CrowdBC: A Blockchain-based Decentralized Framework for Crowdsourcing

Ming Li, Jian Weng, Anjia Yang, Wei Lu,Yue Zhang, Lin Hou, Jia-Nan Liu, Yang Xiang, Robert H. Deng

Abstract—Crowdsourcing systems which utilize the human intelligence to solve complex tasks have gained considerable interest and adoption in recent years. However, the majority of existing crowdsourcing systems rely on central servers, which are subject to the weaknesses of traditional trust-based model, such as single point of failure. They are also vulnerable to distributed denial of service (DDoS) and Sybil attacks due to malicious users involvement. In addition, high service fees from the crowdsourcing platform may hinder the development of crowdsourcing. How to address these potential issues has both research and substantial value. In this paper, we conceptualize a blockchain-based decentralized framework for crowdsourcing named CrowdBC, in which a requester's task can be solved by a crowd of workers without relying on any third trusted institution, users' privacy can be guaranteed and only low transaction fees are required. In particular, we introduce the architecture of our proposed framework, based on which we give a concrete scheme. We further implement a software prototype on Ethereum public test network with real-world dataset. Experiment results show the feasibility, usability and scalability of our proposed crowdsourcing system.

Index Terms—Decentralized framework, crowdsourcing, blockchain, smart contract.

1 Introduction

Over the past few years, crowdsourcing has gained considerable interest and adoption since it is coined in 2006 by Jeff Howe [1]. It is a distributed problem-solving model through an open call for solutions. Nowadays, many large companies choose crowdsourcing as a problem-solving method, ranging from web and mobile development to t-shirt designs. There are numerous famous crowdsourcing applications such as Upwork [2], Amazon Mechanical Turk [3] and UBER [4]. We can expect that this field will change the working style of people significantly.

The human intelligence-based crowdsourcing consists of three groups of roles: requesters, workers and a centralized crowdsourcing system (Fig. 1). Requesters submit tasks which are challenging for computers but easy for human to complete through the crowdsourcing system. A set of workers who are interested in this task compete and submit solutions to the crowdsourcing system, while requesters will then select a proper solution (usually the first or the best one that solves the task) and grant the corresponding workers the reward. Taking the current world's largest freelancer marketplace, Upwork, for example, it requires "clients" (requesters) to deposit a milestone payment into the escrow account before work begins. Then "clients" could interview or hire "freelancers" (workers) to design or write. "Freelancers" who focus on the area of expertise compete for the job and the winners will obtain the reward.

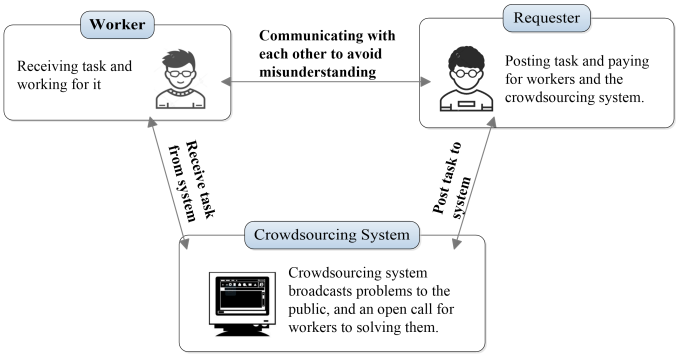

Worker
Receiving task and working for it
Communicating with each other to avoid misunderstanding
Requester
Posting task and paying for workers and the crowdsourcing system.
Receive task from system
Post task to system
Crowdsourcing System
Crowdsourcing system broadcasts problems to the public, and an open call for workers to solving them.

Fig. 1: The system model of traditional crowdsourcing.

However, despite the prosperity of the crowdsourcing systems, they are subject to the weaknesses of traditional trust-based model, which brings about some inevitable challenges. First, traditional crowdsourcing systems are vulnerable to DDoS attacks, remote hijacking and mischief attacks, which makes the services unavailable. Elance and oDesk, operated by Upwork presently, downed services for many workers due to be hit by DDoS attacks in May 2014 [5]. Second, the majority of crowdsourcing systems run business on a centralized server, which suffers from single point of failure inherently. In April 2015, a service outage emerged due to hardware failure in Uber China, which caused passengers can't stop the order at the end of services [6]. Third, user's sensitive information (e.g. name, email address and phone number) and task solutions are saved in the database of crowdsourcing systems, which has the risk of privacy disclosure and data loss. For example, one of the most prevalent crowdsourcing systems Freelancer [7] was reported to breach the Privacy Act for uncovering a user's true identity which contains IP addresses, active account and dummy accounts by Office of the Australian

- *Ming Li, Jian Weng, Anjia Yang, Yue Zhang, Lin Hou and Jianan Liu are with the College of Information Science and Technology and the College of Cyber Security, Jinan University, Guangzhou 510632, China. Jian Weng is the corresponding author. E-mail: limjnu@gmail.com, cryptjweng@gmail.com, anjiayang@gmail.com*
- *Wei Lu is with the School of Data and Computer Science, Guangdong Key Laboratory of Information Security Technology, Sun Yat-sen University, Guangzhou 510006, China.*
- *Yang Xiang is the Dean of Digital Research and Innovation Capability Platform, Swinburne University of Technology;*
- *Robert H.Deng is with the School of Information Systems, Singapore Management University.*

Information Commissioner (OAIC) in December 2015. Fourth, when requesters and workers are in dispute, they need help from the crowdsourcing system to give a subjective arbitration, which may lead to a behavior known as “false-reporting” [8]. Lastly, crowdsourcing companies are interested in maximizing their own benefits and require requesters paying for services, which in turn increase user’s costs. Currently, most of the crowdsourcing systems could demand a sliding services fee for 5% to 20% [2].

There have been many works to deal with part of the above mentioned issues in crowdsourcing systems. Encryption and differential privacy (DP) are used to protect data privacy [9], [10], [11], [12], [13]. Reputation-based mechanisms are proposed to address “false-reporting” and “free-riding” behavior [14], [15], [16]. Distributed architectures are designed to prevent single point of failure and bottleneck problem [17], [18]. However, the majority of these researches are built on the traditional triangular structure crowdsourcing models which suffer from breakdown of trust. Up to now, none of existing works has solved all of the above issues simultaneously. Thus, this research is motivated by this: *Can we design a decentralized crowdsourcing system with reliability, fairness, security and low services fee?* To answer this question, we propose a blockchain-based decentralized framework for crowdsourcing. The framework has many advantages such as increasing user security and service availability, enhancing the flexibility of crowdsourcing with Turing-complete programming language and lowering cost. Therefore, our framework has the potential to disrupt the traditional model in crowdsourcing. In a nutshell, our specific **contributions** are in the following:

- We conceptualize a blockchain-based decentralized framework for crowdsourcing named CrowdBC, which does not depend on any central third party to accomplish crowdsourcing process. CrowdBC guarantees privacy by allowing users to register without true identity and storing encrypted solutions in the distributed storage. Each identity makes a deposit before participation, which can effectively thwart many attacks such as DDoS, Sybil and “false-reporting” attacks. Moreover, users don’t need to pay the costly service fees to traditional crowdsourcing platform anymore, only required to pay a small amount of transaction fees. Our framework also enhances the flexibility of crowdsourcing by using Turing-complete programming language to depict complex crowdsourcing logics.
- We present a concrete scheme based on the proposed framework. Smart contract is used to perform the whole process of crowdsourcing task which contains task posting, task receiving, reward assignment, etc. We introduce three standard smart contracts in the scheme: User Register Contract (URC), User Summary Contract (USC), Requester-Worker Relationship Contract (RWRC), by which crowdsourcing functionalities can be achieved such as posting and receiving a task without relying on any central authority. In particular, compared with the traditional systems, the most useful feature lies in the evaluation of tasks to be completed via smart contract rather than a subjective third party. We expect that this construction will be proved quite impactful and useful in practice.
- We implement the proposed scheme to verify the feasibility through a software prototype based on Ethereum public test network with real-world dataset. Experiment results show the usability and scalability of our proposed crowdsourcing system. Furthermore, we illustrate a discussion of future improvements to this scheme.

The remainder of the paper is organized as follows. In section 2, we present the related work. The preliminaries of blockchain and smart contract are given in section 3. In section 4 we present the system model, threat model, security assumptions and technical challenges. In section 5, the description of our proposed framework is given. Next, we present a concrete crowdsourcing scheme under the framework and analyze the security properties in section 6. A series of experiments are demonstrated in section 7 and finally we conclude and discuss the future work of the paper in section 8.

2 Related work

Research on crowdsourcing has become an emerging trend with the explosive growth of the internet and mobile devices. We mainly review some state-of-art works in three primary areas: centralized crowdsourcing system, distributed crowdsourcing system and blockchain-based crowdsourcing system.

Centralized crowdsourcing system. Several crowdsourcing systems are developed in a centralized way [2], [3], [19], [20]. The crowdsourcing services, like worker selection, incentive mechanism and truth discovery, are provided by these centralized crowdsourcing systems. Upwork [2] and WAZE [19], as two famous crowdsourcing platforms, allow requesters to efficiently hire workers to obtain task solutions (e.g., traffic jams, accidents in WAZE). However, they required users’ detailed information (e.g., WAZE and Freelancer [20]) and stored user information, task data in the centralized platform, which may suffer from privacy leakage [7], DDoS/Sybil attacks [5] and the issue of single point of failure [6]. [8] proposed auction-based mechanisms EFT and DFT, [14] proposed reputation-based incentive mechanism to tackle free-riding and false-reporting attacks in crowdsourcing, but their methods are based on the traditional three parities model, which is not relevant with our scheme. Besides, there exist a number of incentive mechanisms relying on crowdsourcing system to save the task reward, which has the risk of bankrupt inherently [21].

Distributed crowdsourcing system. There also exist several researches on designing the distributed crowdsourcing system. [18] presented D2 protocol to design a distributed crowdsourcing system in Delay Tolerant Network (DTN). The authors aimed to complete a computation task in a collaborative way and achieve the minimal makespan. [22] proposed a task allocation scheme by utilizing social relationship in crowdsourcing system. They concentrated on loading balancing in distributed model. [23] introduced an asynchronous and distributed task selection in mobile crowdsensing. While [18], [22] and [23] focused on task completion in a distributed way, they actually had a centralized system to provide services, which is not consistent with the requirement of our idea that builds the crowdsourcing system in a decentralized way.

Blockchain-based crowdsourcing system. [24] proposed CrowdJury which is a blockchain-based crowdsourcing application for court processing of adjudication. This is most related to our approach, but the details about the design of crowdsourcing protocols are not provided. [25] and [26] presented a blockchain-based crowdfunding which is a specific type of crowdsourcing. [27] raised an alternative protocol employing blockchain to tackle the issue of small-value transactions in crowdsourcing. In addition, the research on blockchain-based crowdsourcing has also gained considerable interest in industry recently, such as microwork [28], Gems [29]. The above mentioned works are limited to their specific applications (i.e., CrowdJury with court adjudication), whilst in comparison, we conceptualize a blockchain-based decentralized framework with much broader goals, such as providing a direction for system designers to design a class of decentralized protocols in crowdsourcing.

3 Background

3.1 Blockchain

Bitcoin [30] is the first idea of a decentralized currency which has attracted lots of attentions. A set of time-ordered transactions are recorded in files called "blocks". Each block contains the hash value of the previous block, and they eventually form a hash chain called "blockchain" which is essentially a public, immutable and ordered distributed ledger. Users offer computing resources to compete for the right of recording transactions into blockchain, and the winner will be rewarded with coins and transaction fee. Blockchain technology provides a new direction for us to reduce the role of the middleman in current society [31]. And we can easily associate blockchain with the financial sector (e.g. Bitcoin), but the innovative potential of blockchain applications is much more than this. The applications in different areas, such as Micropayment schemes [32], naming and storage system [33] and health records sharing [34], are based on blockchain technology.

Transaction: Defined as a data structure, transaction consists of three segments: inputs, outputs and digital signature. For a valid coin transaction, the input must be an unspent of a previous transaction. And all transactions during a period of time are linked together as a structure (e.g. Merkle tree) and filled into a block by a miner. Once the block is confirmed, these transactions will not be able to be changed anymore.

Consensus Protocol: Consensus protocol aims to determine which miner's block will be appended on the blockchain. Generally, the miner gets the recording right by affording a valid proof which can be confirmed correct by other miners, such as a challenging computational puzzle in Bitcoin. In particular, the consensus blockchain is also the longest chain, which refers to the largest work to be produced. There exist several kinds of consensus protocol, such as proof of work (PoW) [30] and proof of stake (PoS) [35].

Network: Blockchain uses a peer-to-peer (P2P) network, which is a distributed application architecture [36]. Unlike the traditional Client/Server mode, Nodes in P2P network have equal privilege without a central coordination by servers or stable hosts, i.e., they are both suppliers and consumers of resources.

Blockchain Paradigm: Blockchain can be viewed as a transaction-based state machine [37]. Each state includes information like a nonce, account balances, data expressing information of the physical world, etc. It's updated from a genesis state to a final state after each transaction. In this scheme, we focus on smart contract execution and state transition. Let's give a description of blockchain paradigm by describing a simplified transaction that depicts a smart contract execution here. A transaction Tx could activate the code execution of smart contract. Then, a valid state transition from σ to σ_{t+1} via Tx is denoted as: $\sigma_{t+1} = F(\sigma, Tx)$, where F refers to arbitrary computation which is carried out by blockchain, and σ can store arbitrary state between transactions.

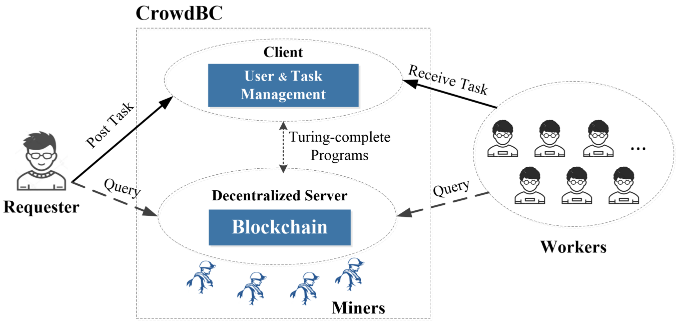


Fig. 2: The system model of CrowdBC.

3.2 Smart Contract

Smart contract, which refers to the Blockchain 2.0 space [38], is proposed by Nick Szabo in 1994 [39]. It depicts complex logics by programming common process into code and represents the implementation of contract-based agreement. It is essentially a self-executing digital contract in a secure environment with no intervention and verified through network peers. The main reason for difficult to realize smart contract before is that it's hard to find a secure environment which is decentralized, unalterable and programmable. The advent of blockchain technology could solve this problem perfectly. Currently, there exist several blockchain platforms supporting smart contract, two famous of which are Ethereum [37] and Hyperledger [40]. They are designed to run smart contract without frauds, downtime or any third party interference.

4 System Model

In this section, we present the system model and the workflow for blockchain-based crowdsourcing. Based on the system model, we formulate the threats and analyze the technical challenges. Before formulating the problem, we show the notations within this paper as in table 1.

4.1 Decentralized Crowdsourcing System Model

As shown in the Fig. 2, there exist four roles in the model of the proposed framework: Requester, Worker, CrowdBC Client, and Miner.

Requesters, identified by $\mathsf{R} = \{R_1, ..., R_i, ..., R_n\}$, post a task by transferring task descriptions into programs. To stimulate workers participating in and prevent any unfair event, a certain amount of reward v and and a monetary penalty π_R are required as the deposit (i.e., $deposit_R = v + \pi_R$) which cannot be redeemed before deadline.

Workers, identified by W=$\{W_1, ..., W_j, ..., W_m\}$, are the community who have certain skills and compete for tasks to get rewards. Worker W_j is selected depending on a tuple of reliability value $\{\beta_{W_j}, category_k(\varepsilon_k, \varphi_k)\}$, where $category_k(\varepsilon_k, \varphi_k)$ denotes that worker W_j has received the task of $category_k$ and submitted solutions to this category task for ε_k times and got φ_k high evaluation. We require that each worker makes a deposit π_{W_j} on blockchain to thwart DDoS and Sybil attack. Upon the evaluation of the solution, W_j is assigned with corresponding rewards v_R. Note that we define the solutions submitted by workers as $s = \{s_1, ..., s_j, ..., s_m\}$.

CrowdBC Client, served as a medium for R and W, is not controlled by any third party, just like Bitcoin Core in Bitcoin. It

TABLE 1: The notations of explanation.

Notation	Explanation
$R = \{R_1, ..., R_i, ..., R_n\}$	Set of requesters, $i = 1, ..., n$
$W = \{W_1, ..., W_j, ..., W_m\}$	Set of requesters, $j = 1, ..., m$
β_{W_j}	The reputation value of worker W_j
K_u^p, K_u^s, K_u^a	The public key, secret key and address of user u
$\mathsf{H}(M)$	The hash value of the message M
$\mathsf{Sign}(M)_{K_u^s}$	The digital signature on message M with secret key K_u^s
$\mathsf{Verify}(M)_{K_u^p}$	The function to check u's digital signature M with public key K_u^p
$coins(v)$	The virtual coin of value v
s_j	The submitted solution of worker W_j
$TI_{deadline}$	The task deadline which refers to the future block height
$TI_{confirm}$	The task confirmation time which refers requester should finish solution confirmation in the special block height
T	T refer to the task information

could run locally on user's personal computer. R and W reach to the agreement by transactions in CrowdBC.

Miners mainly add past transactions into a block and provide a valid proof to claim the block reward and transaction fees (i.e., mining). The security of the underlying blockchain is built on top of these miners. Note that R and W can also be viewed as miners if they participate in the mining work.

In our scheme, we require that R and W register to get the credentials before obtaining services from our system, i.e., $R_i = (K_{R_i}^p, K_{R_i}^s, K_{R_i}^a)$ and $W_j = (K_{W_j}^p, K_{W_j}^s, K_{W_j}^a)$ (private key is saved by users). Then, R_i can post a special category task T by utilizing CrowdBC Client which sends a transaction to the blockchain. There have many category of tasks (e.g., JAVA software development, logo design) which are pre-defined in program, $category = \{category_1, ...category_k\}$. The authenticated worker whose reliability value satisfies the minimum condition can receive the task. Using the Turing-complete programs in blockchain, W can reach agreements with R. Upon the evaluation of the solution, W are assigned with corresponding rewards.

4.2 Threat Models

The potential malicious requesters and workers have specifically different goals to maximize their own profits. We firstly define the threat model which illustrates potential threats and malicious behaviors as follows:

Malicious Requesters: Malicious requesters aim to collect useful solutions without losing the deposit, which is false-reporting attack in essential. To achieve this goal, they may misreport the evaluation solution as low level even if workers contribute high-quality of solution. In addition, they may even deny they have obtained the solutions. Besides, we require requesters make a deposit in our protocol, while they may benefit from not following or even breaking the protocol, which means that malicious requesters may attempt to create a fork chain after they receive the solutions.

Malicious Workers: Malicious workers attempt to obtain rewards without paying sufficient effort, which is free-riding attack. The same as the malicious requesters, they may benefit from not following or even breaking time-locked deposit protocol, which means malicious workers may try to create a fork chain if they receive low level evaluation. They may receive the task but not submit solutions on time, which could discourage requesters from participating in the crowdsourcing system. In particular, they may deny the low-quality solution because there does not exist a third party to audit it, which is a trust problem. Besides, since their reputation and expertise data play a vital role in verifying the qualification when receiving the task, they may improve these data with posting a task by themselves.

Malicious Miners: The malicious miners attempt to earn much virtual coins by forking a chain or colluding with malicious requesters or workers to break the normal execution of program on blockchain.

We formalize a security property as fully fraud resistant and give the definition as follows:

Definition. (Fully Fraud Resistant) The crowdsourcing system is fully fraud resistant if none of the following condition happens:

- Requesters can bring back their rewards for workers if they get effective solutions.
- Workers can redeem their deposits if they do not submit solutions on time or contribute low-quality solutions.

4.3 Security Assumption

Majority Honest Security: The security of crowdsourcing task in CrowdBC is related with the security of blockchain. Considering that requesters and workers can take part in mining, we assume that the attacker (including malicious requesters, workers and miners) cannot break the fundamental security of blockchain, namely, the attacker does not have the majority of power or resource to control the blockchain network and the majority of miners are honest. The network has low latency and messages are synchronous between honest miners.

Assume that there exist n total miners in crowdsourcing system (including requesters and workers who participate in maintaining the system as miners), and α be percentage of malicious miners who try to create a fork chain to their own advantage. Each miner tries to find a block (e.g., provides a POW solution) with probability p for every hash computation, and for the total number of q computations in a round (which can be viewed as a function of current time T, $r = r(T)$). The parameter that reflects the hashing power of the honest miners can be denoted by $X = (1-\alpha)npq$, and similarly $Z = \alpha npq$ represents the hashing power of malicious miners. For malicious miners, they may violate the blockchain protocol by withholding blocks and not broadcasting them to the network, while for honest miners they can only accept one block in a round no matter how many blocks were generated. Thus the probability that honest miners find at least one block in a round is $1-(1-p)^{(1-\alpha)nq}$, and the probability that honest miner find only one block is $(1-\alpha)nqp(1-p)^{((1-\alpha)nq-1)}$. We denote Y the lower bound of the probability, that for $p < 0.1$ and $\theta \in (p, 2)$, we have:

$$\begin{aligned} 1-(1-p)^{(1-\alpha)nq} &\geq (1-\alpha)nqp(1-p)^{((1-\alpha)nq-1)} \\ &\geq \theta e^{(-\theta-p)} = Y \end{aligned} \tag{1}$$

Secure Transfer with Wallet: Compared with traditional crowdsourcing system in money reward or exchange, we use virtual coins in blockchain. Coins can be obtained by mining or transferring with others. We assume that each user who has the secret key can securely possess and transfer it with the client wallet.

Secure Encryption Algorithm: We assume that the solution is encrypted by leveraging a secure public key encryption algorithm. Workers use the corresponding requester's public key to encrypt the solutions. Requesters could decrypt the solutions successfully by the secret key. Specifically, the solutions are saved as cipher text in distributed database.

4.4 Technical Challenges

Traditional crowdsourcing relies on a centralized system to post and receive tasks, which is essentially a three-party system model. The blockchain technology has provided a promising way to solve the trust issue among multiple parties. One may simply adopt the blockchain technology in crowdsourcing by using smart contract to depict the process of task crowdsourcing: requester posts a task with the task reward by smart contract and workers can receive it by operating the contract. When the task is finished, workers give solutions to requester in off-chain. However, as pointed out in the threat model, requesters and workers may benefit from not following this process. Requesters may evaluate the solution to be low-quality even if workers pay high effort, or deny obtaining workers' solutions for short of the arbiters. Workers may pay low effort to submit solutions for the sake of task reward, or even receive the task but not submit solution. Different from the digital currency transfer in Bitcoin, it could be more complicated that numerous malicious behaviors may discourage requesters and workers from participating in the blockchain-based crowdsourcing system. Therefore, how to ensure the fairness between requesters and workers under the decentralized crowdsourcing system is our first challenge:

Challenges 1: *Utilizing the blockchain technology to enhance crowdsourcing, while lacking an efficient scheme to ensure the fairness between requesters and workers under the fully decentralized framework.*

The blockchain is an infinite extended ledger that new blocks will be created continuously. It is improper to put too much data on blockchain. However, crowdsourcing task may contain huge size of data. Taking image tagging task for example, there may be thousands of images to be labeled, which is impossible to put these data in block. What's worse, huge block data has an effect on the message synchronization and takes too much disk size. For example, at 29 May 2018, a full mining node of bitcoin needs to occupy 156G total disk space to synchronize with the network. Thus, how to store the large-scale crowdsourcing data under blockchain-based crowdsourcing system is our second challenge:

Challenges 2: *Performing large scale crowdsourcing process atop on blockchain technology while limited data storage on block.*

5 CrowdBC: Blockchain-based Decentralized Framework for Crowdsourcing

5.1 Overview of CrowdBC

Combining the advantages of blockchain, we formalize a decentralized crowdsourcing framework named CrowdBC. It allows users to finish a crowdsourcing process in the logic plane and store their encrypted solutions in the data plane. First and foremost, CrowdBC Client is designed as the user interface in the logic plane, and it runs locally on user's personal computer without depending on any central server. More importantly, CrowdBC Client allows workers and requesters to interact with the underlying blockchain. Requesters and workers reach an agreement on top of blockchain which is used to achieve eventual consensus on the state of each task. It supports all operations for the crowdsourcing, such as registration, posting task and receiving task.

Three Layers Architecture: Inspired by [33], we divide CrowdBC into three layers: the application layer, blockchain layer and storage layer. As shown in Fig. 3, two layers (application and blockchain layer) lie in the logic plane and one layer (storage layer) lies in the data plane. Workers with special skills could query and compete tasks which are posted by requesters in the application layer. The blockchain layer uses the task state changes as input to achieve consensus between workers and requesters. Notice that, there exist lots of data collected from requesters and workers, because of the limited data storage capacity in blockchain. We separate the logic layer and the data layer and believe this separation can improve CrowdBC's data storage significantly. We put the task metadata (such as data size, owner, hash value, pointer) in the blockchain layer and raw data in the storage layer. Thus, users do not need to trust the data saved in the data layer and they can verify the integrity and authenticity of data in the logic layer.

Underlying Blockchain: Without loss of generality, the blockchain we adopt supports execution of any arbitrary "program" which is short for Turing-complete program (e.g. smart contract). We assume that each blockchain platform has a "Compiler" to compile the "program". And how to build a compiler is out of the scope of this paper and we do not depict here. We design an user interface module in the client for workers and requesters to interact with the "program" and the blockchain.

State Machine Construction: Our framework constructs a state machine to depict task processing. It depicts the task life cycle, and each state represents the global status of the task. The task is triggered from the current state to the next state with users's valid input in the application layer. Fig. 4 shows the different states that a task can be in and how the state transfers. Each task generates a new state machine and the global state of the task is updated successfully when a new block is created. There exist six states: *Pending, Unclaimed, Claimed, Evaluating, Canceled, Completed*. Users can query task's current state at any time by themselves.

5.2 CrowdBC Layers

Now, we present the architecture of CrowdBC which contains two planes: the logic plane and the data plane. The logic plane, which consists of the application layer and blockchain layer, is used as providing user management and task management for requesters and workers. The data plane which is responsible for task data or solution storage mainly refers to the storage layer.

5.2.1 *Application Layer*

The application layer mainly refers to CrowdBC Client. It provides users with entrance to finish a crowdsourcing task and contains three main modules: User Manager (UM), Task Manager (TM) and Program Compiler. The client runs correctly without relying on a central server, even there exist some failed nodes, the services for crowdsourcing are not affected in CrowdBC. This design can improve the security of crowdsourcing system.

To make it more clearly, we introduce each module's function respectively. UM acts as the registration and user information

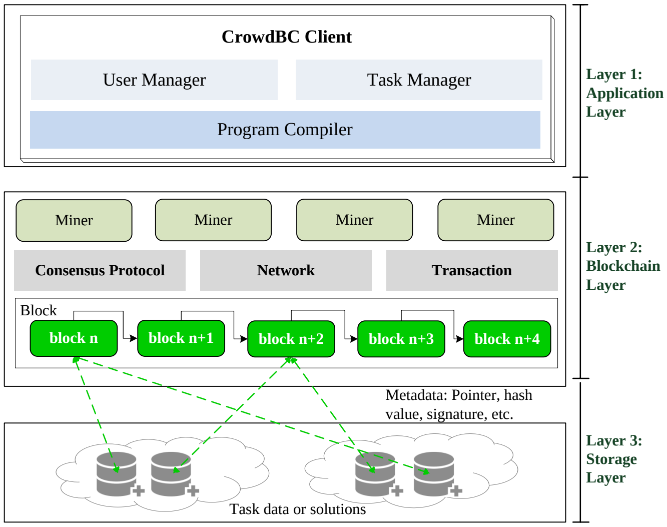


Fig. 3: Overview of CrowdBC's architecture.

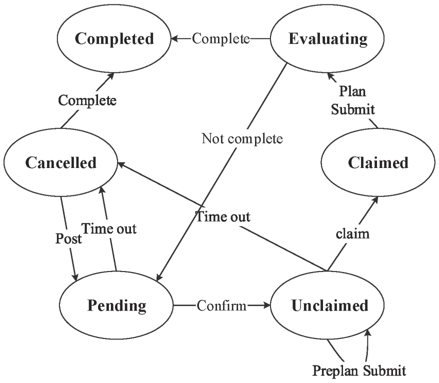


Fig. 4: State machine model for a task.

management. Users should first register before starting the crowdsourcing task. They do not need to provide true identities and just register with key pairs (a public key and private key). Meanwhile, each identification is related with a default reputation value. The value is changed upon the worker's behavior and cannot be updated by himself/herself. A new user fills personal information which refers mainly on key pair and description with the client, and creates a new "program" in the blockchain. A middle module "Program Compiler" is built to convert the new creating "program" into executable language of the blockchain layer. Once the "program" is written into the blockchain, the user registers successfully. Then, he/she can post or receive a task based on TM which is the task management module. Crowdsourcing tasks are depicted into "program" and run in the blockchain, including task posting, task receiving, solution submission and reward assignment. Remarkably, in order to get satisfactory results, the requester only allows qualified workers who reach a minimum reputation value to receive the task. We will give the detailed description about the decentralized crowdsourcing protocol in section 6.4.

5.2.2 *Blockchain Layer*

The blockchain layer is the middle tier and serves two purposes: 1) providing consensus on the order in which "program" is written and 2) running state machine. The "program" is sent to the blockchain layer after being compiled, then they are written to the blockchain after being confirmed by miners. The proposed framework defines the logic of state transition by the "program" via cryptographically-secured transactions. State machine uses valid input from the application layer and triggers task state changes in the blockchain layer ultimately.

Generally speaking, blocks in blockchain layer should not hold Important too much data. Otherwise, it will have an affect on the network synchronization and take too much disk space. In order to reduce the data size stored on blockchain, we separate the metadata (including owner, time stamp, pointer, data hash value, etc.) from the actual storage of data. In detail, the task attachments and solution data are stored off-blockchain in the distributed database. A data pointer which consists of a query string is saved in the blockchain and can be used to find the data in the storage layer. Besides, the hash value of the data is saved in the blockchain, which can guarantee the data have not been changed in the storage layer. By this way, the data storage capacity of our framework increases obviously.

5.2.3 *Storage Layer*

The storage layer is the lowest tier, which is mainly used to store the actual data values of tasks and solutions. We do not adopt any particular storage in the framework, instead allowing multiple storage providers to coexist, such as S3, IPFS [41] or a distributed Hashtable (e.g. Kademilia [42]). The data values are signed by private keys of the owners. Users donnot need to believe the data stored in the storage layer, and they could check the authenticity and integrity of the data values by data's hash and digital signature in the blockchain layer. In addition, worker submits a solution to the system and use requester's public key to encrypt the solution, which means only requester can decrypt it. In this way, we can ensure data security and prevent data from being leaked to irrelevant users. In particular, by storing task data outside of the blockchain, CrowdBC allows values of arbitrary size and satisfies actual demands for crowdsourcing.

5.3 The Crowdsourcing Process in CrowdBC

important

In this section, we describe the general process of our framework. Based on CrowdBC Client, our framework consists of six steps as follows:

Step1. In the first step, requesters and workers register in CrowdBC. CrowdBC Client transfers users' information into the input of "program" which is written into a transaction and will be sent to blockchain. Each registered user is assigned with a public key pair.

Step2. Updating "program" can be seen as a transaction which needs to be confirmed by miners. The following steps are all related with this step and it depicts that the data and status are recorded on the blockchain permanently.

Step3. It is performed by requester to post tasks. Requesters are required to pay reward in advance and payments are deposited on the blockchain. Meanwhile, a rule for workers is set by requesters to ensure that qualified workers could ultimately receive the task. An evaluation function is also required, and thus the solution can be evaluated by miners on the blockchain instead of the requester or the crowdsourcing system.

Step4. Registered workers receive the posted task by interacting with CrowdBC Client. Each worker receives a task should deposit some coins or a reputation value to ensure the quality of the task.

Step5. Workers submit solutions before the task deadline when they finish the task. The solutions are encrypted with the requester public key and sent to the distributed storage. Meanwhile, a hash value and pointer are stored on the blockchain. Requester could find the solutions by the pointer and decrypt them with his private key.

Step6. The last step is about solution collection, reward assignment and task evaluation. Workers or requesters can complete this step initiatively by publishing a transaction to the blockchain. Rewards are automatically assigned to workers according to the evaluation results which determine how many rewards they can obtain and are related to their efforts. High efforts and good performances will get more reward and improve the reputation.

6 A Concrete Implementation of CrowdBC

6.1 Crowdsourcing Smart Contracts

In this section, we present a concrete scheme of CrowdBC. The blockchain which supports smart contracts is adopted in the design. From here on, we denote the "program" as smart contract. We implement three types of smart contracts: User Register Contract (URC), User Summary Contract (USC), Requester-Worker Relationship Contract (RWRC). Fig. 5 shows the contract structures and relationships.

Basically, user (R or W) information is divided into two parts: basic information and detailed information. The basic information which contains name, address and type are saved in the global URC contract. The latter, including one user's profile, expertise, reputation and task list, are saved in USC and updated with the task processing. Notice that USC is created simultaneously when a user successfully registers in URC. Besides, R and W reach the agreement in RWRC which depicts the constraint condition in the task processing.

Specially, there exist three important algorithms: coin processing algorithm lockUtil(·), reputation updating algorithm updateReputation(·) and solution evaluation algorithm solutionEvaluate(·). The first algorithm is to lock the deposits on the blockchain before the deadline and assign reward to the workers upon the last algorithm result. The second algorithm is to manage workers' reputation. The value of reputation and expertise are automatically updated only with the completed task. CrowdBC evaluates the solution on blockchain with the automatically executed smart contract. With respect to the last algorithm, we assume that there exists a trustful truth discovery algorithm that can evaluate the solutions submitted by a well-defined function solutionEvaluate(·) in the programs. There exist some emerged technologies supporting this process [43], [44], [45], [46], such as indistinguishability obfuscation [47], homomorphic encryption [48]. How to design an appropriate evaluation mechanism in the decentralized crowdsourcing framework is an important issue and we will extend this work in the future. The reward assignment and reliability value updating rely on the output of solutionEvaluate(·). For simplicity, the output will be "H" (high level of effort) or "L" (low level of effort).

Traditional crowdsourcing systems focus on detecting cheating or malicious behaviors after workers have submitted results. In contrast, CrowdBC selects trustworthy workers based on reputation and reliability value in smart contract, which can effectively improve the quality of results. In order to achieve this goal, we combine expertise-aware with reputation to choose workers in RWRC. Each worker W_j is associated with an attribute $Lists[]$ which contains the crowdsourcing task in special type that they have received and finished. In particular, $Lists[] = \{category_1(\varepsilon_1, \varphi_1), \ldots, category_k(\varepsilon_k, \varphi_k)\}$. The reliability rel_k of W_j in category $category_k$ task can be calculated by $rel_k = \varepsilon_k / \varphi_k$. It can be used to verify if a worker is somehow topic expert in special category.

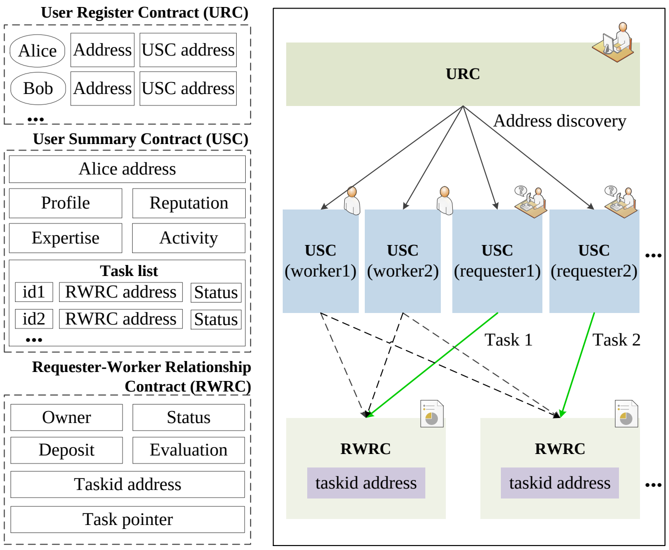


Fig. 5: The structure of smart contracts in CrowdBC and data references.

6.1.1 User Register Contract (URC)

Upon registration, each requester or worker does not need to submit his/her true identity, and will be assigned with a key pair: a "public key" and a "private key". The global URC contract produces a user's address by generating a hash with the public key. The address contains no information about the user, which provides users with much higher-level privacy than users in traditional crowdsourcing systems. Meanwhile, a USC contract corresponding to the new registered user will be created.

As mentioned above, users are allowed to use pseudonyms to finish transactions. However, some workers may register with true identity that can be authenticated in certified institutions, which can increase the probability of receiving task in CrowdBC. Besides, we set rules into the URC contract that registering new identities will be recognized and the mapping of the total user list could be updated. Notice that updating or creating a contract need transaction fee which is paid by the party who publishes it. Transaction fee is given to miners who confirm the transaction and support CrowdBC running persistently.

6.1.2 User Summary Contract (USC)

This contract stores the personal statistics information and evaluation for requesters and workers according to their past behavior. We establish multi-metrics to evaluate workers and requesters in USC for the sake of reducing any subjective judgment, including *profile, reputation, task general description* and *activity*. *Profile* mainly describes user basic information, including skills, profession, etc. Specially, if users register with true identities, profile also contains a digital signature signed by a certificate authority, and users can authenticate identities by their public keys. This metric is set up when users register at the first time and can be updated by themselves. *Reputation* β_W is an important parameter which is initialized with default value and updated with the completion of the task. In our framework, we build the reputation-based incentive mechanism based on [14], which will be described in section 6.3. High reputation value reflects a user's good performance in the past. *Task general description* refers to the summary information about task statistics, including total receiving task lists and high evaluation task list (i.e., (ε, φ)). *Activity* describes the level of activity and working extent for users, including proportion of task-delay, biding number. High activity level depicts hard working with tasks. It is worth noting that these metrics cannot be changed easily by any single third party and are automatically updated only with the related completed task.

Workers find an uncompleted task by querying requester's task list in USC. Each task in USC has a status. Tasks in the state of *Pending* or *Unclaimed* illustrate that they still accept solutions and the qualified workers can receive the task and verify the task signature with requester's public key. USC also contains a list of task addresses which can point to user's previous task in the Requester-Worker Relationship Contract (RWRC).

6.1.3 Requester-Worker Relationship Contract (RWRC)

Requester-Worker Relationship Contract (RWRC) depicts the agreement between requesters and workers, which is about the process of task posting, task receiving, solution evaluation, and reward assignment. It is created when requester posts a task T and publishes the task information, $T = \{desc, K_R^p, coins(v + \pi_R), (\beta_k, \varepsilon_k, \varphi_k), \lambda TI_{deadline}, TI_{confirm}, soluationEvaluation(\cdot)\}$. Requester posts the task T in a transaction with his/her private key and other workers could check it by the corresponding public key.

When W_j wants to receive task, RWRC contains a validation function $\mathsf{checkWorkerQualification}(\cdot)$ and checks if worker's reputation and reliability value satisfy the minimum limited. Generally, a minimum reputation and reliability value is set to avoid low level workers. Meanwhile, requester defines a fixed worker pool W_{pool} to store workers' addresses, the size of workers pool λ is corresponding to required workers, and each worker who satisfies the validation function would add his/her address to W_{pool}. If workers are qualified, they update RWRC by publishing it to the blockchain, which represents they receive a task successfully. RWRC contract cannot receive workers if the size of W_{pool} exceeding λ. Requester cannot assign the task to workers more than he/she could pay, because smart contract is published on the network and each miner would verify.

As mentioned before, the data saved on the blockchain should not be too large due to the limited storage. Thus we put only the task metadata on the blockchain by RWRC and other detail information to the distributed storage layer. Moreover, in order to prevent requesters from behaving as "false-reporting" and workers from behaving as "free-riding" in pursuit of self-interest maximization, a timed-locked deposit protocol is constructed, which will be depicted in section 6.2. Making a deposit before processing crowdsourcing task is a unique feature which is necessary under our framework to guarantee the fairness of the users. Note that if worker submits an effective solution which is confirmed by a miner, the deposit will be returned back to him/her; otherwise, the coin will be deducted by requester and worker's reputation value will be reduced.

Different from the traditional model in which solutions are evaluated by requester or the crowdsourcing system, the solutions in CrowdBC are evaluated by miners. The evaluation function $\mathsf{solutionEvaluate}(\cdot)$ is posted with RWRC by requester and miners could verify the solutions without knowing the solution details.

As to data storage, a particular space is allocated for each RWRC in the storage layer. Task attachments and solutions are stored in the space, and the corresponding hash values are recorded in the blockchain to guarantee solutions unaltered at the source. Especially, to protect data privacy, workers use requester's public key to encrypt the solutions. Requester can decrypt them and verify the integrity of the data values in the blockchain layer. On the other hand, a pointer is created once worker submits the solution successfully. It is also written in RWRC and can be used to find the solution for requester.

6.2 Time-locked Deposit Protocol for Crowdsourcing

Our key idea is letting requesters and workers obtain fair results atop on blockchain without relying on central crowdsourcing platform. To prevent false-reporting and free-riding attacks, participants are required to make a time-locked deposit as a guarantee to regulate their behavior. The deposit which can be funds or reputation value (only for workers) will be assigned to designated entities according to the pre-defined smart contract. Some works have been done to achieve time-locked blockchain deposit protocol [49], [50], [51]. These protocols enable one party (*payer*) to exchange with other parities (*payee*) to lock a certain coins as a guarantee on blockchain. To prevent equivocation attack, the party cannot redeem the deposit until deadline even if he/she has the secret key. However, these protocols cannot be directly applied to crowdsourcing scenario with two reasons: one is that workers are not explicit when requester makes the deposit. The other is that they assume that payees are honest and may not perform malicious behavior, while this is not always the case in CrowdBC.

Inspired by [52], [53], we illustrate F_{RR}^* (RR stands for "refund-or-reward"), a time-locked deposit protocol for crowdsourcing. The adversary will be punished with monetary or reputation penalty when he/she aborts or breaks the task crowdsourcing protocol which is pre-defined in F_{RR}^*. The other honest party will be compensated with adversary's deposit. F_{RR}^* allows requester to send coins to worker under some condition. In special, the condition is to verify if the solutions submitted by workers satisfy pre-defined requirements (i.e., $\mathsf{solutionEvaluate}(\cdot)$). We define F_{RR}^* scheme to be a tuple of phases as shown in Fig. 6.

6.3 Reputation Management

CrowdBC builds the incentive mechanism based on users' past behavior. Requesters are demanded to pay money before they post task, so we mainly focus on workers' reputation. Each worker is assigned with a reputation which can be viewed as one of the important references for requesters when they choose workers. A high reputation reflects their good behaviors on solving tasks in the past; otherwise, they will be limited to participate in some tasks.

Unlike traditional crowdsourcing systems where the reputation management is implemented by the third party, we define the protocols and implement them in the decentralized blockchain. Each worker is tagged with a reputation β_W. β_W is an integer

Time-locked Deposit Protocol for Crowdsourcing F^{*}_{RR}

Phase 1. RDeposit. Upon receiving $RDeposit_{R_i} := \{sid, T, TI_{deadline}, TI_{confirm}, K^{p}_{R_i}, (\beta_k, \varepsilon_k, \delta_k), \lambda, coins(v+\pi_{R_i})_{TI}, solutionEvaluate(\cdot)\}$ from the requester R_i, record the message $RDeposit_{R_i}$ to the blockchain network by creating RWRC and workers can find it, where sid is the session id, $coins(v+\pi_{R_i})_{TI}$ is the deposit that can only be unlocked after $TI_{deadline}$ by the requester or the task being received by workers $W = \{W_1, ..., W_j, ..., W_m\}$ with the signature in Phase 2.

Phase 2. WDeposit. Before $TI_{deadline}$, W_j checks if the remaining deposit is not redeemed and receives the task if he/she satisfies the conditions (i.e., $\beta_{W_j} \geq \beta_k$ and the expertise value satisfies $rel_{W_j} \geq rel_k$ and $\delta_{W_j} \geq \delta_k$). Then, W_j sends $WDeposit_{W_j} := \{sid, T, TI_{deadline}, TI_{confirm}, K^{p}_{R_i}, K^{p}_{W_j}, coins(v_{R_i})_{TI}, coins(\pi_{W_j})_{TI}, redeem(\cdot)\}$ to the blockchain network. $coins(v_{R_i})_{TI}$ is transferred into $WDeposit_{W_j}$ with worker's signature. Creating the redeem script that takes task solution as input following the protocol: $redeem(\cdot) = \mathsf{Verify}(\{\cdot\})_{K^{p}_{R_i}} \wedge (\mathsf{Verify}(\{\cdot\})_{K^{p}_{W_j}} \vee \mathsf{solutionEvaluation}(\cdot))$

Phase 3. Claim. Before $TI_{deadline}$, W_j submits the solution to the distributed storage system and gives the address to R_i. R_i confirms the solution and sends $\mathsf{Sign}(s_j)_{K^{s}_{R_i}}$ to W_j. Then, W_j submits the claim transaction in RWRC by providing signature $\mathsf{Sign}(s_j)_{K^{s}_{R_i}}$ and $\mathsf{Sign}(s_j)_{K^{s}_{W_j}}$) to redeem the reward: $Claim := \{sid, T, K^{p}_{R_i}, K^{p}_{W_j}, coins(v_{R_i})_{TI}, coins(\pi_{W_j})_{TI}, \mathsf{Sign}(s_j)_{K^{s}_{R_i}}, \mathsf{Sign}(s_j)_{K^{s}_{W_j}}\}$.

Phase 4. Reward. Before $TI_{confirm}$, R_i or W_j may initially launch the reward phase. They check if $RDeposit_{R_i}$ and $WDeposit_{W_j}$ are not deleted and get enough confirmations in blockchain for security. Then, one of them broadcasts a transaction that redeems the deposit:

- If $\mathsf{solutionEvaluate}(\cdot)$=H, sending $coins(v_{R_i}+\pi_{W_j})$ to the worker W_j.
- If $\mathsf{solutionEvaluate}(\cdot)$=L, sending $coins(v_{R_i})$ to the requester R_i and $coins(\pi_{W_j})$ to the worker W_j.

Fig. 6: The time-locked deposit protocol for crowdsourcing.

number from the finite set $set(0, 1, ..., \beta^{Max}_{W})$, where β^{Max}_{W} represents the max size of this set. h_k is the average reputation of the whole workers. Updating β_W depends on the outcome of solutionEvaluation$(\cdot)$ in RWRC. If a miner confirms a solution and gives the positive evaluation, worker's reputation will be increased and recorded in blockchain. Note that worker cannot update the reputation by himself, because miners will not confirm this type of transactions. Let "a" refer to the output of the evaluation function. "$a = H$" stands for high effort of action and "$a = L$" stands for low effort of action. Thus, the reputation β_W can be computed as follows:

$$\beta_W = \begin{cases} \min(\beta^{Max}_{W}, \beta_W + 1), & \text{if } a = H \text{ and } \beta_W \geq h_k \\ \beta_W - 1, & \text{if } a = L \text{ and } \beta_W \geq h_k + 1 \\ 0, & \text{if } a = L \text{ and } \beta_W = h_k \\ \beta_W + 1, & \text{if } \beta_W < h_k + 1 \end{cases} \tag{2}$$

where h_k denotes the threshold of the selected social strategy, which is a method of using social norms to control workers' behaviors [14]. If the worker's reputation falls to h_k and receives an "L" feedback by using the evaluation function, his/her reputation will fall to 0 and cannot receive most of the tasks. He/she needs to receive enough simple tasks and get positive feedback until his reputation value reaching h_k again.

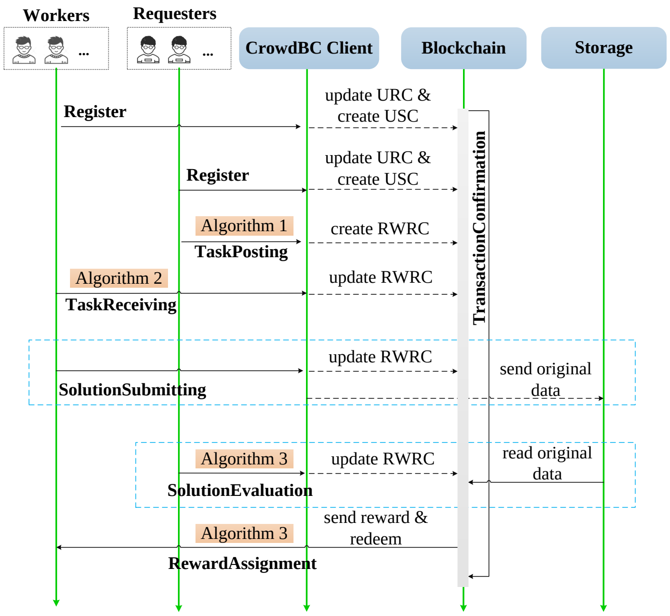


Fig. 7: The process of crowdsourcing in CrowdBC and smart contract updating.

6.4 The Proposed Decentralized Crowdsourcing Protocol

In the section, to formalize the decentralized crowdsourcing protocols, we adopt a designed notational system such that readers may understand our constructions without understanding the precise details of our formal modeling. It consists of six algorithms: Register, TransactionConfirmation, TaskPosting, TaskReceiving, SolutionSubmitting, SolutionEvaluation, RewardAssignment. Users interact with the blockchain by CrowdBC Client. To make it clearer, we elaborate the general contract flow of CrowdBC in Fig. 7.

6.4.1 Register

We require that requesters and workers register in CrowdBC to get their identities (mainly refer to public key, private key and public address) via URC contract, i.e. $R_i = (K^{p}_{R_i}, K^{s}_{R_i}, K^{a}_{R_i}), i = 1, ..., n$ and $W_j = (K^{p}_{W_j}, K^{s}_{W_j}, K^{a}_{W_j}), j = 1, ..., m$. Worker's initial reputation value β_W is is the average reputation value of all workers.

6.4.2 TransactionConfirmation

The processes of creating and updating a contract can be seen as a transaction which needs to be confirmed in the blockchain. Miners can verify the effectiveness of transactions. Workers and requesters can participate in the blockchain as a "miner" and contribute their resources to achieve a trustworthy chain. To model the confirmation of the transaction and the execution of blocks, we define the blockchain state as a pair (BC_{σ}, BC), where BC_{σ} is the previous block and BC is the current one. $BC = \{M_{addr}, (Tx_1..Tx_c..Tx_k), timestamp, blockid, preblockhash\}$, where M_{addr} is the address of a miner, and Tx_c is the contract which needs to be confirmed. Users need to wait for several blocks to ensure that the contract is recorded immutably in blockchain.

6.4.3 TaskPosting

After registration, requester could post a task T to Crowd-BC, $T = \{sid, T, TI_{deadline}, TI_{confirm}, K_R^p, (\beta_k, \varepsilon_k, \delta_k), \lambda, coins(v + \pi_R)_{TI}, \mathsf{solutionEvaluate}(\cdot)\}$. For each task, there is a limited condition $(\beta_k, \varepsilon_k, \delta_k)$ which means the minimum corresponding value of workers who can receive the task. It is worth noting that setting these data too large will have an affect on the number of participants. λ refers to the number of workers required to complete the task. In order to avoid denial of payment by requester, we specify that requester makes a deposit by following F_{RR}^*. Algorithm 1 illustrates the implementation of posting task.

Algorithm 1: Post task

```
Input: requester $R_i$, task description $T$, task reward $coins(v_{R_i} + \pi_{R_i})$,
        the limited condition of workers $(\beta_k, \varepsilon_k, \delta_k)$, finish time
        $TI_{deadline}$, confirm time $TI_{confirm}$, maximum workers number $\lambda$,
        USC address $USC_{R_i}^{addr}$
Output: RWRC contract $RWRC_T$, result validation function
        $solutionEvaluation(\cdot)$, update $USC_{R_i}$
if $R_i$ is unregistered then
    $R_i$ has not been registered;
    goto final;
if $lockUtil(K_{R_i}^p, coins(v_{R_i} + \pi_{R_i}), TI_{deadline} + TI_{confirm})$ is not success
then
    $R_i$ deposits the reward on blockchain failed ;
    goto final;
$\mathsf{checkWorkerQualification}(\cdot) \leftarrow (\beta_k, \varepsilon_k, \delta_k)$ ;
$\mathsf{solutionEvaluation}(\cdot) \leftarrow T$ ;
$W_T^{list}(1, ..., \lambda) \leftarrow \lambda$ ;
$receivedWorkerNum_T \leftarrow 0$ ;
$RDeposit_R \leftarrow \{T, W_T^{list}(1, ..., \lambda), (\beta_k, \varepsilon_k, \delta_k), coins(v_{R_i} +$
$\pi_{R_i}), TI_{deadline}, TI_{confirm}, \mathsf{solutionEvaluation}(\cdot))\}$ ;
$USC_{R_i}^{T_{pool}}$ put $RWRC_T^{addr}$ ;
$\mathsf{updateUSCContract}(RWRC_T^{addr}, USC_{R_i})$ ;
final ;
return $RWRC_T$, $\mathsf{solutionEvaluation}(\cdot)$;
```

6.4.4 TaskReceiving

Worker finds uncompleted tasks in requester's USC contract and receives a task if he/she satisfies the condition set by requester. The condition function $\mathsf{checkWorkerQualification}(T, \beta_{W_j}, rel_{W_j})$ means that a task T can be received by a worker W_j with the value $\beta_{W_j} \geq \beta_T$, $rel_{W_j} \geq rel_T$. Besides, for the sake of making workers do the job industriously, worker deposits a coin or certain reputation value before receiving the task. If he/she chooses coin $coins(v)$ as the deposit, we still use the function $\mathsf{lockUtil}(K_{W_j}^p, coins(v), TI_{deadline} + TI_{confirm})$. Otherwise, reputation is reduced by β_{W_j} within the process of this task and added after completing the task if the worker submits solution on time. Worker signs on T with the $K_{W_j}^s$, which can ensure the correctness for task reward assignment in the end. Algorithm 2 illustrates the implementation of receiving task.

6.4.5 SolutionSubmitting

Worker could submit solution to requester if he/she completes the task (algorithm 3). The task solution, encrypted by requester's public key K_R^p and signed by worker's private key K_W^p, is stored on the distributed database (e.g., IPFS). The hash and pointer of the solution is stored on the blockchain. Requester could get the solution by the pointer and decrypt it using his/her private key.

Algorithm 2: Receive task

```
Input: RWRC contract $RWRC_T$, worker $W_j$, worker deposit coin
        $coin(\pi_{W_j})$, worker deposit reputation $Rep_T$, worker $USC_{W_j}$
Output: update RWRC contract $RWRC_T$ and USC contract $USC_{W_j}$
if $W_j$ is unregistered then
    $W_j$ has not been registered;
    goto final;
if $\mathsf{checkWorkerQualification}(T, \beta_{W_j}, rel_{W_j})$ is dissatisfactory then
    $W_j$ does not satisfy the condition;
    goto final;
if $receivedWorkerNum_T >= W_{num}$ then
    Task $T$ cannot be accepted anymore;
    goto final;
if
$coin(\pi_{W_j}) \neq 0$ & $\mathsf{lockUtil}(K_{W_j}^p, coin(\pi_{W_j}), TI_{deadline} + TI_{confirm} - now)$
is success then
    $W_j$ deposits reward on blockchain succeeded ;
else if $\beta_{W_j} \neq 0$ & $\mathsf{updateUSCContract}(USC_{W_j}, K_{W_j}^p, (\beta_{W_j} - Rep_T))$ is
success then
    $W_j$ deposits reputation on blockchain succeeded ;
else
    $W_j$ makes a deposit in blockchain failed ;
    goto final;
$Sign_{W_j} \leftarrow$ Digital signature on $T$ with $\mathsf{Sign}(T)_{K_{W_j}^s}$ ;
$W_T^{list}(1...W_{num})(T)$ add $Sign_{W_j}$ ;
$USC_W^{T_{pool}}$ put $RWRC_T^{addr}$ ;
$receivedWorkerNum_T$++ ;
$WDeposit_{W_j} \leftarrow$
$\{T, TI_{deadline}, TI_{confirm}, K_{R_i}^p, K_{W_j}^p, coins(v_{R_i})_{TI}, coins(\pi_{W_j})_{TI}, redeem(\cdot)\}$;

$\mathsf{updateUSCContract}(RWRC_T^{addr}, USC_{W_j})$ ;
final ;
return $RWRC_T$ and $USC_{W_j}$;
```

Algorithm 3: Submit solution

```
Input: RWRC contract $RWRC_T$, task solution $s_j$, worker $W_j$, requester
        $R_i$
Output: solution pointer $s_j^{poniter}$, solution hash value $s_j^{hash}$
if $now > TI_{deadline}$ then
    $s_j$ cannot be submitted for timeout;
    goto final;
$Sign_{s_j} \leftarrow$ Digital signature on $s_j$ with $\mathsf{Sign}(s_j)_{K_{W_j}^s}$ ;
$s_j^{encrypted} \leftarrow$ Encrypt the solution $\{s_j, Sign_{s_j})\}$ with $K_{R_i}^p$ ;
$s_j^{hash} \leftarrow \mathsf{H}(s_j^{encrypted})$ ;
$s_j^{poniter} \leftarrow \mathsf{sendDataToIPFS}(s_j^{encrypted})$ ;
$TI_{submit} \leftarrow now$ ;
$RWRC_T^{s_{list}} \leftarrow \{s_j^{hash}, s_j^{poniter}, TI_{submit}\}$ ;
$\mathsf{updateUSCContract}(RWRC_T^{addr}, USC_{W_j})$ ;
$\mathsf{updateUSCContract}(RWRC_T^{addr}, USC_{R_i})$ ;
final ;
return $s_j^{poniter}$ and $s_j^{hash}$;
```

6.4.6 RewardAssignment

After submitting the solution, workers could demand for the process of task evaluation and reward payment, or requesters initiatively finish them when the finish time is on. In our design, we assume that the evaluation result is given under the evaluation function and miners on the blockchain could confirm. As shown in algorithm 4, the task reward paid to workers is quality-contingent payment (i.e., better performance will get more reward). The evaluation result will be automatically synchronized with worker's USC contract to update his/her reputation.

Algorithm 4: Evaluate task and send reward

```
Input: RWRC contract $RWRC_T$, requester $R_i$, workers list $W_{list}$,
       selected social strategy $h_k$
Output: update RWRC contract $RWRC_T$ and USC contract
        $USC_{R_i}, USC_{W_j}$, send reward to related worker $W_j$
$rewardNeedSend \leftarrow coins(v)/\lambda$;
for each $W_j$ in $W_{list}$ do
    if $TI_{submit} \leq TI_{deadline}$ then
        if Verify$(s_j^{hash})_{K_{W_j}^p}$ is not success then
            Check $K_{W_j}^p$ signature failed ;
            continue ;
        $evaluationResult_j \leftarrow$ solutionEvaluation$(T, s_j^{poniter}, s_j^{hash})$ ;
        $oldRep \leftarrow \beta_{W_j} + Rep_T$ ;
        if $oldRep \geq h_k$ & $evaluationResult_j \equiv H$ then
            $\beta_{W_j} \leftarrow min\{\beta_{W_j}^{Max}, oldRep + 1\}$ ;
            $rewardNeedSend \leftarrow coins(v_{R_i} + \pi_{W_j})$ ;
        else if $oldRep \geq h_k$ & $evaluationResult_j \equiv L$ then
            $\beta_{W_j} \leftarrow oldRep - Rep_T$ ;
            $rewardNeedSend \leftarrow coins(\pi_{W_j})$ ;
        else if $oldRep \equiv h_k$ & $evaluationResult_j \equiv L$ then
            $\beta_{W_j} \leftarrow 0$ ;
            $rewardNeedSend \leftarrow coins(\pi_{W_j})$ ;
        else if $oldRep < h_k$ then
            $\beta_{W_j} \leftarrow oldRep + 1$ ;
            $rewardNeedSend \leftarrow coins(\pi_{W_j})$ ;
    else
        $evaluationResult_j \leftarrow L$ ;
        $rewardNeedSend \leftarrow 0$ ;
        $\beta_{W_j} \leftarrow \beta_{W_j} - Rep_T$ ;
    $isSendRewardSuc \leftarrow$ sendReward$(K_{W_j}^p, rewardNeedSend)$ ;
    updateReputation$(USC_{W_j}, K_{W_j}^p, \beta_{W_j})$ ;
    updateUSCContract$(RWRC_T^{addr}, USC_{W_j})$ ;
    updateUSCContract$(RWRC_T^{addr}, USC_{R_i})$ ;
updateAvgReputation$(\cdot)$
final ;
return $RWRC_T, USC_{R_i}, USC_{W_j}, isSendRewardSuc$;
```

6.5 Security Analysis

In our design, CrowdBC fulfills several security properties and we discuss them here.

Fully Fraud Resilient. With the majority honest security assumption in section 4.3, we discuss the fully fraud resilient property in CrowdBC.

Theorem. Assume that $Y \geq (1+\delta)Z$ with $\delta \in (0,1)$, then the probability that CrowdBC-based system violates *fully fraud resilient* property with parameter t is at most $e^{(-\Omega(\delta^3 t))}$.

Proof. The event that CrowdBC does not meet fully fraud resilient property happens if either a malicious requester or a malicious worker succeeds in building a fork chain which was finally accepted by honest miners in the system. We firstly define $C1$ and $C2$ as follows:

- $C1$: the malicious chain that requesters or workers violate fully fraud resilient property.
- $C2$: the normal chain that requesters or workers follow.

We show that for any t, the probability that $C1$ and $C2$ diverge at time $(T-t)$ is at most $e^{(-\Omega(\delta^3 t))}$. First, we show the probability that malicious miners can use blocks mined before time $T-(1+\delta/8)t$ to compensate is at most $e^{(-\Omega(\delta^3 t))}$. Let $t^* \leq T-(1+\delta/8)$ be the time that the latest block was found by honest miner in adversary's chain $C1$. If at time $t' \geq T-t$, the malicious chain $C1$ is accepted by honest miner, then malicious miners must find more blocks than honest miners during $[t^*, t']$ where all honest blocks become orphans. Let $\hat{X}$ denote the total number of rounds when honest miners find blocks, and $\hat{Z}$ denote the number of all block that are mined by the adversary during $[t^*, t']$. Since $t' - t^* \geq (\delta/8)t$, according to Chernoff bounds [54], we have:

$$\begin{aligned} Pr[\hat{X} \leq (1-\delta/5)Y \cdot r(\delta t/8)] \leq e^{-\Omega(Y\delta^2 r(\delta t/8))} = e^{-\Omega(\delta^3 t)} \\ Pr[\hat{Z}] \geq (1+\delta/3)Z \cdot r(\delta t/8)] \leq e^{-\Omega(Z\delta^2 r(\delta t/8))} = e^{-\Omega(\delta^3 t)} \end{aligned} \tag{3}$$

Then with overwhelming probability that:

$$\begin{aligned} \hat{X} &\geq (1-\delta/5)Y \cdot r(\delta t/8) \geq (1-\delta/5)(1+\delta)Z \cdot r(\delta t/8) \\ &> (1+\delta/4)(1+\delta/3)Z \cdot r(\delta t/8) > (1+\delta/4)\hat{Z} \end{aligned} \tag{4}$$

It contradicts the assumption.

During time t, malicious miners will try to obtain a large number of blocks to maintain the advantages of $C1$ over $C2$. However, whenever only one block was found by honest miner, it would be a chance for honest miners to consent. Let $\tilde{Z}$ denote the number of all block that are mined by the adversary during $(1+\delta/8)t$, and $\tilde{Y}$ denote the number of rounds in which only one honest miner find a block. By Chernoff bounds, we have:

$$\begin{aligned} Pr[\tilde{Z} \geq (1+\delta/9)(1+\delta/8)Z \cdot r(t)] \leq e^{-\Omega(Z\delta^2 r(t))} \\ Pr[\tilde{Y} \leq (1-\delta/4)Y \cdot r(t)] \leq e^{-\Omega(Y\delta^2 r(t))} \end{aligned} \tag{5}$$

With overwhelming probability that:

$$\begin{aligned} \tilde{Y} &\geq (1-\delta/4)Y \cdot r(t) \geq (1-\delta/4)(1+\delta)Z \cdot r(t) \\ &> (1+\delta/9)(1+\delta/8)Z \cdot r(t) > \tilde{Z} \end{aligned} \tag{6}$$

Finally, we could conclude that the probability $C1$ catching up with $C2$ is exponentially small in t.

Security against False-reporting and Free-riding Attacks. In CrowdBC, solutions are evaluated with the automatically predefined smart contract. Under the assumption that the majority of miners are honest, malicious requesters cannot tamper the results of smart contract to launch false-reporting attack. In addition, we have proved that malicious requesters have limited probability to create a fork chain which is in their favor. With respect to free-riding attack, workers are required to make deposits in F_{RR}^* before receiving tasks. They are automatically assigned rewards according to the results of evaluation function. Thus, if they contribute low-quality solutions, they will not get rewards.

No Single Point of Failure (SPOF). No single point of failure is obvious with the blockchain-based decentralized architecture. If there are k $(k \geq 3)$ miners in crowdsourcing, more than $\lfloor k/2 \rfloor$ miners are honest and available in CrowdBC with the majority honest security assumption. According to the peer-to-peer architecture, even though there remains only one miner, requesters and workers can access the crowdsourcing service normally. Thus, CrowdBC is exempted from SPOF.

Pseudonymity. Unlike traditional crowdsourcing systems with some true identity in registration phase, which has the risk of user sensitive information leakage, CrowdBC utilizes the pseudonymous Bitcoin-like addresses to denote requesters and workers, which enables privacy-preserving without submitting true identity to finish a crowdsourcing task.

No Trusted Third Party. Requesters and workers could directly finish the crowdsourcing task and share their data without the intervention of any third party. CrowdBC stores task data and solutions in the distributed storage system, and utilizes time-locked deposit protocol and monetary penalty to ensure fairness between requesters and workers. Users are authorized to post or receive task under the tamper-resistant smart contract, which means that they cannot refute these behaviors even there is no third party.

DDoS and Sybil Attack Resistant. CrowdBC requires deposits from requesters and workers to thwart DDoS and Sybil attacks. In addition, users need to pay transaction fees (even though it's few) for miners who maintain the blockchain network. Therefore, malicious attackers may pay a huge cost to launch these attacks under the deposit-based mechanism (the main threat is that attackers have enough coins to launch attacks, which is not this paper concern currently).

Trustworthy Worker Selection. Reputation and expertise statistic value are important factors for workers receiving a task. High reputation and reliability mean high probability to receive the task. In CrowdBC, updating these values can only happen when worker really completes a crowdsourcing task in RWRC contract. USC contract cannot be created by workers themselves and updateReputation$(\cdot)$ function in USC can only be invoked by RWRC contract. In particular, RWRC contract needs to make the deposit and pay transaction fee. Therefore, if a malicious worker wants to brush his reputation or reliability, he needs to pay a high cost. In this way, we can anticipate that requesters can select proper workers by setting the reputation and reliability value and workers would work honestly and diligently.

7 Evaluation Results and Analysis

7.1 System Design

The primary goal of CrowdBC is to design a secure and decentralized crowdsourcing system. We implemented a software prototype on Ethereum public test network to test our proposed scheme and depicted the complex process of crowdsourcing by smart contract[1]. We evaluated the accuracy of CrowdBC by asking the workers to tag images with labels, which was a type of multi-labeling tasks. Extensions to other arbitrary tasks are also possible, requiring to change the evaluation function to evaluate the result accordingly.

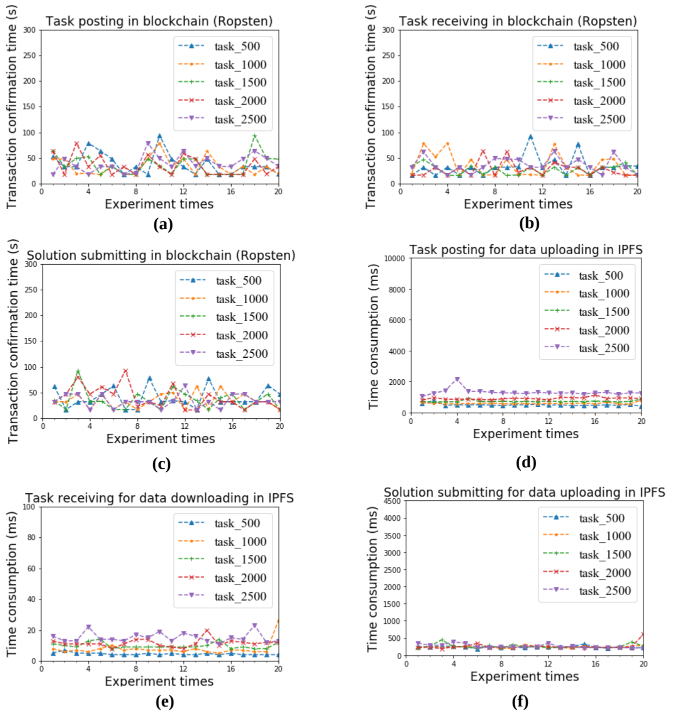
important

CrowdBC was implemented on official Ethereum public test network Ropsten with program language including solidity, java and javascript with roughly 9963 lines of codes, among which, only about 950 lines are for implementing smart contracts in solidity. Solidity is an object-oriented programming language designed for writing smart contracts in Ethereum. Besides, CrowdBC interacts with Ethereum based on web3j, a lightweight library for java applications on the Ethereum network. Especially, we constructed BCCompiler based on web3j. As above mentioned, each new registering of a user and new task could create a new contract. The task information was input by CrowdBC Client which was developed based on javascript. Then it was transformed into the contract, and compiled by BCCompiler. In order to reduce the storage requirement, we build the local IPFS proxy and store these images IPFS. There are 168 connected peers in IPFS currently.

7.2 Time Complexity

To evaluate the utility, security and performance of CrowdBC, we conducted 20 sets of experiments to process image tagging task on the CIFAR-10 dataset which consists of ten classes of images, such as airplane, automobile, bird and cat. The CIFAR-10 dataset contains five training batches and one test batch, each batch with 10000 images. For each set of experiment, we randomly choose images from the training batch to recognize images. Furthermore, 5 workers and 1 requester are registered and each registered user is assigned with 20 ETH coins. Each task is finished by 3 workers. We randomly selected 500, 1000, 1500, 2000, 2500 images from the training batch, which are identified by task_500, task_1000, task_1500, task_2000, task_2500. Further, task solutions are encrypted with requester's public key and saved in the data storage, thus no malicious users can decrypt and read them. In particular, we design evaluation function by majority voting to verify if the solutions belong to the ten classes of images. Moreover, the gas price is set at the average market price, about 20000000000 *Wei* [55] (*Wei* is the smallest subdenomination of ETH [37]). The cost of ETH is computed under the Ethereum gas rules by the formula: $COST_{ETH} = COST_{gas} * GAS_{price}/10^{18}$.

import
ant



(a) (b) (c) (d) (e) (f)

Fig. 8: The time complexity of task crowdsourcing in Ropsten.

We deployed URC, USC, RWRC contract in Ropsten. The average block time for mining in Ropsten is 10s. In our experiments, it consisted of 700 transactions, each of which was mined into one block with block number ranging from 3,213,538 to 3,219,692. As shown (a), (b) and (c) in Fig 8, the average transaction confirmation time (including the execution time in CrowdBC Client) on task posting, task receiving and solution submission were about 37.35s, 32.28s and 36.82s, respectively. Namely, each transaction was confirmed in about $3\tilde{4}$ block time.

If we did not provide enough transaction fees for our contracts, these transactions were waiting to be confirmed in the transaction pool. As shown in Fig 8 (d) and (f), the average data uploading time (including the execution time in CrowdBC Client) in Ropsten on task posting and solution submission were the positive correlation with the data size. However, if the data size was small enough (e.g., solution submission), the uploading time was almost the same (about 246.57ms). In addition, data was downloaded from multiple nodes simultaneously in IPFS, which made the data download very fast (about 13.27ms in task receiving). In particular,

[1] https://github.com/lim60/crowdBC

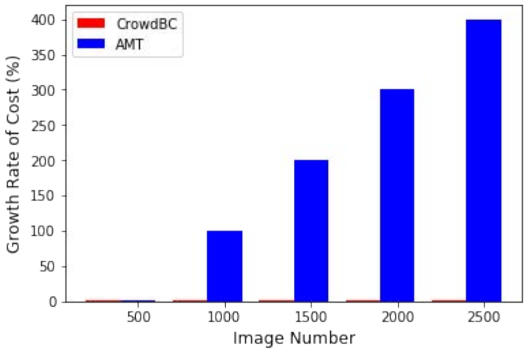


Fig. 9: The comparison of growth rate of cost in CrowdBC and AMT.

there existed some large points at task_2000 in Fig. 8 (f). We found that it was relevant with the uploading data size in the specified time within the whole IPFS system.

We completed part of set experiments to tag images and recorded the accuracy of image tagging by workers with different number of images in each task. The average accuracy was 93.22% on 7500 images, among the error results, these images were really unclear. This indicated the good utility of CrowdBC. Besides, we found that part of miners were offline in the experiments, but it did not affect the process of crowdsourcing and users could still normally post or receive the task, which demonstrates the feature of decentralization in our framework.

7.3 Task Cost

In our experiments, the average transaction fee was 0.011 ETH per 100 image. According to AMT reward policy, about 0.45$ is paid for each 100 images tagging or identifying task. It's worth noting that when we first designed the experiment in February 2017, 1 ETH price was about 12$, and the cost was acceptable (i.e.,0.14$). As the sharp rise of ETH price in May 2018, 1 ETH was up to 784.21$ according to the ETH market price [55], which was unpractical in real life. We can see that the cost was general lower than AMT platform by using the ETH price in February 2017, and the more image quantities had, the cheaper the cost was. Note that the cost (including transaction fee and gas) in CrowdBC has not changed a lot with the image number increased to 2500, while the cost in AMT added about 400% (Fig. 9). Therefore, we can conclude that CrowdBC is applicable for the task with large reward.

In sum, we conducted the whole crowdsourcing process with a practical example in CrowdBC, which illustrates that the blockchain-based framework is feasible. However, we realize that a more low-cost and public blockchain for CrowdBC is necessary. Hyperledger as a well known blockchain fabric might be a solution, which is also a future work of our framework.

8 Conclusion and Future Work

In this paper, we presented the design of CrowdBC, a blockchain-based decentralized framework for crowdsourcing. We analyzed that the traditional centralized crowdsourcing system suffers from privacy disclosure, single point of failure and high services fee. We formalized CrowdBC to handle these centralized problems. Meanwhile, we enhanced the flexibility of crowdsourcing by smart contract to depict complex crowdsourcing logic. A series of design algorithms based on smart contract were proposed to construct a concrete scheme under the framework. Besides, we evaluated our approach on Ethereum by implementing components providing decentralized crowdsourcing services.

TABLE 2: Time consumption on data uploading and downloading in IPFS

Name	Task Posting (Uploading)		Task Receiving (Downloading)		Solution Submission (Uploading)	
	Size (Kb)	Time (ms)	Size (Kb)	Time (ms)	Size (Kb)	Time (ms)
Task_500	1179.69	499.40	1179.69	4.55	4.7846	237.00
Task_1000	2357.45	607.54	2357.45	7.90	9.6877	238.15
Task_1500	3541.86	731.81	3541.86	13.85	15.0338	256.15
Task_2000	4723.76	906.05	4723.76	11.60	20.2921	258.70
Task_2500	5901.87	1308.65	5901.87	18.10	25.7626	478.90

We are still in the early stage of blockchain technology and identify several meaningful future works. First, we only implemented the basic process of crowdsourcing currently, but there exist much more complex scenes to handle. Second, designing an efficient evaluation mechanism is crucial in CrowdBC. We resume that requester could provide an evaluation function when posting the task. However, we should also consider that requester does not know about the solution, and thus giving an efficient evaluation function is becoming difficult.

Acknowledgments

This work was supported by National Key R&D Plan of China (Grant No. 2017YFB0802203), National Natural Science Foundation of China (Grant Nos. U1736203, 61732021, 61472165 and 61373158), Guangdong Provincial Engineering Technology Research Center on Network Security Detection and Defence (Grant No. 2014B090904067), Guangdong Provincial Special Funds for Applied Technology Research and development and Transformation of Important Scientific and Technological Achieve (Grant No. 2016B010124009), the Zhuhai Top Discipline–Information SecurityGuangzhou Key Laboratory of Data Security and Privacy Preserving, Guangdong Key Laboratory of Data Security and Privacy Preserving, China Postdoctoral Science Foundation funded project (Grant No. 2017M612842).

References

[1] J. Howe, "The rise of crowdsourcing," *Wired magazine*, vol. 53, no. 10, pp. 1–4, Oct. 2006.

[2] "Upwork," "https://www.upwork.com/", [Online].

[3] "Amazon mechanical turk," "https://www.mturk.com/mturk/welcome", [Online].

[4] "Uber," "https://www.uber.com/", [Online].

[5] "Elance and odesk hit by ddos," "https://gigaom.com/2014/03/18/elance-hit-by-major-ddos-attack-downing-service-for-many-freelancers/", [Online].

[6] "Uber china statement on service outage," "http://shanghaiist.com/2015/04/18/uber_chinese_operations_recently_hacked.php/", [Online].

[7] "Freelancer," "http://www.smh.com.au/business/freelancer-contests-20000-privacy-breach-fine-from-oaic-20160112-gm4aw2.html", [Online].

[8] 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.

[9] H. To, G. Ghinita, L. Fan, and C. Shahabi, “Differentially private location protection for worker datasets in spatial crowdsourcing,” *IEEE Transactions on Mobile Computing*, vol. 16, no. 4, pp. 934–949, 2017.

[10] G. Zhuo, Q. Jia, L. Guo, M. Li, and P. Li, “Privacy-preserving verifiable set operation in big data for cloud-assisted mobile crowdsourcing,” *IEEE Internet of Things Journal*, vol. 4, no. 2, pp. 572–582, 2017.

[11] B. Halder, “Evolution of crowdsourcing: potential data protection, privacy and security concerns under the new media age,” *Revista Democracia Digital e Governo Eletrônico*, vol. 1, no. 10, pp. 377–393, 2014.

[12] J. R. Kan Yang, Kuan Zhang, “Security and privacy in mobile crowdsourcing networks: challenges and opportunities,” *IEEE Communications Magazine*, vol. 53, no. 8, pp. 75–81, 2015.

[13] E. Toch, “Crowdsourcing privacy preferences in context-aware applications,” *Personal and Ubiquitous Computing*, vol. 18, no. 1, pp. 129–141, 2014.

[14] M. v. d. S. Yu Zhang, “Reputation-based incentive protocols in crowdsourcing applications,” in *2012 Proceedings IEEE INFOCOM*, Florida, USC, 2012, pp. 2140–2148.

[15] X. F. J. T. Dejun Yang, Guoliang Xue, “Crowdsourcing to smartphones: incentive mechanism design for mobile phone sensing,” in *Proceedings of the 18th annual international conference on Mobile computing and networking, Mobicom 2012*, Istanbul, Turkey, 2012, pp. 173–184.

[16] G. C. Dan Peng, Fan Wu, “Pay as how well you do: A quality based incentive mechanism for crowdsensing,” in *Proceedings of the 16th ACM International Symposium on Mobile Ad Hoc Networking and Computing, MobiHoc 2015*, Hangzhou, China, 2015, pp. 177–186.

[17] E. Toch, “Crowdsourcing privacy preferences in context-aware applications,” *Personal and ubiquitous computing*, vol. 18, no. 1, pp. 129–141, 2014.

[18] S. Zhang, J. Wu, and S. Lu, “Minimum makespan workload dissemination in dtns: Making full utilization of computational surplus around,” in *Proceedings of the fourteenth ACM international symposium on Mobile ad hoc networking and computing*. ACM, 2013, pp. 293–296.

[19] “Waze,” ”https://www.waze.com/”, [Online].

[20] “Freelancer,” ”https://www.freelancer.com/”, [Online].

[21] “Quirky,” ”http://siliconangle.com/blog/2015/12/14/bankruptcy-judge-approves-sale-of-quirky-assets”, ”[Online]”.

[22] P. Yang, Q. Li, Y. Yan, X.-Y. Li, Y. Xiong, B. Wang, and X. Sun, “friend is treasure: Exploring and exploiting mobile social contacts for efficient task offloading,” *IEEE Transactions on Vehicular Technology*, vol. 65, no. 7, pp. 5485–5496, 2016.

[23] M. H. Cheung, R. Southwell, F. Hou, and J. Huang, “Distributed time-sensitive task selection in mobile crowdsensing,” in *Proceedings of the 16th ACM International Symposium on Mobile Ad Hoc Networking and Computing*. ACM, 2015, pp. 157–166.

[24] A. S. Federico Ast, “The crowdjury, a crowdsourced justice system for the collaboration era,” 2015.

[25] V. Jacynycz, A. Calvo, S. Hassan, and A. A. Sánchez-Ruiz, “Betfunding: A distributed bounty-based crowdfunding platform over ethereum,” in *Distributed Computing and Artificial Intelligence, 13th International Conference*, vol. 474, Sevilla, Spain, 2016, pp. 403–411.

[26] H. Zhu and Z. Z. Zhou, “Analysis and outlook of applications of blockchain technology to equity crowdfunding in china,” *Financial Innovation*, vol. 2, no. 1, p. 29, 2016.

[27] F. Buccafurri, G. Lax, S. Nicolazzo, and A. Nocera, “Tweetchain: An alternative to blockchain for crowd-based applications,” in *International Conference on Web Engineering*. Springer, 2017, pp. 386–393.

[28] “Microwork,” ”http://www.microwork.io/”, ”[Online]”.

[29] “Gems,” ”https://icodrops.com/gems/”, ”[Online]”.

[30] S. Nakamoto, “Bitcoin: A peer-to-peer electronic cash system,” 2009.

[31] A. Wright and P. De Filippi, “Decentralized blockchain technology and the rise of lex cryptographia,” 2015.

[32] “Wikipedia. list of cryptocurrencies,” ”https://en.wikipedia.org/wiki/List_of_cryptocurrencies”, [Online].

[33] R. S. M. J. F. Muneeb Ali, Jude Nelson, “Blockstack: A global naming and storage system secured by blockchains,” in *USENIX Annual Technical Conference, USENIX ATC 2016*, Denver, CO, 2016, pp. 181–194.

[34] T. V. Asaph Azaria, Ariel Ekblaw, “Medrec: Using blockchain for medical data access and permission management,” in *2nd International Conference on Open and Big Data, OBD 2016*, Vienna, Austria, Aug. 2016, pp. 25–30.

[35] J. C. A. N. J. A. K. E. W. F. Joseph Bonneau, Andrew Miller, “Sok: Research perspectives and challenges for bitcoin and cryptocurrencies,” in *IEEE Symposium on Security and Privacy, S&P 2015*, CA, USA, May. 2015, pp. 17–21.

[36] D. Christian and W. Roger, “Information propagation in the bitcoin network,” in *Peer-to-Peer Computing (P2P), 2013 IEEE Thirteenth International Conference on*. IEEE, 2013, pp. 1–10.

[37] W. Gavin, “Ethereum: A secure decentralised generalised transaction ledger,” *Ethereum Project Yellow Paper*, vol. 151, 2014.

[38] S. Melanie, *Blockchain: Blueprint for a new economy*. ”O’Reilly Media, Inc.”, 2015.

[39] N. Szabo, “Formalizing and securing relationships on public networks,” *First Monday*, vol. 2, no. 9, 1997.

[40] “Hyperledger white paper (2015),” ”www.the-blockchain.com/docs/Hyperledger\%20Whitepaper.pdf”, [Online].

[41] J. Benet, “Ipfs-content addressed, versioned, p2p file system,” *arXiv preprint arXiv:1407.3561*, 2014.

[42] D. M. Petar Maymounkov, “Kademlia: A peer-to-peer information system based on the xor metric,” in *International Workshop on Peer-to-Peer Systems*, vol. 2429, MA, USA, March 2002, pp. 53–65.

[43] A. T. M. S. T. J. N. Robin Wentao Ouyang, Lance M. Kapla, “Parallel and streaming truth discovery in large-scale quantitative crowdsourcing,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 10, pp. 1045–9219, Oct. 2016.

[44] C. Miao, W. Jiang, L. Su, Y. Li, S. Guo, Z. Qin, H. Xiao, J. Gao, and K. Ren, “Cloud-enabled privacy-preserving truth discovery in crowd sensing systems,” in *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems*. ACM, 2015, pp. 183–196.

[45] X. Z. Depeng Dang, Ying Liu, “A crowdsourcing worker quality evaluation algorithm on mapreduce for big data applications,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 7, pp. 1879–1888, July 2016.

[46] H. Z. B. Y. Z. Gang Wang, Tianyi Wang, “Man vs. machine: Practical adversarial detection of malicious crowdsourcing workers,” in *Proceedings of the 23rd USENIX Security Symposium. Usenix Security 2014*, vol. 14, San Diego, CA, 2014.

[47] B. W. Amit Sahai, “How to use indistinguishability obfuscation: deniable encryption, and more,” in *Proceedings of the forty-sixth annual ACM symposium on Theory of computing, STOC 2014*, New York, USA, 2014, pp. 475–484.

[48] Y. Kan, Z. Kuan, R. Ju, and S. Xuemin, “Security and privacy in mobile crowdsourcing networks: challenges and opportunities,” *IEEE Communications Magazine*, vol. 53, no. 8, pp. 75–81, 2015.

[49] D. M. Marcin Andrychowicz, Stefan Dziembowski, “Secure multiparty computations on bitcoin,” in *IEEE Symposium on Security and Privacy, S&P 2014*, San Jose, CA, 2014, pp. 443–458.

[50] T. Ruffing, A. Kate, and D. Schröder, “Liar, liar, coins on fire!: Penalizing equivocation by loss of bitcoins,” in *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2015, pp. 219–230.

[51] X. Yu, M. T. Shiwen, Y. Li, and R. D. Huijie, “Fair deposits against double-spending for bitcoin transactions,” in *Dependable and Secure Computing, 2017 IEEE Conference on*. IEEE, 2017, pp. 44–51.

[52] R. Kumaresan and I. Bentov, “How to use bitcoin to incentivize correct computations,” in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*. ACM, 2014, pp. 30–41.

[53] I. Bentov and R. Kumaresan, “How to use bitcoin to design fair protocols,” in *International Cryptology Conference*. Springer, 2014, pp. 421–439.

[54] J. Garay, A. Kiayias, and N. Leonardos, “The bitcoin backbone protocol: Analysis and applications,” in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2015, pp. 281–310.

[55] “Ethereum market,” ”https://etherscan.io/”, [Online].

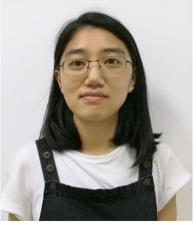
Ming Li received his B.S. in electronic information engineering from University of South China in 2009, and M.S. in information processing from Northwestern Polytechnical University in 2012. From 2012 to 2015, he worked in Huawei Technologies Co., Ltd. He is currently a Ph. D. student at Jinan University. His research interests include crowdsourcing, blockchain and its privacy and security.

Jian Weng is a professor and the Executive Dean with College of Information Science and Technology in Jinan University. He received B.S. degree and M.S. degree at South China University of Technology in 2001 and 2004 respectively, and Ph.D. degree at Shanghai Jiao Tong University in 2008. His research areas include public key cryptography, cloud security, blockchain, etc. He has published 80 papers in international conferences and journals such as CRYPTO, EUROCRYPT, ASIACRYPT, TCC, PKC, CT-RSA, IEEE TDSC, etc. He also serves as associate editor of IEEE Transactions on Vehicular Technology.

Anjia Yang received the B.S. degree from Jilin University in 2011, and the Ph.D. degree from the City University of Hong Kong in 2015, respectively. He is currently a postdoctoral researcher in Jinan University, Guangzhou. His research interests include blockchain security, RFID security and privacy, applied cryptography, and cloud computing.

Wei Lu received the B.S. degree in Automation from Northeast University, China in 2002, the M.S. degree and the Ph.D. degree in Computer Science from Shanghai Jiao Tong University, China in 2005 and 2007 respectively. He was a research assistant at Hong Kong Polytechnic University from 2006 to 2007. He is currently an Associate Professor with the School of Data and Computer Science, Sun Yat-sen University, Guangzhou, China. His research interests include multimedia forensics and security, signal processing, computer vision and machine learning.

Yue Zhang received his B.S. in information security from Xi'an University of Posts & Telecommunications in 2013, and M.S. in information security from Xi'an University of Posts & Telecommunications in 2016. From 2016, he started his Ph. D. at Jinan University. His research interests include blockchain, system security and Android security.



Lin Hou is a Ph.D. student in Jinan University. She received her bachelor degree from Wuhan University of Technology and a dual degree from Huazhong University of Science and Technology. Her research mainly focuses on asymmetric cryptography and privacy.

Jia-Nan Liu is a Ph.D. student of Jinan University. He was born in July, 1992. He received B.S. degree and M.S. degree at Zhengzhou University and Jinan University in 2013 and 2016 respectively. His research interesting includes cryptography and cloud computing security.

Yang Xiang received his PhD in Computer Science from Deakin University, Australia. He is currently the Dean of Digital Research & Innovation Capability Platform, Swinburne University of Technology. He is the Director of the Network Security and Computing Lab (NSCLab) and the Associate Head of School (Industry Engagement). He is the Chief Investigator of several projects in network and system security, funded by the Australian Research Council (ARC). His research interests include network and system security, distributed systems, and networking. He serves as an Associate Editor of the IEEE TRANSACTIONS ONCOMPUTERS, IEEE-TRANSACTIONS ONPARALLEL AND DISTRIBUTED SYSTEMS, and Security and Communication Networks, and an Editor of the Journal of Network and Computer Applications.

Robert H. Deng obtained his M.S. Degree from National University of Defense Technology, China, in 1981, and B.S. and Ph.D. Degrees from Illinois Institute of Technology in 1983 and 1985 respectively. He has been a professor at the School of Information Systems, Singapore Management University since 2004. Prior to this, he was the principal scientist and manager in the Infocomm Security Department, Institute for Infocomm Research, Singapore. His research interests include data security and privacy, multimedia security, network and system security. He was an associate editor of the IEEE Transactions on Information Forensics and Security from 2009 to 2012. He is currently an associate editor of the IEEE Transactions on Dependable and Secure Computing, and member of Editorial Board of the Journal of Computer Science and Technology (the Chinese Academy of Sciences), and the International Journal of Information Security (Springer), respectively.