Hybrid Blockchain Design for Privacy Preserving Crowdsourcing Platform

Saide Zhu, Huafu Hu, Yingshu Li, Wei Li
Department of Computer Science, Georgia State University, Atlanta, GA, USA
Email: *szhu5@student.gsu.edu, hhu4@student.gsu.edu, yili@gsu.edu, wli28@gsu.edu*

***Abstract*—Blockchain has been treated as one of the most promising technologies to promote crowdsourcing by providing new nice features, such as decentralization and accountability. Unfortunately, some inherent limitations of blockchain have been rarely addressed by the most existing works when applying blockchain into crowdsourcing, which becomes the performance bottleneck of crowdsourcing systems. In this paper, we propose a novel hybrid blockchain crowdsourcing platform to achieve decentralization and privacy preservation. Our platform integrates with a hybrid blockchain structure, dual-ledgers, and dual-consensus algorithms to ensure secure communication between the requesters and the workers. Moreover, the smart contract and zero-knowledge proof are employed to ensure automatic operation of the tasks and the protection users' privacy, respectively. Finally, we conduct experiments to confirm the efficiency of the adopted consensus algorithm on our platform by comparing it with the state-of-the-art.**

***Index Terms*—hybrid blockchain, crowdsourcing, smart contract, privacy preservation.**

I. Introduction

Crowdsourcing that outsources tasks to a non-specific, and usually large, mass network has already been popular and widely applied for years [1]. A typical example is Wikipedia, which allows everyone to join the platform and answer specific questions, and thus solving diverse customer needs [2]. Another application is Uber, which crowds the passengers' post and sends to the social idle drivers' clients [3]. Recently, Amazon also provides its crowdsourcing marketplace called Amazon Mechanical Turk (MTurk) to help companies to harness intelligence from a global view and enable workers to work on their interested tasks [4]. As an advanced paradigm, crowdsourcing is destined to have a bright future for important real-world applications.

Although crowdsourcing can solve many problems at present, research shows that there are still many issues in the traditional crowdsourcing model. First, the traditional crowdsourcing model relies on a centralized trusted third-party platform, which directly leads to a single point of failure. For instance, Wikipedia suffered from 8 service outrage in 2018, and thus users could not access the data from the server [5]. Second, the centralized platform masters all the transactions, which may bring silently misbehave problems. Third, when there is a conflict of opinions between the task requester and the worker, the problems of Free-riding (receive rewards without making real efforts) and False-reporting (dishonest requesters try to repudiate the payment) would happen in the system [6]. Last but not least, during the communications between task requesters and workers, their sensitive information is likely to suffer the threat of privacy leakage; for instance, Uber disclosed in Nov 2017 due to 57 million driver's information had been hacked [7]. To tackle the aforementioned issues, many countermeasures have been proposed. For examples, in [8], [9], the k-anonymity algorithm and differential privacy model were adopted to prevent the disclosure of sensitive user information during the transaction process; and in [10], [11], the reputation systems were introduced to solve the problems of free-riding and false-reporting in crowdsourcing. All of these studies cannot solve the above mentioned issues for the current crowdsourcing systems at the same time.

With the rapid development of blockchain technology, smart contracts with many cryptographic tools can be incorporated into blockchain networks. Thus, by exploiting blockchain technology, the privacy issues in the traditional crowdsourcing systems can be solved more effectively and efficiently. The prior works [12]–[14] proposed different solutions for location leakage problems. The identity authentication systems were designed by [15], [16] to provide a security guarantee for users who join the blockchain. In [14], the combination of public and private chains is used to provide private service for some workers, preventing privacy leakage. However, it is worthy of deeper discussion on the authority of the agent entity. Nevertheless, all of these existing works ignore the scalability and transaction verification efficiency of the blockchain network, limiting the performance of the crowdsourcing systems.

Thus, for the purpose of performance improvement, the objective of this paper is to develop a distributed crowdsourcing platform to securely and flexibly process user access control and task management using blockchain technology. Unavoidably, here are some challenges in building a blockchain-embedded crowdsourcing platform: (i) The traditional consensus protocols, *e.g,*,Proof-of-Work (PoW), in the blockchain network may have the scalability bottleneck and cause a lot of energy consumption [17]. Thus, adopting suitable consensus protocols is expected to speed up transaction verification rate. (ii) In the traditional blockchain networks, all transactions are public for providing accountability and transparency, which however, may leak users' private information. There, the challenge lies in co-existence of privacy and transparence in blockchain-based crowdsourcing. (iii) On a hybrid blockchain platform, the interactions between different blockchains and the ledger management should be well designed.

Authorized licensed use limited to: UNIVERSIDADE ESTADUAL DE MARINGA. Downloaded on July 14,2025 at 17:04:21 UTC from IEEE Xplore. Restrictions apply.

In order to address these challenges, in this paper, we propose to carry out the following activities. We first develop a framework of the hybrid blockchain-embedded crowdsourcing platform consisting of a public chain and multiple private subchains. In particular, the subchains are dynamically established to satisfy the needs of private tasks, and the information of private answers is kept secretly on the subchains but can be verified on the public chain by exploiting dual-ledgers and zero-knowledge proof. Then, to enhance transaction verification throughput, Delegated Proof of Stake (DPOS) and Practical Byzantine Fault Tolerance (PBFT) consensuses are respectively deployed on the public chain and the subchains, and smart contracts of the public chain and the subchains are designed accordingly. Finally, intensive experiments are set up to evaluate the performance of our proposed platform. The contributions of this paper are summarized as follows.

- A novel hybrid blockchain platform is proposed for distributed crowdsourcing, in which transaction privacy and transparency can be effectively balanced.
- The transaction verification rate is greatly increased by deploying DPOS and PBFT consensuses, reducing energy consumption in crowdsourcing.
- The smart contracts are developed on both public chain and subchains to control user access as well as to protect user privacy.
- Experiments are conducted to evaluate the effectiveness of the applied consensus algorithms in terms of execution time and throughput.

The remainder of this paper is organized as follows. The existing related works are presented in Section II. In Section III, the preliminaries are introduced. Our hybrid blockchain platform and the implementation process are proposed in Section IV and Section V, respectively. The experiments are conducted in Section V-F. At last, this paper is concluded in Section VII.

II. Related Work

In this section, we briefly summarize the most related works and discuss how they enlighten us to build a hybrid blockchain-based crowdsourcing platform.

One of the recent hot topics in mobile crowdsourcing is incorporating blockchain into the crowdsourcing platform for performance improvement. Li *et al.* proposed a blockchain-based decentralized crowdsourcing framework and implements several smart contracts to show the effectiveness of their framework [18]. However, the uploaded information from system users is recorded on the blockchain in public; thereby the privacy issue is not considered. Lu *et al.* designed a private and anonymous decentralized crowdsourcing platform called ZebraLancer which increases the privacy level of the crowdsourcing system but may cause a big data load of the platform, so the performance of the system's efficiency is still debatable [15]. In [19], blockchain and smart contracts were used to orchestrate the interactions between mobile crowdsensing providers and mobile users, but this work focused on the smart execution of incentive mechanism not on task processing. Zou *et al.* proposed a Proof-of-Trust consensus on hybrid blockchain to meet the practical requirements in service industry [20]. Their proposed consensus structure is similar to Hyperledger Fabric, which can greatly increase the transaction verification speed while still lacks privacy-preserving mechanisms integrated into the system. Jia *et al.* proposed a hybrid blockchain platform to achieve secure transaction and location privacy preservation in crowdsourcing. However, the centralization problems on their agent nodes still need more discussion [14].

III. Preliminary

A. DPOS Consensus

The core of the blockchain is the consensus protocol. The first generation of digital currency blockchain represented by Bitcoin that adopts the Proof-of-Work (POW) consensus [21]. POW guarantees that every node in the network can get fair reward corresponding to their effort made through mining. However, due to the large energy consumption and slow transaction verification speed, it is difficult to meet the needs of enterprise-level blockchain networks. Therefore, in this paper, we adopt the Byzantine Fault Tolerance (BFT)-based Delegated Proof of Stake (DPOS) consensus protocol [22] on the public chain.

In DPOS consensus, there are three roles, including stakeholders, witnesses (block producers or validators), and alternative nodes. Stakeholders are holders of tokens who have the right to vote for witnesses in the network. At the beginning of every time cycle, the stakeholders re-elect the witnesses in the network. Witnesses are super-nodes in the DPOS blockchain network and verify the previous transactions and package the newly generated transactions into blocks. For example, in the EOS network, there are 21 witnesses nodes that take turns to share the accounting rights of the blockchain every 3 seconds. The remaining nodes in the network are called alternative nodes. Although the alternative nodes do not make transaction verification, they still need to update the latest information of the ledgers in time for next round of witnesses election. Since the witnesses are voted by the stakeholders, it replaced the POW mining mechanism that needs every node to work for a computational puzzle. As a result, there is a big reduction in energy consumption. Witnesses' faster block generating rate also drastically reduces the time to complete a transaction's verification. At the same time, the rotation of accounting rights of the witnesses ensures the distributed characteristics of the network. It is also worth mentioning that when a witness fails or does something evil, the stakeholders can re-elect the witnesses to discharge the abnormal nodes. Similar to POW, DPOS also treats the longest chain as the only legal one. So, when the fork appears, a chain with more witnesses working on it is treated as legal. If all the witnesses work on the same chain at the same time, the blocks produced at full speed. The BFT mechanism in DPOS requires that the number of malicious or faulty nodes in the network does not exceed one-third of the total number of nodes. So that the evil node cannot reach the block generating speed of honest nodes even if they

Authorized licensed use limited to: UNIVERSIDADE ESTADUAL DE MARINGA. Downloaded on July 14,2025 at 17:04:21 UTC from IEEE Xplore. Restrictions apply.

all work on the same fork. Therefore, the security of DPOS is guaranteed. Many enterprise applications that adopt the DPOS consensus, such as EOS.io [23] and Bitshares [24], have also shown its robustness in practice.

In addition to the advantages of DPOS in verification speed and energy consumption, DPOS provides the finality feature. When a block receives an acknowledgment from $2n/3 + 1$ nodes (n the number of total nodes in the network), the block is permanently and legally written into the ledger and will no longer be replaced by a longer chain.

B. PBFT Consensus

Another consensus that can tolerate Byzantine failures is Practical Byzantine Fault Tolerance (PBFT). The algorithm was proposed by Castro and Liskov in 1999, which enhances the performance of the original Byzantine fault-tolerant algorithm. Each node in the blockchain network shares the same ledger; essentially the blockchain is a replicated state machine. Therefore, in recent years, the PBFT algorithm has been applied to the blockchain applications more widely as a consensus protocol, such as Zilliqa [25] and Hyperledger [26].

First, the network is required to dynamically select a leader node, and all nodes in the network can participate in the election. The election process is repeated in a round-robin manner. After the leader node gets selected, the leader needs to send a heartbeat message to the peer nodes to prove its liveness. If one or more nodes in the network do not receive the heartbeat message, a new round of leader elections is performed to ensure the stability of the network.

Roughly speaking, PBFT consensus algorithm can be divided into five steps: request, pre-prepare, prepare, commit and reply. In the request phase, a client sends a request to the leader node. When receiving the request, the leader broadcasts the pre-prepare massage to inform the entire network that he is the current leader, and sends the client's request to other peer nodes. All replica nodes authenticate the pre-prepared messages when receiving them. After accepting $2f + 1$ pre-prepared information, the peer nodes multi-cast a prepare message to all other nodes, where f is the number of Byzantine Failure nodes in the network. In the commit phase, each replica node verifies whether more than $2f + 1$ of identical requests have been received. Once the acceptance condition of the commit message is satisfied, the replica node writes a commit message to the message log and accepts the command for execution. In the last phase, the nodes execute the request and send a reply to the client. And the client should wait for more than $f + 1$ replies with the same result from different nodes to ensure the correctness of the execution.

PBFT consensus, in theory, is more efficient than Proof-of-Work consensus and can reach a high transaction verification speed when the transaction volume is not too high. Another advantage of PBFT consensus is that the transaction finality can be achieved more easily without waiting for multiple confirmations. Once the transactions are included in a block, these transactions are immediately considered finalized and no one can change them. Last but not least, PBFT can achieve the network agreement without intensive computations, which means the energy consumption of executing the PBFT consensus is relatively lower.

C. Smart Contract

The smart contract was introduced at the blockchain 2.0 stage. It was first implemented based on Ethereum platform with *Solidity* language. The smart contract can be written in the form of code and stored on a blockchain. Once the terms of the contract trigger a condition, the code is automatically executed, which saves a lot of artificial communication and supervision costs. Since the smart contract is stored on a blockchain, it inherits some nice properties, such as the features of accountability and decentralization. Once a smart contract is written, it can be trusted by the users. The blockchain network ensures that the terms of the contract cannot be changed.

When combined with the crowdsourcing systems, a blockchain platform should have the ability to process the crowdsourcing data and assign different permissions to various entities. Thus a programmable interface should be public to developers, which can be accomplished via the smart contract.

D. Zero Knowledge Proof

With the widespread use of smart contracts, the privacy preservation problem for the implementation of smart contracts has become an emerging topic. Developers are trying to deploy general cryptography methodology in smart contracts, including the compilers for multi-party computation [27], [28] and zero-knowledge proof [29], [30].

Among these approaches, ZK-proof allows a prover to generate encrypted evidence, and a verifier performs a verification of a private state by performing a generated calculation on the witness without leaking any relevant information. Inspired by platforms such as Hawk [31], Zcash [32], and Ethereum, in this paper, we also use Zero-Knowledge Succinct Non-Interactive Argument of Knowledge (ZK-SNARK) to build optimized ZK-proof to help verify transactions between the public chain and the subchains.

Specifically, the ZK-SNARK algorithm on the blockchain contains three modules. First, there is a setup algorithm generating a public parameter to establish the SNARK. Then, the prover module uses the SNARK to generate an attestation to verify the authenticity of a statement. Finally, the correctness of the proof can be verified by calling the verifier module on the chain.

IV. Platform Framework

Our crowdsourcing platform is built based on a hybrid blockchain network that consists of a public chain and multiple subchains, in which public tasks are posted on the public chain and private tasks are uploaded to the subchains for privacy protection. The platform framework is presented in Fig. 1.

On the platform, there are three types of entities, including *requester*, *worker*, and *validator*, each of which is introduced in the following.

Authorized licensed use limited to: UNIVERSIDADE ESTADUAL DE MARINGA. Downloaded on July 14,2025 at 17:04:21 UTC from IEEE Xplore. Restrictions apply.

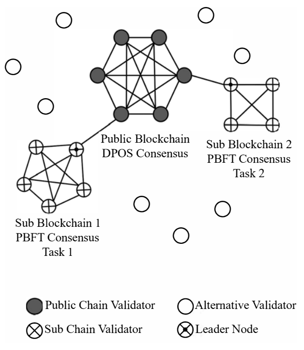

Fig. 1. Framework of our blockchain-embedded crowdsourcing platform

- **Requester:** Requesters can announce tasks, collect answers from workers, and make payments to workers. Each requester is required to make a deposit to the blockchain via smart contract. The deposit is used to reward the workers who process tasks as well as the validators who verify transactions. It is worth emphasizing that any requester cannot register as a validator for the blockchain network, but can potentially become a worker. The requesters have the rights to vote for public-chain validators and subchain validators in which they post their private tasks on the corresponding subchains.
- **Workers:** Workers process requesters' tasks and obtain payments from the requesters. If a worker implements a public task, his answers are uploaded to the public chain; otherwise, his answers are uploaded to a subchain. Each worker has the right to vote for public-chain validators and subchain validators in which he implements tasks on the corresponding subchains. Any worker can not register as a validator, but can potentially become a requester.
- **Validators:** In the blockchain network, there is a set of nodes working as validators which can be classified into three categories according to their duties, *i.e.*, *public-chain validator*, *alternative validator*, and *subchain validator*. (i) Public-chain validators work as the verification nodes for the public chain, which take turns to get the accounting right. They provide a verification service for all public transactions and share a public-chain ledger. In every time cycle, the public validators may change. (ii) Alternative validators are those nodes who are not selected as the public-chain validators. They do not have the right to produce blocks, but still need to share the public-chain ledger. Alternative validators have the potential to be selected as the subchain validators. (iii) Subchain validators are elected from current alternative validators. They perform transaction verification for subchains where the private tasks are posted and store the information of both the public-chain and the subchain ledgers.

In order to join the blockchain-embedded crowdsourcing system, the above three entities should first register as the legitimate members of the platform. On a public chain, everyone can join and generate transactions, greatly increasing the probability of being attacked by malicious platform users. Although the blockchain can ensure that all transactions during the transmission process are recorded and cannot be falsified, the authenticity of entities' identities cannot be verified, so the blockchain is vulnerable to Sybil attack. Therefore, in this paper, we employ a trusted third-party Certificated Authority (CA) to verify entities' identities and issue certificates. Only the entities that have obtained the legal certificates can join the crowdsourcing platform.

Each requester can post public and/or private tasks on the platform. If a task is public, the validators pass the task information to the workers on the public chain; otherwise, the validators send the task information to the workers on the subchain according to the smart contracts. Then, the workers choose tasks they are interested to implement. When a public task is completed, the workers submit the answers to the public-chain validators for verification, and the transaction information is recorded in the public-chain ledger. When a private task is completed, the workers send the answers to the subchain validators for verification, and the transaction information is recorded in the subchain ledger. After collecting the answers, the leader on a subchain extracts the Merkel roots in all block headers, forms a new Merkel tree with all the extracted block Merkel roots as leaf nodes, generates a zero-knowledge proof commitment, and publish it to the public chain. Since the subchain validators on the same subchain share the same ledger, this generated commitment can be checked to prevent the leader node from being malicious. In this way, the public-chain ledger records the transaction information of all public tasks and the commitment information of private tasks on subchains while preserving the answers of all private tasks. Meanwhile, each subchain copies the public-chain ledger and records all the transaction information of its subchain tasks. This information can only be accessed through the permission of the smart contracts on the subchains. When the answer collection is completed, the rewards are distributed to the workers based on the reward policy in the smart contracts, and the extra deposits are returned to the corresponding requesters' accounts.

V. Implementation Process

On our blockchain-embedded crowdsourcing platform, there are six major phases of the implementation, including: *Identity Authetication*, *Public Chain Setup*, *Subchain Setup*, *Smart Contract Deployment*, *Zero-Knowledge Poof*, which are illustrated in this section.

A. Identity Authentication

Identity authentication has always been a critical part of crowdsourcing [16]. To ensure the success rate of task completion, a trusted third-party CA is adopted to verify the identities of entities who intend to participate into the crowdsourcing platform as well as to issue certificates to indicate their legitimacy.

At the beginning, each entity generates a pair of keys (pk_{ut}, sk_{ut}) using RSA algorithm, where $ut \in \{r, w, v\}$ to

Authorized licensed use limited to: UNIVERSIDADE ESTADUAL DE MARINGA. Downloaded on July 14,2025 at 17:04:21 UTC from IEEE Xplore. Restrictions apply.

indicate the type of entity (*i.e.*, r, w, and v represent requester, worker, and validator, respectively.), pk_{ut} is the entity's public key, and sk_{ut} is the entity's private key. Then, the entity sends the encrypted message $E(pk_c, pk_{ut}, ID_{ut}, Sig(sk_{ut}, ID_{ut}))$ to CA for registration, in which pk_c is the public key of CA, ID_{ut} is the entity's real identity information, and $Sig(sk_{ut}, ID_{ut})$ is the entity's digital signature.

When receiving an entity's registration $E(pk_c, pk_{ut}, ID_{ut}, Sig(sk_{ut}, ID_{ut}))$, CA verifies whether ID_{ut} and pk_{ut} have been stored in his historic information as each entity is allowed to use one public key to avoid sybil attack. If ID_{ut} and pk_{ut} pass the verification, CA generates a unique certified identity ID^c_{ut} and a certification $Cert_{ut}\{ID^c_{ut}, ut, pk_{ut}\}$, and returns the encrypted certification $E(pk_{ut}, Cert_{ut}, Sig(sk_c, Cert_{ut}))$ to the entity, where sk_c and $Sig(sk_c, Cert_{ut})$ are respectively the private key and digital signature of CA. Only the entities who receive the certifications can join the blockchain-embedded crowdsourcing platform as legitimate users.

B. Public Chain Setup

Once all the validators complete their identity registration, they start to establish a public chain by running BFT-based DPOS consensus [22]. The registered requesters and workers are the stakeholders in the blockchain network and vote for the public-chain validators. In DPOS, the voting mechanism is performed round by round. When malicious behavior of any public-chain validator is detected or the deadline of a round reaches, the current requesters and workers need to re-vote for the public-chain validators. During each round of voting, the 21 validators with the most votes become the public-chain validators in the system, rotate the accounting rights, and help verify the public tasks in this round. The remaining validators become the alternative validators who share the public-chain ledger and wait for the voting process of the subchain validators.

C. Subchain Setup

For each private task, a subchain is temporarily created on the crowdsourcing platform to process answer collection and transaction verification by running the PBFT consensus [33], [34]. The subchain validators are elected from the existing alternative validators. On each subchain, the corresponding requester and workers are voters for the subchain validators. According to the PBFT consensus protocol [33], to resist Byzantine failure attack, the total number of nodes should be larger than 3 times of the number of the Byzantine failure nodes in the blockchain network. In other words, to guarantee the Byzantine Fault Tolerance for the platform, there must have at least 4 subchain validators on each subchain. If the number of workers on a subchain is larger than a system threshold τ, the subchain validators are elected by the requester and corresponding workers, and the number of the validators is determined by the system based on the expected transaction volume; if the number of workers on a subchain is smaller than τ, they may not be able to pick sufficient validators to achieve Byzantine Fault Tolerance. In this situation, 4 validators are enough to meet the maximum transaction throughput of the subchain. Thus, the smart contract will randomly select 4 subchain validators from the existing alternative validators.

When the private task is completed, the corresponding subchain validators will be released and become the alternative validators on the public chain.

D. Task Announcement & Response

The information of each task is posted on the platform in the form of $Task\{T, ID_t, t_1, t_2, R, Num, p, pk_r, cert_r, Sig(sk_c, Cert_r)\}$, in which $T \in \{public, private\}$ denotes the type of the task, ID_t is the ID number of the task, t_1 is the deadline for the workers to make response to the task announcement, t_2 is the deadline for the workers to submit their answers, R is the total amount of rewards for completing the task, Num is the maximum number of desired answers for the task, p implies the task implementation preference (such as location, working duration, and data quality) that is used by the task requester to select desired workers, pk_r is the task requester's public key, $cert_r$ is the task requester's certification.

If a worker is interested in some tasks, he should make a response before the deadline in the form of $Res\{ID_t, ID^c_w, t_r, cap, pk_w, cert_w, Sig(sk_c, Cert_w)\}$. Specifically, the response information includes the task identity ID_t, the worker's registered identity ID^c_w, the response time t_r, the work's capacity cap (such as location, available time, and quality of offered data), the worker's public key pk_w, and the worker's certification $Cert_w$ and digital signature. In addition, the worker's certification can be verified with pk_c, $Cert_w$, and $Sig(sk_c, Cert_w)$.

When receiving a worker's response message, each requester first checks whether the reply time t_r is prior to the deadline t_1, *i.e.*, ensuring that $t_r \leq t_1$, and whether the worker has a valid certification. If the worker passes such verification process, a permission $permit(ID_t, ID^c_w, cert_w)$ is issued to the worker and is replied in the form of $E(pk_w, permit, Sig(sk_r, permit))$, indicating that the worker can carry out the task. Then, the worker needs to complete the task and submit the answer before its deadline in the form of $Ans(T, ID_t, A_{ID^c_w}, ID^c_w, t_a, permit)$, where $A_{ID^c_w}$ records the answer contents, and t_a is the submission time.

On the platform, the methods of recording task answers in the blockchain are determined according to the type of tasks. If a task belongs to the public type, the answers are directly verified by the public-chain validators and recorded in the public-chain ledger; otherwise, the answers are submitted via temporary subchains.

E. Smart Contract Deployment

Different from Bitcoin's public chain, smart contracts are deployed in the proposed hybrid blockchain system to constrain entities' behaviors and provide them with different permissions. In this subsection, the deployment of smart contracts on the platform are detailed.

Authorized licensed use limited to: UNIVERSIDADE ESTADUAL DE MARINGA. Downloaded on July 14,2025 at 17:04:21 UTC from IEEE Xplore. Restrictions apply.

1) Task Announcement & Response Process: When a requester publishes a task, he needs to make a deposit to the smart contracts. The blockchain system checks whether the task is issued by a legitimate requester and has enough amount of deposit that should be greater than the total task rewards. If these conditions are satisfied, the task information will be flooded to the online workers. When the workers reply to the tasks, the smart contracts verify the workers' certifications and their response time. Next, for valid responses, the smart contracts are executed according to the type of tasks.

2) Answer Collection & Reward Distribution on Public Chain: When receiving an answer $Ans(T, ID_t, A_{ID^c_w}, ID^c_w, t_a, permit)$ of a public task, the smart contracts should ensure that i) the answer is submitted by a permitted worker by verifying $permit$; ii) the answer $A_{ID^c_w}$ is not repeated in the answer pool $Pool_a$; and iii) the answer is submitted on time, *i.e.,* $t_a \leq t_2$. If the three constraints simultaneously hold, the answer is treated as valid. If the number of valid answers exceeds the value of Num, only the first Num valid answers are recorded into blocks; otherwise, all the valid answers are recorded. If a task belongs to the private type, an attestation π_a is sent from the leader of the subchain for verification. Then, the smart contract runs ZK-SNARK verifier to check the correctness of tack completion and records it into the public-chain ledger.

Once the answer collection stage is finished, the smart contracts compute the reward of each answer and generate a zero-knowledge proof π_r with the task requester's private key sk_r as the witness to commit the successful running of the reward distribution policy. Also, a validation fee should be paid to validators in the blockchain network. In this paper, we suppose that the requesters are responsible for paying such validation fee. In reality, different task requesters may adopt different reward distribution policies and various algorithms to charge validation fee. After paying rewards and validation fee, the rest of the deposits are returned back to the requesters' account. But, if a malicious requester is detected (*e.g.*, posting tasks without valid certification), his deposit will be distributed to the corresponding workers.

3) Answer Collection & Reward Distribution on Subchain: Since the subchain validators do not need to help verify the transactions of the public chain, they only copy the transaction records from the public-chain ledger and record detailed private answers on the subchain ledgers. It is worth mentioning that the answers of all private tasks are not visible on the public chain, but these answers need to be verified on the public chain. Accordingly, on each subchain, the leader selected by the PBFT consensus can utilize zero-knowledge proof by generating a commitment on all subchain blocks of this task and records it on the public chain, which will be described in Section V-F.

On each subchain, the procedures of answer collection and reward distribution are similar to those on the public chain. Furthermore, to achieve a better access control feature, an access list model can be added into the subchain smart contracts, which will be investigated in our future work.

F. Zero-Knowledge Proof

The information on the subchains is passed to the public chain in the form of ZK-SNARK [31] attestation to verify that all the answers are submitted correctly by the subchain leaders for the workers. Note that in a block, every transaction can be tracked using Merkel root. When a task is completed, one or multiple blocks may be generated on the subchain. In order to improve the efficiency of the verification, the leader extracts the Merkel roots in all the block headers to form a new Merkel tree and computes the root of this new tree. This new root, denoted by $MT.Root(mt.root_{ID_t})$, represents that all the answers are submitted for the private task with identity ID_t. The leader then counts the total number of answers $|A|$, generate a pair of keys (epk, esk), a public ZK-SNARK parameter PP, finds the latest time t_c of all the answers, and then generates an attestation π_a. This attestation together with esk are used as ZK-proof's witness uploaded to public validators. The smart contracts on the public chain directly call *libsnark* to verify the validity of attestation [35]. Since the validators on the subchains can see all the transaction information of this task, these validators can be employed to verify the correctness of the attestation by exposing the PP on the subchain, preventing the leader from doing evil when submitting the commitment.

VI. Performance Evaluation

In this section, we conduct experiment to evaluate the performance of PBFT consensus on our proposed crowdsourcing platform by comparing with Casper consensuses on Ethereum (based on POW and POS [36]), and then analyze the experiment results in detail.

A. Experiment Environment

For the comparison purpose, we build two separate private blockchains, where one is on Ethereum platform utilizing Casper consensus and the other is on Hyperledger fabric platform adopting PBFT consensus. The experiment is performed on the desktops installed with Intel i7-7700k, 32GB RAM, 512 SSD hard drive, and the Ubuntu 16.04 operating system in VMWare, respectively testing the transaction verification speed of Ethereum and Hyperledger Fabric private chains. On Ethereum platform, the Eth Geth 1.8.22-stable application is installed to establish a connection with Ethereum main network, and the solidity compiler is deployed locally to compile the smart contract and create a private chain. On Hyperledger fabric platform, Hyperledger Fabric V1.4 is installed, and Hyperledger composer is used to interact with the built-in private blockchain. On both two private chains, we use the Go language to deploy smart contracts and a chaincode to accomplish multiple money transactions in a row between different accounts. For each blockchain, we create two accounts to send and receive transactions, respectively. The performance of Ethereum and Hyperledger Fabrics is measured in terms of execution time and throughput. (i) Execution time is the total time interval, during which all transactions are completed on the platform. (ii) Throughput

Authorized licensed use limited to: UNIVERSIDADE ESTADUAL DE MARINGA. Downloaded on July 14,2025 at 17:04:21 UTC from IEEE Xplore. Restrictions apply.

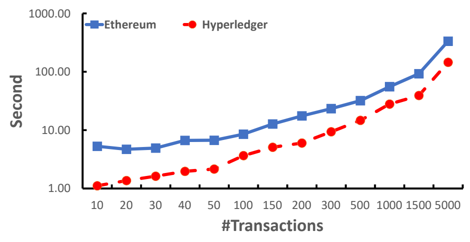

Fig. 2. Semi-log Plot on Execution Time with Maximum 5000 Transactions

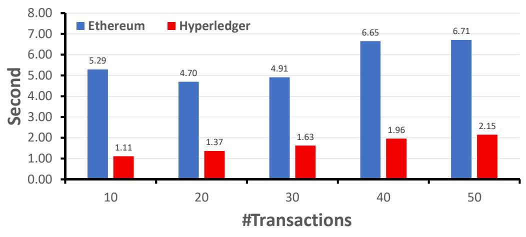

Fig. 3. Execution Time with Maximum 50 Transactions

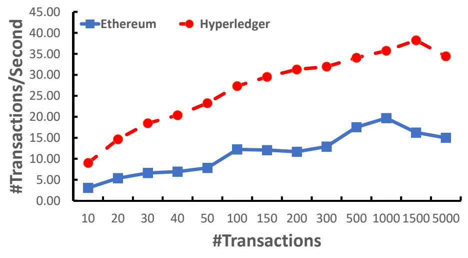

Fig. 4. Semi-log Plot on Throughput with Maximum 5000 Transactions

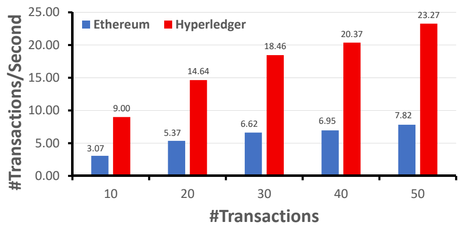

Fig. 5. Throughput with Maximum 50 Transactions

refers to the number of successful transactions per second, which can represent the processing capabilities of a system.

B. Result and Analysis

With the same transaction quantity, 20 rounds of experiments on both platforms are performed, and the averaged results are presented in Figs. 2-5.

Fig. 2 shows the semi-log plot of the execution time corresponding to different numbers of money transfer transactions. From the results, one can clearly see that as the number of transactions increases, the execution time of both Ethereum and Hyperledger platforms shows an increasing trend. With the number of transactions, the execution time of the Hyperledger is much smaller than that of Ethereum. In addition, as the number of transactions increases, the difference between the execution time of the two platforms increases.

Since the subchain on our proposed hybrid platform is temporarily constructed based on task service, usually the transaction volume is not very large on the subchains. Thus, the performance with small transaction volume should be paid more attention. In Fig. 3, we extract the results with the number of transactions less than 50. We find that the execution time of the Hyperledger private chain increases steadily with the increase of the number of transactions, while the private chain based on Ethereum has slight fluctuations, which stems from the randomness of the block producing rate in PoW consensus mechanism. We also find that with small transaction volume as shown in Fig. 3, Hyperledger's execution time is 3-5 seconds faster than Ethereum.

The semi-log plot of the throughput is shown in Fig. 4, from which it can be seen that Hyperledger outperforms Ethereum. The throughput of both Hyperledger Fabric and Ethereum platforms is increased first when the transaction volume grows up and then is reduced when the transaction volume is larger than a certain value. In our experiments, Hyperledger achieves the maximum throughput when the transaction volume is around 1,500, and Ethereum reaches its maximum when the transaction volume is 1,000. Such a certain value to obtain the maximum throughput indicates the platform capacity, and performance bottleneck will appear when the transaction volume becomes larger than this certain value.

Moreover, the throughput with the number of transactions smaller than 50 are extracted. From Fig. 5, we can conclude that (i) the throughput of both Hyperledger and Ethereum increases steadily with the increase of the transaction volume; and (ii) the throughput of Hyperledger is about three times of the throughput in Ethereum.

Remarks: Our experimental environment is built on local VMware, which to a certain extent limits the performance of Hyperledger and Ethereum private chains. The hardware configuration also greatly affects the efficiency of the transaction verification process. Nevertheless, these experiment settings do not affect the comparison between the performance of the two blockchains.

VII. Conclusion

In this paper, we propose a hybrid blockchain platform for crowdsourcing. By respectively employing DPOS and PBFT consensuses on the public chain and the subchains, our platform can achieve higher transaction throughput and less execution time compared with traditional PoW/PoS-based blockchain. In addition, our platform can realize permission control and privacy protection by deploying smart contracts and zero knowledge proof. Finally, we conduct intensive experiments to validate the efficiency of our platform through showing the superiority of Hyperledger over Ethereum.

As our preliminary work, the experiment in this paper mainly focuses on the efficiency of PBFT consensus used on our platform. In our future work, we will plan to realize

Authorized licensed use limited to: UNIVERSIDADE ESTADUAL DE MARINGA. Downloaded on July 14,2025 at 17:04:21 UTC from IEEE Xplore. Restrictions apply.

the fundamental functions of our crowdsourcing platform and further test the performance of the public chain as well as the interaction between the public chain and the subchains.

ACKNOWLEDGMENT

This research work is partly supported by the U.S. National Science Foundation under Grant No.1829674 and No.1741277.

REFERENCES

[1] W. Feng, Z. Yan, H. Zhang, K. Zeng, Y. Xiao, and Y. T. Hou, "A survey on security, privacy, and trust in mobile crowdsourcing," *IEEE Internet of Things Journal*, vol. 5, no. 4, pp. 2971–2992, 2018.

[2] wikipedia. [Online]. Available: https://www.wikipedia.org/

[3] Uber: Move the way you want. [Online]. Available: https://www.uber.com/

[4] Amazon mechanical turk. [Online]. Available: https://www.mturk.com/

[5] Problems at wikipedia. [Online]. Available: https://downdetector.com/status/wikipedia

[6] 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.

[7] (2017) Uber hid 2016 breach, paying hackers to delete stolen data. [Online]. Available: https://www.nytimes.com/2017/11/21/technology/uber-hack.html

[8] Y. Wang, Z. Cai, Z. Chi, X. Tong, and L. Li, "A differentially k-anonymity-based location privacy-preserving for mobile crowdsourcing systems," in *2017 International Conference on Identification, Information and Knowledge in the Internet of Things, IIKI 2017, Shandong, China, October 19-21, 2017*, 2017, pp. 28–34.

[9] T. Zhu, G. Li, W. Zhou, and P. S. Yu, "Differentially private data publishing and analysis: A survey," *IEEE Trans. Knowl. Data Eng.*, vol. 29, no. 8, pp. 1619–1638, 2017.

[10] Y. Zhang and M. van der Schaar, "Reputation-based incentive protocols in crowdsourcing applications," in *Proceedings of the IEEE INFOCOM 2012, Orlando, FL, USA, March 25-30, 2012*, 2012, pp. 2140–2148.

[11] C. Tanas, S. Delgado-Segura, and J. Herrera-Joancomartí, "An integrated reward and reputation mechanism for MCS preserving users' privacy," in *Data Privacy Management, and Security Assurance - 10th International Workshop, DPM 2015, and 4th International Workshop, QASA 2015, Vienna, Austria, September 21-22, 2015. Revised Selected Papers*, 2015, pp. 83–99.

[12] Y. Wang, Z. Cai, X. Tong, Y. Gao, and G. Yin, "Truthful incentive mechanism with location privacy-preserving for mobile crowdsourcing systems," *Computer Networks*, vol. 135, pp. 32–43, 2018.

[13] Z. Chi, Y. Wang, Y. Huang, and X. Tong, "The novel location privacy-preserving CKD for mobile crowdsourcing systems," *IEEE Access*, vol. 6, pp. 5678–5687, 2018.

[14] B. Jia, T. Zhou, W. Li, Z. Liu, and J. Zhang, "A blockchain-based location privacy protection incentive mechanism in crowd sensing networks," *Sensors*, vol. 18, no. 11, p. 3894, 2018.

[15] Y. Lu, Q. Tang, and G. Wang, "Zebralancer: Private and anonymous crowdsourcing system atop open blockchain," in *38th IEEE International Conference on Distributed Computing Systems, ICDCS 2018, Vienna, Austria, July 2-6, 2018*, 2018, pp. 853–865.

[16] S. Matsumoto and R. M. Reischuk, "IKP: turning a PKI around with decentralized automated incentives," in *2017 IEEE Symposium on Security and Privacy, SP 2017, San Jose, CA, USA, May 22-26, 2017*, 2017, pp. 410–426.

[17] J. Bonneau, A. Miller, J. Clark, A. Narayanan, J. A. Kroll, and E. W. Felten, "Sok: Research perspectives and challenges for bitcoin and cryptocurrencies," in *Security and Privacy (SP), 2015 IEEE Symposium on*. IEEE, 2015, pp. 104–121.

[18] M. Li, J. Weng, A. Yang, and W. Lu, "Crowdbc: A blockchain-based decentralized framework for crowdsourcing," *IACR Cryptology ePrint Archive*, vol. 2017, p. 444, 2017.

[19] D. Chatzopoulos, S. Gujar, B. Faltings, and P. Hui, "Privacy preserving and cost optimal mobile crowdsensing using smart contracts on blockchain," in *15th IEEE International Conference on Mobile Ad Hoc and Sensor Systems, MASS 2018, Chengdu, China, October 9-12, 2018*, 2018, pp. 442–450.

[20] J. Zou, B. Ye, L. Qu, Y. Wang, and M. A. O. L. Li, "A proof-of-trust consensus protocol for enhancing accountability in crowdsourcing services," *IEEE Transactions on Services Computing*, pp. 1–1, 2018, early access.

[21] S. Nakamoto. (2009) Bitcoin: A peer-to-peer electronic cash system. [Online]. Available: http://www.bitcoin.org/bitcoin.pdf

[22] Delegated proof of stake: Features and tradeoffs. [Online]. Available: https://multicoin.capital/wp-content/uploads/2018/03/DPoS-Features-and-Tradeoffs.pdf

[23] Eos: The most powerful infrastructure for decentralized applications. [Online]. Available: https://eos.io/

[24] Bitshares blockchain: Industrial-grade decentralized platform. [Online]. Available: https://bitshares.org/

[25] Make your idea possible. [Online]. Available: https://zilliqa.com/

[26] Hyperledger: Open source blockchain technologies. [Online]. Available: https://www.hyperledger.org/

[27] A. Rastogi, M. A. Hammer, and M. Hicks, "Wysteria: A programming language for generic, mixed-mode multiparty computations," in *2014 IEEE Symposium on Security and Privacy, SP 2014, Berkeley, CA, USA, May 18-21, 2014*, 2014, pp. 655–670.

[28] C. Liu, X. S. Wang, K. Nayak, Y. Huang, and E. Shi, "Oblivm: A programming framework for secure computation," in *2015 IEEE Symposium on Security and Privacy, SP 2015, San Jose, CA, USA, May 17-21, 2015*, 2015, pp. 359–376.

[29] M. Fredrikson and B. Livshits, "Zø: An optimizing distributing zero-knowledge compiler," in *Proceedings of the 23rd USENIX Security Symposium, San Diego, CA, USA, August 20-22, 2014.*, 2014, pp. 909–924.

[30] B. Parno, J. Howell, C. Gentry, and M. Raykova, "Pinocchio: nearly practical verifiable computation," *Commun. ACM*, vol. 59, no. 2, pp. 103–112, 2016.

[31] A. E. Kosba, A. Miller, E. Shi, Z. Wen, and C. Papamanthou, "Hawk: The blockchain model of cryptography and privacy-preserving smart contracts," in *IEEE Symposium on Security and Privacy, SP 2016, San Jose, CA, USA, May 22-26, 2016*, 2016, pp. 839–858.

[32] Zcash: A privacy-protecting, digital currency built on strong sciences. [Online]. Available: https://z.cash/

[33] M. Castro and B. Liskov, "Practical byzantine fault tolerance," in *Proceedings of the Third USENIX Symposium on Operating Systems Design and Implementation (OSDI), New Orleans, Louisiana, USA, February 22-25, 1999*, 1999, pp. 173–186.

[34] H. Sukhwani, J. M. Martínez, X. Chang, K. S. Trivedi, and A. Rindos, "Performance modeling of PBFT consensus process for permissioned blockchain network (hyperledger fabric)," in *36th IEEE Symposium on Reliable Distributed Systems, SRDS 2017, Hong Kong, Hong Kong, September 26-29, 2017*, 2017, pp. 253–255.

[35] E. Ben-Sasson, A. Chiesa, D. Genkin, E. Tromer, and M. Virza, "Snarks for C: verifying program executions succinctly and in zero knowledge," in *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part II*, 2013, pp. 90–108.

[36] V. Buterin and V. Griffith, "Casper the friendly finality gadget," *CoRR*, vol. abs/1710.09437, 2017.

Authorized licensed use limited to: UNIVERSIDADE ESTADUAL DE MARINGA. Downloaded on July 14,2025 at 17:04:21 UTC from IEEE Xplore. Restrictions apply.