting” (i.e., dishonest task publisher refusing to pay for the task despite the task being completed successfully) and “free-riding” (i.e., dishonest workers receiving rewards without any contributions to the task) [5]. Moreover, the task distribution rules in the centralized crowdsourcing system are usually set according to the requests’ demands only, leading to the low task distribution percentage, as parts of the preferred workers have been allocated, which dramatically decreases the crowdsourcing participants’ satisfaction.

In recent years, as a highly innovative and emerging technology, blockchain has been widely applied in many fields with success. With the advantages of decentralized consensus, cryptography, peer-to-peer network, smart
contract, and other technical strengths, blockchain technology for crowdsourcing can enable task publishers and workers who are original strangers to establish mutual cooperation trust quickly and avoid the centralization defects in crowdsourcing systems [6, 7]. A blockchain-based crowdsourcing application system becomes promising with fault-tolerance, peer-to-peer credibility, information traceability, and function extensibility. However, blockchain, as an evolving technique, also poses several practical challenges. For example, data transparency ensures that everything submitted on blockchain is publicly visible among all nodes, making it easy for malicious participants to steal task data and results. Furthermore, free-riding attacks are even more straightforward to implement on blockchain than centralized crowdsourcing systems. Besides, crowdsourcing tasks usually contain a large amount of data. The performance weakness in distributed systems makes data storage and processing a nonnegligible concern on blockchain.

This study proposes a blockchain-based decentralized crowdsourcing framework with a task distribution mechanism considering both the publishers and workers implemented. A preference-based task distribution mechanism is designed to better match requirements from task publishers with capabilities at task works with higher mutual satisfaction coverage. Task completion results are evaluated for correctness by multiple parties, with conclusions calculated and guaranteed based on blockchain consensus. Sometimes, the solution cannot be known without correct calculation for a given task. While the traditional crowdsourcing systems assume more trust in the task publishers’ honesty or the fairness of central coordinators, it is not trust-oriented but role-based. We construct a centreless distributed task result evaluation with trackable and verifiable trust on blockchain both in the past and currently. Node reputation is used to facilitate the evaluation procedure from the past perspective, and majority voting lowers the success chance of current node misbehaving. We have implemented all the presented mechanisms on an Ethereum-based blockchain system with function specifications for crowdsourcing. Theoretical analysis shows the security robustness of the prototype system, and empirical tests demonstrate its reliability and efficiency.

The rest paper is organized as follows. Section 2 gives an overview of the related work and technology. Section 3 proposes the blockchain-based decentralized crowdsourcing framework and the detailed crowdsourcing procedures. Sections 4 and 5, respectively, describe the task distribution mechanism and solution verification implemented on the proposed framework. Section 6 presents the theoretical and experimental analysis. Section 7 concludes with a summary of the contributions and possible future work.

2. Related Work

2.1. Centralized Crowdsourcing System (CCS). As the world’s largest human resources’ platform, Upwork [8] represents the typical centralized crowdsourcing system (CCS). However, it was forced to shut down a series of services by a Distributed Denial-of-Service (DDoS) attack in May 2014. CCS has the whole crowdsourcing process relying on a centralized server which makes the system always vulnerable to malicious attacks. Such trust-based server-centric system establishment also suffers the risks of low motivation, information leakage, and unfairness. Therefore, past research studies on centralized crowdsourcing schemes mainly focus on setting incentive mechanisms and solving fairness issues. For example, based on auction, Zhang et al. [9] propose a centralized crowdsourcing scheme named EFF and DFF, which realizes dispute arbitration with the assistance of a trusted third party. Meanwhile, a reputation-based incentive mechanism is widely used in crowdsourcing. Others establish a reputation model in the crowdsourcing system, which solves the fairness problem to some extent but still ignores the confidentiality of crowdsourcing information [10].

2.2. Blockchain-Based Crowdsourcing System (BCS). Given the defects of centralized crowdsourcing schemes above, recent scholars have conducted in-depth studies on blockchain-based crowdsourcing schemes. Some use blockchain as a payment channel for crowdsourcing task rewards, but the scheme fails to defend against the malicious attacks because the crowdsourcing-related information is completely open [11]. Others try to design a blockchain-based crowdsourcing framework such as CrowdBC [12]. Even if such a scheme uses asymmetric encryption and signature algorithms, the occurrence of free-riding attacks still cannot be avoided entirely. Solution evaluation is an inevitable challenge in distributed crowdsourcing. Hybrid blockchain crowdsourcing platforms, such as zkCrowd [13], consist of a public chain running DPOS consensus and multiple private sub-chains running PBFT consensus to form a hybrid blockchain architecture. Like CrowdBC, the scheme does not specify the solution validation method and is similarly challenged to resist free-riding attacks.

To sum up, although the current blockchain-based crowdsourcing schemes can better alleviate the defects of the traditional centralized ones in single point of failure, lack of trust, and other weaknesses, they always are short of the task distribution mechanism with high task completion and workers’ satisfaction, as well as the reliable verification of crowdsourcing achievements.

As a makeup for the shortcomings of the existing centralized crowdsourcing platform, we propose a generalized blockchain-based distributed crowdsourcing framework with a task distribution mechanism and solution verification method. In a nutshell, the contributions of this study are as follows:

1. The proposed framework automatically uses smart contracts to control the crowdsourcing process without trusted third parties. It ensures efficient data storage and processing by combining off-chain storage with on-chain verification. In on-chain interaction, commitment protocol solves the contradiction between blockchain transparency and data privacy.
(2) Analytic hierarchy process (AHP) is flexibly applied to generate the preference list of crowdsourcing participants. Furthermore, considering both the preference list of tasks and workers, the preference-based task distribution (PTD) mechanism is proposed. The proposed mechanism is proved to improve the overall distribution percentage as well as the satisfaction of task distribution in crowdsourcing.

(3) Given the inherent problem of solution evaluation, an elastic reputation model referring to users’ historical behavior is introduced to the proposed framework. Based on it, solution evaluation strategies in the distributed environment are designed to realize the trusted verification process without the involvement of publishers or workers.

(4) Theoretical analysis verifies that the proposed blockchain-based crowdsourcing scheme can effectively resist the common attacks in crowdsourcing. We implement a prototype system and conduct experiments to show that the proposed PTD mechanism has a relatively high task distribution percentage and gains the highest satisfaction of workers than other mechanisms. Besides, the framework is proved to have low performance overhead and is feasible to implement in practice.

3. Blockchain-Based Crowdsourcing System

3.1. System Model. As illustrated in Figure 1, the proposed blockchain-based decentralized crowdsourcing system consists of the following functional roles and components.

(1) Publisher: a publisher is the system user who initiates a crowdsourcing task. Before publishing, the publisher needs to deposit enough budgets in the smart contract on blockchain for the task’s reward payment.

(2) Worker: a worker processes a crowdsourcing task and produces a solution for it. After depositing on blockchain, a user is qualified to be a worker and then competes for a task. Once assigned a task, a worker can submit solutions through the crowdsourcing system and get the corresponding rewards if the solutions pass validation.

(3) Evaluator: an evaluator is the one who participates in the solution assessment and only exists in the solution assessment stage in a task cycle. Depending on the type of task, the smart contract or other workers in the system may play the evaluator role.

(4) Blockchain: it is the decentralized platform for the publisher, worker, and evaluators to communicate with one another. All operations of different stages in a whole crowdsourcing task cycle are completed via blockchain. The crowdsourcing-related information is transferred through blockchain transactions.

The proposed system comprises three layers: an application layer, a blockchain layer, and a data storage layer. As shown in Figure 2, the application layer exposes the service interfaces for users to interact directly with the blockchain for functions such as user registration, task submission, and solutions’ management. The blockchain layer implements the major crowdsourcing procedure functions as smart contracts. The crowdsourcing-related data are maintained on-chain for task interactions and transactions and off-chain for a large amount of task content data. The storage layer uses the interplanetary file system (IPFS) to store raw data, including task files and solutions. Only task metadata are stored in the blockchain layer, such as the location pointers to task contents and the hash values.

3.2. System Overview. This section explains the individual phases of crowdsourcing. Symbols used and their meaning are defined in Table 1, to facilitate the following presentation. Only registered users are allowed to participate in crowdsourcing. There are two alternative roles for a user to behave, switching between a publisher and a worker in different task cycles. That means a user can be a publisher to announce tasks or be a worker to undertake other’s tasks. The registration process is similar to the other blockchain systems. When registering, users only provide their public key pk_U without any identity information. pk_U is used to generate users’ unique blockchain address ID_U. Compared with the traditional centralized crowdsourcing system, blockchain address excludes sensitive information that can protect privacy safety to the greatest extent.

A complete task cycle in crowdsourcing includes five phases.

3.2.1. Task Announcement

(a) Task preparation: a publisher R_t generates a pair of keys for the task ID_T with the public key pk_T announced. Any worker can use pk_T to encrypt a solution guaranteed on the off-chain storage.

(b) Task announcement: R_t uses the interface provided by the system as application client to publish a task via blockchain. The task is published as publishTask(ID_T, t_Type, td, tr, Pointer_ID_T, τ, N_W, p, pk_T), t_Type ∈ TCA, TUA means the task types, including task with certain answer (TCA) and task with uncertain answer (TUA). td is the task description, and tr is the task requirements in which R_t needs to describe the preferences for workers, and the corresponding weights are to help generate the preference list, which will be detailed and described in Section 4.1. Pointer_ID_T is the pointer of the task-related files. dt = \{d_t1, d_t2\} represents the deadline. t_1 and t_2 are, respectively, the deadline for request and submission. τ is the budget for task reward and N_W is the maximum number of expected workers. \{ep, rp\} means the policy, where ep and rp are the evaluation policy and the reward policy, respectively. After
Figure 1: System model of blockchain-based decentralized crowdsourcing system.

Figure 2: Architecture of blockchain-based decentralized crowdsourcing system.

Table 1: Symbol definitions.

| Symbol | Definition |
|--------|------------|
| $U \in \{R, W, E\}$ | Crowdsourcing participants or system users, including publisher $R$, worker $W$, and evaluator $E$ |
| $pk$ | Public key for a crowdsourcing participant |
| $sk$ | Secret key for a crowdsourcing participant |
| ID$_P$ | Blockchain address for a crowdsourcing participant |
| ID$_T$ | Unique identification for a crowdsourcing task |
| $rv$ | Reputation value for a crowdsourcing participant |
| $E(\cdot)$ | Asymmetric encryption algorithm |
| hash(\cdot) | Secure hash function |
publishing, a task contract for the task $ID_T$ is generated, and $R_j$ needs to deposit enough blockchain credits in it.

3.2.2. Task Assignment. A worker may volunteer to undertake a task. The worker $W_j$ who expects to complete the task should send the request to the task contract in the form of requestTask($ID_T, ID_{W_j}, rv_{W_j}, rt$), where $rt$ is the timestamp for the request.

The task contract may issue the permission Permit($ID_T, ID_{W_j}$) while receiving the request. A worker’s reputation value is evaluated for permit issuance. A task can no longer be requested if the number of existing requests is over $N_W$.

3.2.3. Solution Submission. To prevent the free-riding attack, a worker submits its solution digest as commitment instead of the solution content onto the blockchain contract. The related operations of commitment generation are off-chain.

A commitment is one of the significant cryptographic primitives that refers to a two-stage protocol between the committer and the verifier. The first stage is to commit. The committer sends the commitment of the message $m$ to the verifier to guarantee $m$ not revisable. The second stage is to verify the commitment.

The common commitment protocol generally utilizes three polynomial-time algorithms:

- **Setup** ($1^k$) = crs: the algorithm inputs a “1” bit string of length $k$ and outputs an open common reference string crs.
- **Commit** ($crs, m$) = ($com, evi$): the inputs are the common reference string crs and the message $m$ required to be committed. The outputs are the commitment com and the evidence evi used to verify the commitment. The commitment com is publicly exposed.
- **Verify** ($crs, com, evi, m$): the committer releases the message $m$ and the evidence evi to the verifier, which checks if the output of Verify is 1.

The common commitment protocol has the following two characteristics:

- Hiding: the commitment com does not leak any information about the message content $m$.
- Binding: no committer can easily make another message $m'$ ($m' \neq m$) pass validation

So, the worker $W_j$ submits a solution before the deadline $T_2$ in the form of submitAnswer($ID_T, ID_{W_j}, st, (crs_{W_j}, com_{W_j}), Pointer_{W_j}, Permit$). $st$ is the timestamp of submission and Pointer$_{W_j}$ is the pointer of encrypted solution $E(Ans_{W_j}/pk_v)$. The detailed steps look like the following.

First, $W_j$ uses the blockchain address to generate the common reference string, i.e., Setup($1^{ID_{W_j}}$) = crs$_{W_j}$. Then, $W_j$ executes Commit($hash(Ans_{ID_{W_j}}), crs_{W_j}$) to generate the commitment $com_{W_j}$ and the evidence $evi_{W_j}$ of solution $hash(Ans_{W_j})$. At last, ($crs_{W_j}, com_{W_j}$) is submitted as the solution in this phase.

3.2.4. Solution Assessment

(a) Commitment verification: $W_j$ opens the solution $hash(Ans_{W_j})$ and the evidence $evi_{W_j}$ first. Task contract executes the algorithm Verify to verify whether the solution has been modified or not in the form of Verify($crs_{W_j}, com_{W_j}, de c_{W_j}, hash(Ans_{W_j})$). If the commitment passes the verification, then enter the next step. If not, the solution is discarded, and this crowdsourcing activity is deemed dishonest with the punishment of deducting all the task deposits.

(b) Solution verification: the verification methods are divided into two scenarios according to the task type. If the task type is TCA, the smart contract behaves as an evaluator to execute the verification automatically. If the task type is TUA, other workers act as the evaluators to participate in the assessment work. As in Figure 3, the workers who are willing to assess others’ solutions send their requests to the task contract. The contract chooses the final evaluators as the method illustrated in Section 5. After the evaluator election, the evaluator candidates are informed of the final results. All the chosen evaluators generate a set of keys ($pk_e, sk_e$) as the key agreement methods in [15, 16]. Group public key $pk_v$ is sent to the workers to encrypt the solutions to ensure the solutions are only publicly open to the evaluators. And then, all the evaluators use $sk_e$ to decrypt and execute the commitment verification algorithm again to avoid the solution-modifying cheats. If the workers are found dishonest, all their task deposits can be deducted as a fine. If the commitments pass verification, the evaluators assess the solutions and give corresponding scores respectively. Both assessment methods are detailed in Section 5. After assessment, the smart contract collects the final solutions of the task.

3.2.5. Task Settlement. The smart contract for the task receives the instructions and automatically triggers the reward distribution. The collected solutions as task answers are sent to the publisher, and the reward distribution work is conducted according to the reward distribution policy $rp$. The workers who have previously been permitted for the task but have not submitted solutions will have their task deposits fully deducted. After reward distribution, the remaining is returned to the publisher.

According to the contributions, when the cycle of a crowdsourcing task finishes, roles who participate will have their reputations updated to the corresponding values. The reputation calculation is presented in Section 5 with details.
Task Distribution

Task distribution usually refers to distributing specific tasks to selected workers according to the established rules, to ensure the maximum resource utilization or high efficiency of task completion. Due to the open characteristic of crowdsourcing, the task distribution process is highly complicated for the heterogeneous task difficulties and workers’ abilities. Currently, the existing task distribution mechanisms have an apparent preference for task publishers, leading to the lack of workers’ satisfaction.

To improve the satisfaction of task distribution, we apply the analytic hierarchy process (AHP) to generate the preference list for both the tasks and the workers, and the preference-based task distribution (PTD) mechanism is proposed, guaranteeing the task distribution percentage and improving the chosen workers’ satisfaction.

4.1. Preference List. The preference list reflects the order from the most preferred worker/task to the least favourite worker/task for a particular task/worker. AHP is introduced in generating the preference list for both the tasks and the workers, and the preference-based task distribution (PTD) mechanism is proposed, guaranteeing the task distribution percentage and improving the chosen workers’ satisfaction.

4. Task Distribution

Step 1: establish a pairwise judgment matrix (PJM) for the tasks and workers. As illustrated below, every element $a_{ij} (i, j = 1, 2, \ldots, m)$ in the matrix reflects the task’s or the worker’s relative preference of attribute $a_i$ over another attribute $a_j$. According to the design principle of AHP, $a_j$ is usually defined from 1 to 9 with increasing importance. Obviously, PJM satisfies $a_{ji} = 1/a_{ij}$:

\[
\begin{pmatrix}
a_{11} & a_{12} & a_{13} & \cdots & a_{1m} \\
a_{21} & 1/a_{12} & a_{13} & \cdots & a_{1m} \\
a_{31} & 1/a_{13} & 1/a_{23} & 1 & \cdots & a_{3m} \\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
a_{m1} & 1/a_{1m} & 1/a_{2m} & \cdots & 1/a_{3m} & \cdots & 1
\end{pmatrix}
\]

Step 2: calculate the weigh factor $w_i$ of each attribute $a_i$ to generate vector $W = (w_1, w_2, \ldots, w_m)^T$. The calculation of $w_i$ is defined as

\[
w_i = \frac{\left(\prod_{j=1}^{m} a_{ij}\right)^{1/m}}{\sum_{i=1}^{m} \left(\prod_{j=1}^{m} a_{ij}\right)^{1/m}}.
\]

Obviously, it satisfies $\sum_{i=1}^{m} w_i = 1$.

Step 3: establish the choices’ matrix $C$ by combining all the choices and their corresponding attributes. For the task, choices are the alternative workers. Similarly, choices are the alternative tasks for the

\[
\begin{array}{c}
\text{Task Contract} \\
\text{Worker} \\
\text{IPFS} \\
\text{Evaluators} \\
\text{Solutions Encrypted solutions}
\end{array}
\]

Figure 3: Diagram of obtaining task answers for evaluators (Step 1: send requests for evaluation; Step 2: return the result of evaluator selection; Step 3: send the public key of evaluators; Steps 4–7: store and upload after encrypting the answer plaintext; Steps 8–10: get the pointer, download, and decrypt the answer ciphertext).

Security and Communication Networks
worker. As illustrated below, \( v_{ij} \) means the value of attribute \( a_i \) of choice \( c_j \):

| Choice/Attribute | \( a_1 \) | \( a_2 \) | \( a_3 \) | ... | \( a_m \) |
|------------------|---|---|---|---|---|
| \( c_1 \)        | \( v_{11} \) | \( v_{12} \) | \( v_{13} \) | ... | \( v_{1m} \) |
| \( c_2 \)        | \( v_{21} \) | \( v_{22} \) | \( v_{23} \) | ... | \( v_{2m} \) |
| \( c_3 \)        | \( v_{31} \) | \( v_{32} \) | \( v_{33} \) | ... | \( v_{3m} \) |
| ...              | ... | ... | ... | ... | ... |
| \( c_n \)        | \( v_{n1} \) | \( v_{n2} \) | \( v_{n3} \) | ... | 1 |

(3)

Step 4: normalize the choices matrix \( C \) to balance the contribution of all the attributes made to the preference. For each attribute set by the task or worker, its value might be high or low as its negative or positive contribution. In this way, if attribute \( a_j \) makes positive contributions to the preference, \( v_{ij} \) is changed to \( v_{ij}/\max_{m}^{m} v_{ij} \), otherwise \( v_{ij} \) is changed to \( \min_{m}^{m} v_{ij}/v_{ij} \). The normalized choices matrix is named \( C' \).

Step 5: calculate the final scores and sort them in the descending order to get the preference list. Construct the final score matrix \( S \) by calculating

\[
S_{nx1} = C'_{nxn} W_{mx1}.
\]

Take the preference list of the task \( T_i \), for instance, the reputation value \( (a_1) \), expected rate of reward \( (a_2) \), and the number of completed tasks \( (a_3) \) are deemed to be the three attributes when choosing workers. The publisher of \( T_i \) sets \( a_1 = 7, a_2 = 4, \) and \( a_3 = 2 \) which regard \( a_4 \) as 7/4 times as \( a_2 \) and \( a_2 \) as 2 times as \( a_3 \). The PJM of \( T_i \) is as follows:

| Attribute/Attribute | \( a_1 \) | \( a_2 \) | \( a_3 \) |
|---------------------|---|---|---|
| \( a_1 \)           | 1 | 2 | 7/2 |
| \( a_2 \)           | 4/7 | 1 | 2 |
| \( a_3 \)           | 2/7 | 1/4 | 1 |

(4)

The vector of weigh factors \( W = (0.57, 0.31, 0.12)^T \) is obtained by formula (2). Suppose there are 4 alternative workers. Assume the choice matrix \( C \) is constructed as

| Choice/Attribute | \( a_1 \) | \( a_2 \) | \( a_3 \) |
|------------------|---|---|---|
| \( c_1 \)        | 0.95 | 0.12 | 10 |
| \( c_2 \)        | 0.83 | 0.15 | 23 |
| \( c_3 \)        | 0.67 | 0.20 | 6 |
| \( c_4 \)        | 0.74 | 0.17 | 12 |

(5)

After normalization, \( C' \) is changed into

| Choice/Attribute | \( a_1 \) | \( a_2 \) | \( a_3 \) |
|------------------|---|---|---|
| \( c_1 \)        | 1 | 1 | 0.65 |
| \( c_2 \)        | 0.87 | 0.8 | 0.75 |
| \( c_3 \)        | 0.71 | 0.6 | 1 |
| \( c_4 \)        | 0.78 | 0.71 | 0.85 |

(6)

The final score matrix is

\[
S_{4x1} = C'_{4x3} W_{3x1} = (0.96, 0.83, 0.71, 0.77)^T,
\]

so we have \( TPL_{T_i} = [W_1, W_2, W_3, W_4]^T \).

The above calculation methods should be implemented in a smart contract on blockchain to guarantee the automatic generation of preference lists in task assignment stage. Workers are required to provide their preferences for tasks and the relative weights during user registration, which are also recorded in the smart contract. Similarly, for each task, the publisher gives the preferences for workers and the relative weight in the task requirements when publishing. As the above method, the smart contract receives the preference attributes of all the tasks and the requested workers to help generate their preference lists.

4.2. Preference-Based Task Distribution (PTD) Mechanism. Compared with the existing task distribution mechanisms that only focus on the demands of the publisher, the preference-based task distribution (PTD) mechanism considers both tasks’ and workers’ preferences put forward.

PTD is run on the smart contract with the saved TPL and WPL. As illustrated in Algorithm 1, PTD traverses every task given in the WPL firstly. For each worker \( W_i \) whose status is undistributed, with his first rank task \( T_j \), he receives his task completion proposal. If \( T_j \) still exists in undistributed worker places, \( W_i \) is accepted temporarily in the temporary distributed list TempDistributed\(_j\). If not, PTD looks up TempDistributed\(_j\) to find whether or not there exists \( W'_i \) ranking lower than \( W_i \) in TPL\(_j\). Replace \( W'_i \) with \( W_i \) upon finding successfully. PTD finishes running with all workers distributed.

5. Solution Verification Method

The quality of crowdsourcing solutions varies due to the difference of workers’ abilities, so the solution assessment is among one of the most challenging issues in crowdsourcing applications. We establish an elastic blockchain-based reputation mechanism based on the participants’ historical crowdsourcing behaviors and propose two optional solution evaluation methods.

Many related studies propose to build a reputation mechanism according to workers’ historical performance by collecting and analyzing their behaviors to obtain the reputation value that accurately and objectively reflects the participants’ trustworthiness and stability. Unlike the existing reputation models interpreting the user’s historical behaviors as a series of discrete events without considering the rational continuity, we design an elastic reputation model that continuously motivates workers to make positive behaviors by back-citing historical factors. Besides, the linear cumulative reputation score is mapped to a reputation value by the Sigmod function with the overall growth showing a trend of first sharp and then mild, which matches the demands in crowdsourcing scenarios.

Compared with similar crowdsourcing schemes, applying the proposed reputation mechanism helps set the screening threshold for worker selection, increasing the
crowdsourcing quality. Based on the reputation mechanism, two proposed verification methods effectively ensure the fairness and impartiality of solution evaluation.

5.1. Reputation Mechanism. Task completion and solution evaluation are two crowdsourcing activities that affect users’ reputation in the system. It is crucial to note that a user may switch between several roles in different task cycles. Therefore, all users should be involved in the reputation measurement. By writing the calculation method of reputation mechanism into the smart contract, the dynamic calculation and automatic adjustment of the user’s reputation can be realized to avoid dishonest users from modifying without permission.

Given the characteristic of crowdsourcing, the reputation mechanism design should satisfy the following demands.

1. The users’ total reputation calculation should combine their historical behaviors. Users with a higher percentage of positive historical behaviors get a higher reputation reward after each positive behavior.

2. Negative behaviors should be strongly punished, and continuing negative behaviors leads to the reputation dropping dramatically.

3. To avoid the endless reputation growth that leads to centralized power, the growth shows a trend of slowing down momentum.

Specifically, the reputation model is divided into the task completion score model and the solution evaluation score model. The user’s total reputation combines the scores in these two aspects. The parameters involved in the reputation mechanism are defined in Table 2.

5.1.1. Task Completion Score Model. In the phase of solution verification, the solutions submitted by workers are assessed individually. The verification steps are different based on the task type. For TCA-type tasks, the solutions are judged as “right” or “wrong.” Due to the different abilities of individuals, the solutions of TUA are evaluated for the detailed scores varying between 0 and 100.

For TCA-type tasks, a worker gets a positive reward for reputation if the solution is concluded as “right” and negative as punishment if not. Similarly, for TUA-type tasks, a worker receives reputation reward if the solution’s final score exceeds the average and reputation punishment if lower. The calculation of rScore is

\[
\begin{align*}
\text{rScore}^P_{xk} & = \begin{cases} \\
\text{finalScore}_{xk} & \text{TUA}, \\
1 & \text{TCA}, \\
100 & \text{TUA}, \\
-1 & \text{TCA} \end{cases} \\
\text{rScore}^N_{xk} & = \frac{\text{finalScore}_{xk} - \text{ave}}{100}, \quad \text{TUA}, \\
\end{align*}
\]

5.1.2. Solution Evaluation Score Model. Each evaluator gives the score for a solution from two aspects, that is, integrity and quality. To avoid the collusions between evaluators and workers, several evaluators simultaneously participate in a task’s evaluation phase. The final score is computed after excluding all the outliers from malicious evaluators. A common way is to set thresholds for the standard deviation of all the integrity scores and quality scores. If the given score is beyond the threshold in any two aspects, it is tagged as an outlier, with all remains computing the final score. The calculation model of the integrity score and the quality score are

\[
\begin{align*}
\text{integrityScore}_{xk} & = \frac{\sum_{i=0}^{\infty} q_{yk} * R_y}{\sum_{i=0}^{\infty} R_y}, \\
\text{qualityScore}_{xk} & = \frac{\sum_{i=0}^{\infty} q_{yk} * R_y}{\sum_{i=0}^{\infty} R_y}. \\
\end{align*}
\]
Table 2: Symbol definition in reputation mechanism.

| Symbol                                  | Definition                                                                 |
|-----------------------------------------|---------------------------------------------------------------------------|
| \( i_{yk} \)                            | Integrity score of solution \( k \) given by evaluator \( y \)             |
| \( q_{yk} \)                            | Quality score of solution \( k \) given by evaluator \( y \)               |
| \( \text{integrityScore}_{yk} \)        | Integrity score of solution \( k \) submitted by worker \( x \)           |
| \( \text{qualityScore}_{yk} \)         | Quality score of solution \( k \) submitted by worker \( x \)             |
| \( \text{finalScore}_{yk} \)           | Final score of solution \( k \)                                           |
| \( \text{rScore}^{p}_{yk} \)           | Score reward for participating in crowdsourcing activity \( y \)          |
| \( \text{rScore}^{N}_{yk} \)           | Score punishment for participating in crowdsourcing activity \( y \)       |
| \( S_{U}^{i} \)                         | Total reputation score of user \( U \) by the end of activity \( i \)      |
| \( R_{U}^{i} \)                         | Total reputation value of user \( U \) by the end of activity \( i \)      |

where \( ec \) is the number of evaluators that participating in the consensus of the final score. Each score from the evaluator \( y \) is weighed by their reputation value \( R_{y} \).

Combining the integrity score and quality score, \( \text{finalScore}_{yk} \) is calculated as

\[
\text{finalScore}_{yk} = \text{integrityScore}_{yk} \cdot \text{qualityScore}_{yk}. \tag{9}
\]

The reputation score obtained for evaluator \( y \) by evaluating the solution \( k \) is calculated as

\[
\text{rScore}^{p}_{yk} = \frac{20 - \left| \text{qualityScore}_{yk} - q_{yk} \right| - \left| \text{integrityScore}_{yk} - i_{yk} \right|}{20} \cdot j, \tag{10}
\]

\[
\text{rScore}^{N}_{yk} = -\text{rScore}^{p}_{yk},
\]

where \( \varphi \in [0, 1] \) is to differ the reputation score reward of task completion from solution evaluation.

5.1.3. Reputation Value Calculation Model. In view of the influence of historical reputation on current growth, the total reputation score is calculated as

\[
S_{U}^{i} = \begin{cases} 
0, & i = 0, \\
S_{U}^{i-1} + \frac{p}{p + q} \cdot \text{rScore}^{p}_{y} + \alpha \cdot \text{rScore}^{N}_{y}, & i > 0,
\end{cases} \tag{11}
\]

where \( q \) and \( p \) are, respectively, the number of negative and positive behaviors by the end of crowdsourcing activity \( i \). \( \alpha (\alpha > 1) \) is applied to adjust the severity of punishment.

As the continuous accumulation of reputation scores leads to the centralized power, a user’s reputation score is mapped into the reputation value with a certain range. The calculation model of the reputation value is

\[
R_{U}^{i} = \frac{1}{1 + e^{-S_{U}^{i}}}. \tag{12}
\]

In equation (11), the reputation value is reshaped by the Sigmod function. As in Figure 4, the reputation value increases fast first and flat afterward within \([0, 1]\), which meets the requirements of the reputation mechanism design, to precisely differentiate defamed ones and generalize reputable ones. The initial reputation is set 0.5 when a user first enters the system, that is, the neutral position with no historical evidence.

Given the time relevance of users’ reputation value, a history factor \( \lambda (\lambda < 0.5) \) is applied to increase the recent reputation value in the proportion of the total. Let \( R_{U}^{i} \) be the total reputation value of the user \( U \) by the end of the crowdsourcing activity \( i \) and \( r_{U}^{i} \) be the obtained reputation value of the user \( U \) by involving in the crowdsourcing activity \( i \). The total reputation value is calculated as

\[
R_{U}^{i} = \begin{cases} 
0.5, & i = 0, \\
\lambda \cdot R_{U}^{i-1} + (1 - \lambda) \cdot r_{U}^{i}, & i > 0.
\end{cases} \tag{13}
\]

Setting the bottom line of the reputation value \( \delta (\delta < 0.5) \) helps to resist the persistent negative behaviors. Suppose a user still has negative behaviors after his or her reputation value is below \( \delta \). In that case, all the task deposits are deducted, and the user is unable to participate in any crowdsourcing activity in future. The only way to restore the reputation is to pay double deposits to enter the reputation restoration phase. At that time, users have to make positive contributions continuously to restore the reputation slowly. Only after reaching \( \delta \), can the user’s reputation value be measured normally. If the user still has negative behaviors during this period, all the deposits will be deducted, and the user is removed from the system as a malicious user. The calculation of reputation restoration is

\[
R_{U}^{i} = R_{U}^{i-1} + \beta \cdot r_{U}^{i}, \tag{14}
\]

where \( \beta \) is the factor of restoration used to pace the reputation growth. It is worth noticing that the behavior of tampering the solution mentioned in Section 3 leads to the
deduction of all the deposits together with the reputation value.

5.2. Two Alternative Solution Verification Methods Based on Reputation

5.2.1. TCA Solution Verification Based on Majority Voting.
This type of task is a particular case in crowdsourcing with a deterministic answer. The integrity comparison naturally becomes the effective verification method, which is called majority voting. Specifically, the task’s smart contract collects all the solutions and makes comparisons between them. The identical solution submitted by the majority is considered correct. However, unlike the simple majority voting, the solutions are weighed by the reputation value of their submitters, which improves the fairness of the final answer.

Take task ID, for example; suppose the task is completed by the worker set \( W = \{W_1, \ldots, W_N\} \) with the corresponding reputation value set \( R_{\text{ID}} = \{R_{W_1}, \ldots, R_{W_N}\} \). The solution submitted by worker \( W_j \) \((j \in [1, N])\) is weighed by \( w = R_{W_j} / \sum_{i=0}^{N} R_{W_i} \). Ultimately, the correct answer is determined in this way as mutual consensus among all the workers. By implementing this verification method into a smart contract, the solution verification can be automatically carried out in a specific phase.

5.2.2. The Distributed Consensus of the TUA Solutions.
On account of the individual differences, the TUA solutions are assessed by several evaluators to reach a distributed consensus. Setting the reputation threshold for workers when requesting the tasks brings new users’ entry barriers to the system. Involving in solution evaluation is the opportunity for new users to increase reputation value quickly.

More specifically, any worker, except those who have completed the task, can send the request to the contract after the deadline of solution submission. All candidate evaluators are sorted according to their reputation value and divided into multiple intervals of the number of evaluators required for the task, ensuring that at least one user exists in each interval. As shown in Figure 5, one worker is randomly selected from each interval to be the evaluator.

The advantages of evaluators’ selection above can be summarized as below:

1. Allowing workers with different reputations to be evenly distributed among candidate evaluators, so as to avoid workers with high reputations having supreme power in assessment
2. Giving new users some opportunities to involve in the evaluation work and improve their reputation, to avoid the emergence of system entry barriers
3. Reducing the possibility that colluding workers are selected simultaneously and improving the fairness and accuracy of evaluation results

Every solution is graded by all the evaluators from both the integrity and quality aspects as described in Section 5.1, and the final score is calculated by combining all the scores from evaluators.

6. Performance

6.1. Theoretical Security Analysis. Considering the mainstream attacks in the crowdsourcing system, this section summarizes the proposed scheme and carries out security analysis from the theoretical level. The comparisons with other crowdsourcing schemes in references are shown in Table 3.

6.1.1. Free-Riding Attack. In the solution submission stage, workers submit the commitment of the solution instead of the solution itself. A solution’s commitment is associated with the worker submitter’s blockchain address which ensures the uniqueness of the submitter. Workers publish the solution hash and the verification key after the submission deadline. All nodes can verify the commitment, so there is no possibility for dishonest workers to carry out the free-riding attack.

6.1.2. False-Reporting Attack. Only when a publisher has pledged the task rewards will the task be considered valid in the system. The two proposed solution assessment methods are all decentralized, and a publisher does not partake in the evaluation. Compared with the existing centralized scheme, the proposed scheme can effectively resist the false-reporting attack.

6.1.3. Collusion Attack. Collusion mainly occurs between publishers and workers or evaluators and workers to influence the solution verification or reward distribution process. In the scheme presented, neither the publisher nor the worker participates in the solution verification and reward distribution, so they cannot influence the fairness of crowdsourcing through collusion. Therefore, the scheme can effectively resist collusion attacks.

6.1.4. Distributed Denial-of-Service (DDoS) Attack. The decentralized architecture constructed by blockchain has the characteristics of a typical distributed system. Even if there are partial failure nodes, it may not have a fatal impact on the overall service of the system, so the proposed scheme can resist DDoS attacks elastically.

6.1.5. No Trusted Third Party. The fair exchange between the solutions and rewards can be fulfilled without any trusted third party. The incentive mechanism is recorded in the blockchain after the task announcement that cannot be modified. The difficulty of changing the incentive mechanism is undoubtedly equivalent to destroying the essential characteristics of the blockchain.

To conduct performance tests and analysis, we have developed a prototype system based on Ethereum with image rendering as the experimental crowdsourcing task. PTD mechanism and solution verification methods are all...
implemented. All the experiments are conducted on a PC (Intel Core i7-10710U CPU with 4.7GHz, 16G RAM, and Windows 10 OS) using Solidity 7.0 and Web3 1.0.

The experiment environment has 100 registered workers and published multiple tasks varying from 20 to 100, with each task requesting 3 workers to complete. Each worker can only be associated with one task simultaneously. The task experiments dataset is selected from CIFAR-10 to simulate the image marking tasks. The dataset consists of 60,000 32×32 color images, which are composed of 10 categories such as aircraft, cars, and birds.

6.2. Task Distribution Mechanism Evaluation. To ensure the fair experiment background, the attributes used in the generation of the preference list of all the workers are the reputation value, expected rate of reward, and the number of completed tasks. Meanwhile, the preference list for each worker is generated randomly to guarantee the authenticity of experiments.

The mechanism performance is evaluated from two aspects: (1) percentage of distributed tasks and (2) satisfaction of crowdsourcing participants. To better measure the performance of the proposed PTD, publisher determined distribution (PDD) and random task allocation (RTA) mechanisms are chosen to be the comparisons.

As illustrated in Algorithm 2, the publisher determined distribution (PDD) mechanism relies on the standards set by the publisher upon task publication which simply considers the quality of the task achievements without the workers’ preference.

Random task allocation (RTA) mechanism randomly allocates specific tasks to unallocated workers, which only depends on the workers’ status.

As presented in Figure 6, we test the overall task distribution percentage of the proposed PTD, PDD, and RTA under different ratios of task number over worker number to better intuitively display the distribution effects of different mechanisms. As the worker number needed for each task is set to 3, PTD and RTA have the 100% task distribution.
percentage when the ratio is 0.2 for the sufficient supply of workers. With the increase of the ratio of tasks to workers, the task distribution percentage decreases due to the constant number of workers. RTA wins the comparison as its random task distribution process makes the tasks being distributed maximum. The proposed PTD distributes tasks considering the workers’ preferences which allow workers to complete their relatively preferred tasks and let tasks choose relatively suitable workers. This also has the tasks distributed maximum, making the percentage of task distribution the same to RTA. The preference of task and the limit of one task per worker make PDD have a lower task distribution percentage than PTD and RTA. The result proves that the proposed PTD mechanism helps guarantee the high percentage of task distribution.

The satisfaction is considered from both sides of the tasks and the workers, which is related to their preference lists. That means workers’ degree of satisfaction (DoS\(_W\)) is calculated as

\[
\text{DoS}_W = \sum_{T=1}^{N} \sum_{W=1}^{n_W} \frac{1}{\text{rank}_{WPL}(T)},
\]

where \(N\) is the number of available crowdsourcing tasks, \(n_W\) is the number of distributed workers, and \(\text{rank}_{WPL}(T)\) is the rank of task \(T\) in the preference list of the worker \(W\). When task \(T\) happens to be the most preferred task in \(WPL_W\), DoS\(_W\) achieves maximum with 1.

The tasks’ degree of satisfaction (DoS\(_T\)) is calculated as

\[
\text{DoS}_T = \sum_{T=1}^{N} \frac{\text{num}_{TPL}(T)}{n_T},
\]

where \(n_T\) is the number of workers needed for task \(T\), num\(_{TPL}\) is the number of workers assigned to task \(T\) who are also in the top \(N_W\) of the preference list (\(N_W\) is the number of needed workers for task \(T\)), and DoS\(_T\) achieves maximum with 1 when the distributed result is the same as top \(N_W\) of TPL\(_T\).

**Algorithm 2**: Publisher determined distribution (PDD) mechanism.

![Algorithm 2: Publisher determined distribution (PDD) mechanism.](image)

**Figure 6**: Comparisons of PTD, PDD, and RTA in task distribution percentage under different ratios of the task number over the worker number.
Although the RDA mechanism has not too much to do with preference, we still calculate DoS according to the task matching results for comparisons. As shown in Figure 7(a), DoS\textsubscript{W} increases with the increase of task numbers for the workers participating in the tasks; increase for plenty of available tasks brings workers the opportunity to complete their relatively preferred task. The results show DoS\textsubscript{W} of proposed PTD is higher than the two other mechanisms, which means PTD really helps improve the satisfaction of workers. Figure 7(b) presents the comparisons of DoS\textsubscript{T} for three mechanisms. DoS\textsubscript{W} of proposed PTD is the highest and increases gradually with the increase of task number, but the growth rate gradually slows down due to the constant number of workers. Although PDD takes task preference as the only indicator to select workers, overall satisfaction is limited by the percentage of task allocation, so task satisfaction does not achieve good results. Due to the random selection of workers, although the satisfaction of RTA increases slightly with the increase of the number of tasks, the effect is still not ideal.

6.3. Reputation Mechanism Evaluation. In order to verify whether the reputation mechanism is in line with the design goal, we draw the change curve of a typical worker’s reputation value who participates in the image rendering task several times. The relevant parameters of the reputation mechanism in the experiments are set as follows: \( \alpha = 6, \lambda = 0.3, \delta = 0.1, \) and \( \beta = 0.01. \)

Assuming that the worker has completed 36 tasks in total, with the 7th, 13th, 18th, 19th, and 20th solutions judged to be “wrong,” the result in Figure 8 shows that the growth rate of worker’s reputation value is fast at first and then slows down. If there exists a wrong answer, the worker needs at least 4 correct answers to make the reputation value back to the original that satisfies the design requirements. After completing the 18th task, the user continuously makes mistakes, which led to the rapid decline of the reputation value. After the 19th mistake, the reputation value drops to below the bottom line, so the 20th mistake makes the reputation value be forced to 0. At this point, the worker’s task deposits are fully deducted. If no measures are taken, the worker will not be able to participate in any crowdsourcing activity. In the experiment, the worker immediately pays the double deposits and enters the reputation restoration phase at the 21st task. After 10 positive behaviors, the reputation value slowly increases and finally exceeds the bottom line.

The experimental results show that the negative behavior of users lead to the rapid decline of the reputation value whose recovery is slow. If a user continues to make negative behaviors, the reputation value will drop rapidly and eventually fall below the bottom line and then be set to 0. Meanwhile, the setting of the reputation restoration phase restrains the frequency of their negative behaviors, which realizes the quality control of crowdsourcing.

6.4. Performance Evaluation

6.4.1. Off-Chain Computing Overhead. Off-chain computing is computations performed locally by users. In order to test the off-chain computing overhead from the perspective of users, we select two crowdsourcing schemes for comparison. Zhang et al. and Li et al. [9, 12] are the typical representatives of the centralized crowdsourcing scheme and the blockchain crowdsourcing scheme, respectively. The basic operation symbols of off-chain computing are shown in Table 4, and the execution time of different operations is sorted as \( T_{AV} < T_{H} < T_{AS} \approx T_{AE} \approx T_{AD} < T_{CG} < T_{EC}. \)

The results of the off-chain computing overhead comparison are shown in Table 5. Here, “−” indicates that the computational overhead is negligible and “\( \approx \)” means that the scheme does not have this user role. The results suggest that the users in [9, 10] do not have any additional off-chain calculation. Still, this kind of the centralized crowdsourcing scheme cannot meet the demand of practical application on account of the defects in security, fairness, reliability, and other aspects. In [12], the user’s off-chain computing mainly comes from the preparation before the solution submission and the relevant verification work before the solution evaluation. The primary operations include using an asymmetric encryption algorithm to generate the encrypted solution, using a signature algorithm to ensure the uniqueness of the submitter, and using a hash algorithm to ensure the integrity of the off-chain storage. Although this approach avoids the free-riding attack to some degree, only the evaluator knows whether the worker has carried out the free-riding attack due to the lack of adequate evaluation means. In case of the collusion between the evaluators and the dishonest workers, the free-riding attack will still occur. Our scheme uses the commitment protocol to record the commitment value of the solution hash in the open blockchain ledger. After disclosing the solution hash, all nodes can perform commitment verification, and there is no possibility of conducting the free-riding attack or collusion attack. Even if the commitment generation and validation operations add additional computing overhead to the users, the security benefits far exceed the computing itself.

6.4.2. On-Chain Time Overhead. We carry out several experiments on the prototype system to evaluate the on-chain time overhead in different crowdsourcing phases. Clarar.IO is a professional library of 3D models, from which 200 model files are downloaded for the test. We register 1 publisher and 10 workers in the system. Each task demands 5 workers to complete. 10, 20, 40, 80, and 160 model images are randomly selected from the downloaded rendering model files to form crowdsourcing tasks of 5 different task sizes, labeled as Task_10, Task_20, Task_40, Task_80, and Task_100, respectively. For the test task belonging to the TCA type, the solution is assessed by smart contract in the first assessment way.

Regardless of the time overhead of off-chain computing and upload or download from IPFS, we only compute the on-chain execution time required in different crowdsourcing phases. The experiment results are presented in Table 6. The time consumption of the four phases of task announcement, task allocation, solution upload, and reward
allocation is not affected by the task size with the time cost at the millisecond level. The average time costs are 64.75 ms, 45.02 ms, 18.52 ms, and 61.70 ms, respectively. In the case of small task size, the time required for assessment does not change variably, which basically remained at about 120 ms. As the task size growing, the time required gradually increases for the number of solutions needed assessment increase sharply.

To reflect the delay of task crowdsourcing, we distribute task_20 to 5 workers and calculated the consensus periods when the task goes through the whole crowdsourcing cycle in the proposed system. The experiment is carried out in the system prototype based on Ethereum with block gas limit set to $4 \times 10^6$ gas, and the average gas requirements for different operations are counted as in Table 7, where task_20 experiences 5 consensus periods.
7. Conclusion

Aiming at the severe defects of the centralized crowdsourcing systems, such as single-point failure, lack of fairness, and privacy leakage, this study presents a general crowdsourcing framework on distributed serverless infrastructure with reliability, security, and fairness. In the design of the crowdsourcing process, we combine commitment protocol to ensure the secure on-chain interaction of solutions and effectively resist the free-riding attack. To improve the percentage of task distribution and the satisfaction of crowdsourcing participants, the preference-based task distribution (PTD) mechanism is proposed. By establishing the reputation mechanism, all crowdsourcing behaviors are audited and evaluated for consistency and trustworthiness. Besides, we alternatively introduce two solution assessment methods to overcome the inherent evaluation issues in crowdsourcing. The theoretical security analysis and empirical performance benchmark prove that the proposed framework with distributed task assignment and solution verification has more vital advantages than existing schemes.

As the crowdsourcing tasks become more diversified and dynamic, the proposed blockchain-based crowdsourcing system still has some further research points, which we are planning to finish shortly:

1. The system has advantages pass the theoretical security analysis, and we will further prove its reliability from the practical perspective by testing in various application scenarios
2. We will optimize the proposed reputation mechanism to prevent the persistent reputation accumulation for premeditated attacks
3. The verification strategy for TCA will be intensely studied by constructing a particular task scheduling mechanism to realize the secure on-chain cross-validation with the application of technologies such as Zero-Knowledge Proof 4, 7 (ZKP).

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (Grant nos. 61802186 and 61472189).

References

[1] J. Howe, “The rise of crowdsourcing,” Wired magazine, vol. 14, no. 6, pp. 1–4, 2006.
[2] A. Ghezzi, D. Gabelloni, A. Martini, and A. Natalicchio, “Crowdsourcing: a review and suggestions for future research,” International Journal of Management Reviews, vol. 20, no. 2, pp. 343–363, 2018.
[3] K. Wazny, “"Crowdsourcing” ten years in: a review,” Journal of global health, vol. 7, no. 2, pp. 1-13, 2017.
[4] O. A. Acar, “Motivations and solution appropriateness in crowdsourcing challenges for innovation,” Research Policy, vol. 48, no. 8, Article ID 103716, 2019.
Y. Gong, S. Engelenburg, and M. Janssen, “A reference architecture for blockchain-based crowdsourcing platforms,” *Journal of Theoretical and Applied Electronic Commerce Research*, vol. 16, no. 6, pp. 937–958, 2021.

Z. Zheng, S. Xie, H. N. Dai, X. Chen, and H. Wang, “Blockchain challenges and opportunities: a survey,” *International Journal of Web and Grid Services*, vol. 14, no. 4, pp. 352–375, 2018.

B. K. Mohanta, S. S. Panda, and D. Jena, “An overview of smart contract and use cases in blockchain technology,” *Computing, Communication and Networking Technologies*, vol. 10, pp. 1–4, 2018.

D. D. Green, “Fueling the gig economy: a case study evaluation of upwork.com,” *Manage Econ Res J*, vol. 4, no. 1, Article ID 3399, 2018.

X. Zhang, G. Xue, R. Yu, D. Yang, and J. Tang, “Keep your promise: mechanism design against free-riding and false-reporting in crowdsourcing,” *IEEE Internet of Things Journal*, vol. 2, no. 6, pp. 562–572, 2015.

Y. Zhang and M. Van der Schaar, “Reputation-based incentive protocols in crowdsourcing applications,” in *Proceedings of the 31st IEEE Annual International Conference on Computer Communications*, pp. 2140–2148, Orlando, UAS, March 2012.

C. Tanas, S. Delgado-Segura, and J. Herrera-Joancomartí, “An integrated reward and reputation mechanism for MCS preserving users' privacy,” in *Proceedings of the 10th International Workshop on Data Privacy Management, and Security Assurance*, pp. 83–99, Vienna, Austria, September 2015.

M. Li, J. Weng, A. Yang et al., “Crowdbc: a blockchain-based decentralized framework for crowdsourcing,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 30, no. 6, pp. 1251–1266, 2018.

S. Zhu, Z. Cai, H. Hu, Y. Li, and W. Li, “zkCrowd: a hybrid blockchain-based crowdsourcing platform,” *IEEE Transactions on Industrial Informatics*, vol. 16, no. 6, pp. 4196–4205, 2019.

R. Canetti and M. Fischlin, “Universally composable commitments,” in *Proceedings of the 21st Annual International Cryptology Conference*, pp. 19–40, California, USA, August 2001.

Q. Zhang, Y. Gan, L. Liu, X. Wang, X. Luo, and Y. Li, “An authenticated asymmetric group key agreement based on attribute encryption,” *Journal of Network and Computer Applications*, vol. 123, no. 123, pp. 1–10, 2018.

Y. Liu, C. Cheng, J. Cao, and T. Jiang, “An improved authenticated group key transfer protocol based on secret sharing,” *IEEE Transactions on Computers*, vol. 62, no. 11, pp. 2335–2336, 2012.

K. Sharpe Wessling, J. Huber, and O. Netzer, “MTurk character misrepresentation: assessment and solutions,” *Journal of Consumer Research*, vol. 44, no. 1, pp. 211–230, 2017.

Y. Lu, Q. Tang, and G. Wang, “ZebraLancer: private and anonymous crowdsourcing system atop open blockchain,” in *Proceedings of the 38th IEEE International Conference on Distributed Computing Systems*, pp. 2–6, Vienna, Austria, July 2018.