zkCrowd: A Hybrid Blockchain-Based Crowdsourcing Platform

Saide Zhu, Zhipeng Cai, *Senior Member, IEEE*, Huafu Hu, Yingshu Li, *Senior Member, IEEE*, and Wei Li, *Member, IEEE*

***Abstract*—Blockchain, a promising decentralized paradigm, can be exploited not only to overcome the shortcomings of the traditional crowdsourcing systems, but also to bring technical innovations, such as decentralization and accountability. Nevertheless, some critical inherent limitations of blockchain have been rarely addressed in the literature when it is incorporated into crowdsourcing, which may yield the performance bottleneck in the crowdsourcing systems. To further leverage the superiority of combining blockchain and crowdsourcing, in this article, we propose an innovative hybrid blockchain crowdsourcing platform, named zkCrowd. Our zkCrowd integrates with a hybrid blockchain structure, smart contract, dual ledgers, and dual consensus protocols to secure communications, verify transactions, and preserve privacy. Both the theoretical analysis and experiments are performed to evaluate the advantages of zkCrowd over the state of the art.**

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

I. Introduction

With the rapid development and wide application of crowdsourcing, the shortcomings of the traditional crowdsourcing systems are gradually exposed.

1) The traditional centralized crowdsourcing systems are vulnerable to a single point of failure. For example, users could not access Wikipedia data from the server due to eight service outages in 2018 [1].
2) In such a centralized system where a server controls all the transactions, the issue of controller's silent misbehavior is likely to occur without effective detection.
3) When a conflict of opinions exists between the task requesters and the workers, the issues of free riding (workers receive rewards without making real efforts) and false reporting (requesters try to repudiate the payment) could happen in the system [2].
4) During the procedure of task assignment, the sensitive information (e.g., location and preference) of crowdsourcing participants may be revealed to the public. For example, in November 2017, 57 million Uber drivers' information was hacked [3].

To solve these issues, a number of countermeasures have been proposed. To name some—cryptographic techniques were used to preserve user's private information [4]–[6]; and reputation systems were adopted to tackle the issues of free-riding and false-reporting [7], [8]. However, most of the existing works fail to simultaneously overcome the aforementioned issues for crowdsourcing.

The recent big progress of blockchain technology enables the seamless integration of smart contracts and novel cryptographic tools into blockchain networks, which provides an innovative way to improve the performance of crowdsourcing. In [9] and [10], a variety of solutions have been presented for location privacy issues. In order to offer a security guarantee for the users' participation, identity authentication systems were designed in [11] and [12]. In [10], the combination of public and private chains is used to protect user's private information, in which, however, the authority of the agent entity needs to be further discussed. Nevertheless, all of these works ignore the scalability and verification efficiency of the blockchain network architecture which are the performance bottleneck of the crowdsourcing systems.

Motivated by the above-mentioned observations, in this article, we propose to design a blockchain-enabled crowdsourcing platform to provide users with diverse privacy protection while improving the efficiency of crowdsourcing. To achieve our goal, we have to tackle the following challenging problems.

1) *Which consensus protocol should be adopted to enhance the transaction verification speed as well as to provide security guarantee?* For instance, under the traditional proof-of-work (POW) consensus protocol, the blockchain networks suffer from issues of low verification rate, scalability bottleneck, and huge energy consumption [13].
2) *How to balance the tradeoff between transparency and privacy in crowdsourcing?* Transparency is one of the most important advantages of blockchain to achieve accountability. But, on the other hand, transparency could result in privacy leakage.
3) *How to effectively verify the protected transaction information?* A crowdsourcing platform is expected to ensure that the correctness of private information is well-verified

Manuscript received May 31, 2019; revised August 11, 2019 and August 29, 2019; accepted August 30, 2019. Date of publication September 16, 2019; date of current version February 28, 2020. This work was supported by the National Science Foundation (NSF) under Grant 1829674, Grant 1741277, Grant 1912753, and Grant 1704287. Paper no. TII-19-2302. *(Corresponding author: Wei Li.)*

S. Zhu, Z. Cai, Y. Li, and W. Li are with the Department of Computer Science, Georgia State University, Atlanta, GA 30302 USA (e-mail: szhu5@student.gsu.edu; zcai@gsu.edu; yili@gsu.edu; wli28@gsu.edu).

H. Hu is with Omega Bio-tek Inc., Norcross, GA 30071 USA (e-mail: hhu4@student.gsu.edu).

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TII.2019.2941735

while the exact transaction information is unknown to the public.

4) *How to integrate the public and private transaction information?* As user's privacy preference varies in person, it is desired to offer flexible privacy protection. In this situation, some transactions may be public, while others are private. The integration of transaction information needs to be well arranged to meet the users' privacy requirements.

Our research endeavor for dealing with the aforementioned challenges is to design a hybrid blockchain-enabled crowdsourcing platform, called zkCrowd. The zkCrowd is composed of a public chain and multiple private subchains, in which public and private tasks are managed on the public chain and subchains, respectively. By exploiting such a hybrid blockchain architecture, the answers of public and private tasks in crowdsourcing can be separately verified, so as to provide privacy protection for the answers of private tasks. For the purpose of enhancing verification speed, delegated proof of stake (DPOS) and practical Byzantine fault tolerance (PBFT) consensuses are implemented on the public chain and subchains, respectively. Compared with existing hybrid blockchains, such as Smilo [14], AERGO [15], and Xinfin [16], our zkCrowd has the following major advantages: 1) in zkCrowd, the cross-chain communication only involves the transmission of answer confirmation from the subchains to the public chain without the issue of cross-chain communication atomicity. Therefore, there is no need to build a bridge module for transaction synchronization, improving verification efficiency for crowdsourcing tasks as well as reducing energy consumption; 2) in crowdsourcing systems, due to the dynamic of tasks and mobility of users, the number of tasks is fluctuating. Inspired by this characteristic, on zkCrowd, a subchain is dynamically created and released according to the task requirement. After a private task has been finished, the subchain is released, and the corresponding subchain validators can turn back to the public chain. That is, the subchain validators do not need to be always in the monitor state, reducing resource consumption; and 3) our zkCrowd has a unified election mechanism for the public chain and subchains. The voting mechanism of the subchain validators in PBFT is combined with the voting witnesses in DPOS. Therefore, the dynamic subchain establishment does not yield any extra overhead to the blockchain system when selecting validators. The effectiveness of our zkCrowd is evaluated through comprehensive performance analysis and experiment comparison. To sum up, the major contributions of this article are addressed in the following:

1) An innovative hybrid blockchain platform, zkCrowd, is elaborately designed for distributed crowdsourcing, in which transaction privacy and transparency can be effectively balanced.
2) By utilizing DPOS and PBFT consensus protocols, the transaction verification efficiency can be significantly increased, reducing transaction latency and energy consumption in the crowdsourcing system.
3) Diverse privacy protection is achieved by accomplishing zero-knowledge proof (ZK-proof) and flexible task-based permission control in the smart contracts on the proposed hybrid blockchain architecture.
4) Both the theoretical analysis and experiments can validate the advantages of our zkCrowd compared with the state of the art.

The remaining organization of this article is as follows. The existing related literature and preliminaries are summarized in Section II. In Section III, the preliminaries are introduced. Our hybrid blockchain platform, zkCrowd, and the implementation process are detailed in Section IV and Section V, respectively. The theoretical analysis is shown in Section VI, and the experiments are analyzed in Section V-G. Finally, Section VIII concludes this article.

II. Related Work

Recently, the integration of blockchain and crowdsourcing has been paid tremendous attention by researchers.

In [17], blockchain and smart contracts were employed to facilitate the communication between mobile crowdsensing providers and mobile users, but authors focused on the smart execution of incentive mechanism rather than task processing. In [4], a blockchain-based decentralized crowdsourcing framework was proposed with the deployment of smart contracts. A proof-of-trust (POT) consensus on hybrid blockchain was designed to satisfy the needs of the service industry [18], in which the structure of POT consensus is similar to that of hyperledger fabric for enhancing transaction verification rate. But, these works [4], [17] did not consider user privacy preservation. To protect user privacy, ZebraLancer [11], a private and anonymous crowdsourcing platform, was developed, in which however, a large data load may be caused and thus, the system efficiency may be reduced. The hybrid blockchain platform of [10] can secure transaction as well as preserve location privacy for crowdsourcing. However, the centralization problem of the agent nodes on the platform still need more discussion.

On the other hand, novel cryptographic techniques have been implemented to ensure the security and privacy of blockchain. For example, ZK-proof deployment on smart contracts [19], self-driven and correct authentication of user identity [12], and automatic access control on blockchain [20]. However, the above-mentioned techniques have not been widely applied to the blockchain-based crowdsourcing systems.

III. Preliminary

A. DPOS Consensus

In the BFT-based DPOS consensus [21], there are three types of nodes—stakeholders, witnesses (block producers or validators), and alternative nodes. Specifically, stakeholders have the rights to vote for elections of witnesses, witnesses verify the previous transactions and package the newly generated transactions into blocks, and alternate nodes who are the remaining nodes update the latest information of the ledgers in time. For example, in the EOS.io network, there are 21 witness nodes that take turns to share the accounting rights of the blockchain every 3 s. Different from the POW mining mechanism where every node

needs to compute a puzzle, the witnesses in DPOS are voted by the stakeholders, leading to reduced energy consumption, faster block generation rate, and shorter transaction verification time. The robustness of DPOS has been confirmed by enterprises, such as [22] and Bitshares [23].

B. PBFT Consensus

Another consensus that can tolerate Byzantine failures is the PBFT consensus, which was proposed by Castro and Liskov; it can tolerate Byzantine failures and improve the performance of the original Byzantine fault-tolerant algorithm.

There are five major stages in PBFT—request, preprepare, prepare, commit, and reply. The whole process ensures the correctness of the execution. A client first sends a request to the leader node. The leader election process is repeated in a round-robin manner by term in advance. When receiving the request, the leader broadcasts the preprepare message to inform the entire network and sends the client's request to other peer nodes. After accepting enough prepare information, the peer nodes multicast a prepare message to all other nodes. The replica node verifies and writes a commit message to the message log and accepts the command for execution. At last, the nodes execute the request and send a reply to the client. The client should wait for more than $f + 1$ replies to ensure the correctness of the execution, where f is the number of Byzantine failure nodes in the network. The PBFT consensus can achieve a high-transaction verification speed at a low transaction volume [24]. This is why PBFT is suitable as a private chain consensus. Additionally, in PBFT, the transaction finality can be achieved more easily without waiting for multiple confirmations. Moreover, the network agreement of PBFT can be accomplished without intensive computations, further reducing the energy consumption.

C. Merkle Tree

Merkle tree, which is a type of data structure (usually a binary tree and possibly a multifork tree), stores hash values of data contents. In a blockchain (e.g., Bitcoin), Merkle tree is used to quickly summarize and verify the integrity of block data by hashing transactions. Each hash node is generated from two adjacent data nodes continuously and recursively, leaving only one Merkle root stored in the block headers.

The main advantages of using a Merkle tree in a blockchain system is that it greatly improves the efficiency and scalability of the blockchain, so that the block header only needs to contain the root hash value without having to encapsulate all the underlying data, which makes the verification run efficiently. Also, in the blockchain, the Merkle tree supports the simplified payment verification protocol (SPV), which allows transaction data to be executed without running a full node. Thus, the payment confirmation problem can be solved under light client conditions.

D. Smart Contract

The smart contracts were first deployed on Ethereum using *Solidity* language. In the smart contracts, the codes will be automatically executed when a condition is triggered by the contract contents, reducing the cost of communication and supervision. Since the smart contracts are implemented on a blockchain, they possess some critical features including accountability and decentralization. Also, the blockchain can guarantee that the contents of the smart contracts cannot be changed. When it comes to crowdsourcing, the blockchain platforms are expected to process data and control access permission for various tasks. Therefore, the smart contracts can be employed as a programmable interface for developers to set up different procedures.

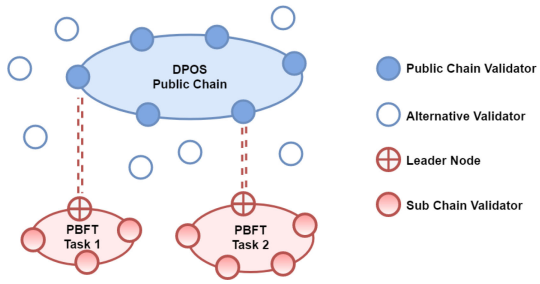


Fig. 1. Blockchain architecture of zkCrowd.

E. Zero-Knowledge Proof

Protecting user privacy is an important focus in the blockchain. By using ZK-proof, private information can be preserved during the verification process [25], [26]. In this article, zero-knowledge succinct noninteractive argument of knowledge (ZK-SNARK) is utilized to construct optimized ZK-proof to help verify transactions in the communications between the public chain and the subchains. Typically, the ZK-SNARK algorithm has three components: 1) a setup algorithm is run to produce a public parameter to establish the SNARK; 2) an attestation is generated by a prover and sent to the SNARK verifier on the blockchain; and 3) the correctness of the proof is verified via calling the verifier module.

IV. Framework of zkCrowd

Our zkCrowd platform is established on a hybrid blockchain network being composed of a public chain and multiple subchains. As shown in Fig. 1, on zkCrowd, the public chain and the subchains are used to publish and record the information of public and private crowdsourcing tasks, respectively.

The entities on zkCrowd include the following.

1) Requester. Requesters post tasks, collect answers from workers, and pay to workers and validators for their efforts. Before publishing tasks, they need to deposit an amount of money into the blockchain through the smart contracts, in which the deposit is used to reward the workers who make effort in completing tasks as well as the validators who help verify transactions. Every requester has the right to vote the validators on the public chain and the subchains. A requester may simultaneously play the role of a worker for a different task but is not allowed to work as a validator.

2) Workers. Workers carry out requesters' tasks and receive payments from the requesters via signing the smart contracts. If a worker participates in a private task, his answers should be submitted to subchain validators. Each worker has the right to vote for public-chain validators and subchain validators. Any worker cannot register as a validator, but can potentially become a requester.

3) Validators. There exist three kinds of validators: a) *public chain validators* verify transactions of the public chain, take turns to perform the accounting right, share a public chain ledger, and need to be elected in each time cycle; b) *alternative validators* are the nodes that are not elected as the public chain validators. They should also share the public chain ledger but do not have the right to generate blocks; and c) *subchain validators* are elected among the existing alternative validators to verify transactions on the subchains and record both the public chain and subchain ledgers. Any validator cannot work as a public chain validator and a subchain validator at the same time.

All the entities must register as legitimate participants through a trustworthy certificate authority (TCA) when joining zkCrowd. Only the entities that have obtained the legal certificates can carry out the activities on zkCrowd.

When the public chain has been set up, the requesters can post public and/or private tasks on zkCrowd, and the workers select their preferred tasks but need the permission from the corresponding requesters to process the tasks. The type of tasks is checked via the smart contracts. For the public tasks, the answers are submitted to the public chain for verification, and the transaction information is kept in the public chain ledger; while for the private tasks, a task-based subchain is built, the answers are uploaded to the corresponding subchain for verification, and the transaction information is written into the subchain ledger. On each subchain, after the subchain ledger gets updated, the leader constructs a new Merkle tree of which the leaves are the block Merkle roots extracted from all block headers and posts a transaction confirmation that is computed using ZK-SNARK on the public chain.

This confirmation can be used to trace the leader's behavior, because on each subchain, the subchain ledger is shared by all subchain validators. On the other hand, the information of private tasks is allowed to be accessed via the permission stated in the smart contracts and thus, can be preserved on the subchains without being revealed to the public. Finally, the payments are assigned to the workers and the validators according to the distribution method (that can be determined by the requesters based on their task requests), and the remaining deposits will be returned to the requesters.

V. Implementation of zkCrowd

This section details the major phases in the implementation of zkCrowd. For a better presentation, the work flow of the platform implementation is presented in Fig. 2, and the notations are summarized in Table I.

A. Identity Authentication

In crowdsourcing, the authentication of user identities is an important issue [12], but the blockchain itself cannot verify this.

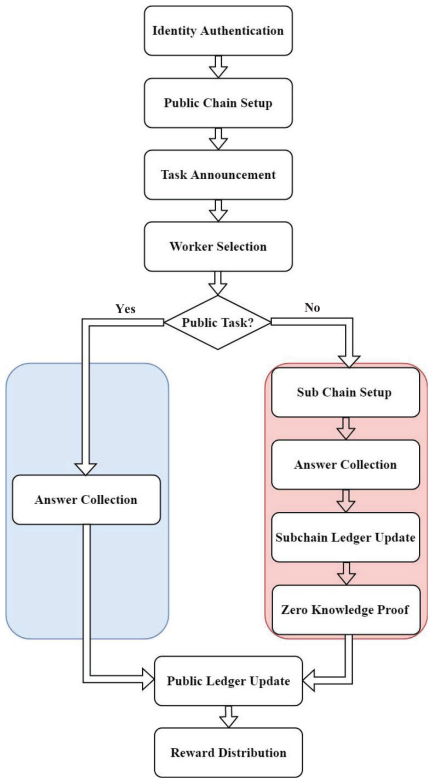


Fig. 2. Work flow of zkCrowd implementation process.

TABLE I
Notation and Definition

Notation	Definition
pk_*	Public key of entity *
sk_*	Private key of entity *
ut	User type in {r, w, v}
ID_*	Real identity information of entity *
ID_*^c	Certified identity of entity *
ID_t	Task ID
T	Task type in {public, private}
t_1	Deadline for workers to reply to tasks
t_2	Deadline for workers to submit answers
t_r	Worker's response time
t_a	Answer submission time
t_c	Latest answer submission time
R	Total rewards of a task implementation
Num	Maximum number of answers for a task
p	Task implementation preference
cap_w	Worker's capacity
$permit$	Permission to process task
Ans	Task answer
$A_{ID_w^c}$	Answer content submitted by worker
π_a	Attestation of ZK-proof
PP	Public parameter of ZK-proof
$mt.root$	Merkle tree root

To solve the issue, a TCA is brought to take charge of identity authentication and certificate issuance for the entities.

Initially, an entity prepares key (pk_{ut}, sk_{ut}) using Rivest–Shamir–Adleman (RSA) scheme, in which pk_{ut} and sk_{ut} are, respectively, the entity's public and private keys, and $ut \in \{r, w, v\}$ is the entity type (r, w, and v stand for requester, worker, and validator, respectively). The entity registers by sending $E(pk_c, pk_{ut}, ID_{ut}, Sig(sk_{ut}, ID_{ut}))$ to TCA, where

pk_c is TCA's public key, ID_{ut} is the entity's true identity, and $Sig(\cdot)$ is the secure digital signature algorithm. Then, TCA checks if ID_{ut} and pk_{ut} are in his historic information or not because everyone on zkCrowd cannot use different public keys to prevent Sybil attack. Once ID_{ut} and pk_{ut} are verified to be valid, a unique certified identity ID^c_{ut} and a certification $Cert_{ut}\{ID^c_{ut}, ut, pk_{ut}\}$ are issued and sent back to the entity in the form of $E(pk_{ut}, Cert_{ut}, Sig(sk_c, Cert_{ut}))$ with sk_c being TCA's private key. After successful registration, the entity can participate in the activities on zkCrowd.

B. Public Chain Establishment

On zkCrowd, by running the DPOS consensus, the public chain is established by all validators who are issued certificates by TCA [21]. After that, the public chain validators are elected by the registered requesters and workers periodically using the election mechanism of DPOS. In particular, a validator re-election process should be launched if the malicious behaviors of any public chain validators are found. In every election process, the 21 validators who receive the most votes serve as the public chain validators to perform the accounting right in turns as well as the transaction verification in the current period. The rest of the validators are the alternative validators that share the public chain ledger and are available for the election of subchain validators.

C. Task Announcement and Worker Selection

Each task is published on zkCrowd with the information $Task\{T, ID_t, t_1, t_2, R, Num, p, pk_r, Cert_r, Sig(sk_c, Cert_r)\}$, where $T \in \{\text{public, private}\}$ represents the task type, ID_t is the task ID, t_1 and t_2 are, respectively, the deadlines for the workers to reply to the task information and submit the task answers, R is the total amount of payment for the task implementation, Num is the maximum number of expected answers for the task, p is the task requirements (such as location, duration, and data quality) needed by the task requester to choose workers, pk_r is the task requester's public key, $Cert_r$ is the task requester's certification. Meanwhile, each task requester must make a deposit to the smart contracts. On zkCrowd, the smart contracts are also executed to check whether or not the task is announced by a registered requester and the deposit is enough to pay to the workers and validators. The certificates of all requesters and workers can be checked using TCA's digital signatures (i.e., $Sig(sk_c, Cert_r)$ and $Sig(sk_c, Cert_w)$). Once a task successfully passes such a checking process, the task information is delivered to the online workers of zkCrowd.

A worker should return a reply before the response deadline if he prefers to process a task. The reply message is as follows: $Res\{ID_t, ID^c_w, t_r, E(cap_w), pk_w, Cert_w, Sig(sk_c, Cert_w)\}$, where ID^c_w is the worker's registered ID, t_r is the response timestamp, cap_w indicates the worker's capacity (such as location, available time, quality of offered data, and others), and pk_w and $Cert_w$ are, respectively, the worker's public key and certificate. Since cap_w may contain the worker's private information, it should not be publicly available when submitted to the blockchain. To prevent privacy leakage, the worker encrypts cap_w with the requester's public key pk_r, i.e., $E(cap_w) = E(pk_r, cap_w, Sig(sk_w, cap_w))$. When the reply is delivered to the requester, he examines the response time to make sure $t_r \leq t_1$, uses sk_r to recover cap_w, and verifies the worker's certificate as well. By considering the task requirements p and the worker's capacity cap_w, the requester can select his preferred workers, issue a permission $\text{permit}(ID_t, ID^c_w, Cert_w)$ to each selected worker, and send out the notification message $E(pk_w, \text{permit}, Sig(sk_r, \text{permit}))$. Each selected worker should upload the answer to zkCrowd before t_2 in the form of $Ans(T, ID_t, A_{ID^c_w}, ID^c_w, t_a, \text{permit})$ with $A_{ID^c_w}$ recording the answer content and t_a being the submission timestamp.

D. Subchain Establishment

Whenever a private task needs to be processed on zkCrowd, a subchain is built to collect answers and verify transactions. In other words, the establishment of subchains is dynamically determined by the demand of private tasks.

On each subchain, the PBFT consensus [24], [27] is adopted, and the validators are elected among the current highest voted alternative validators. To defend the Byzantine failure attack under the PBFT consensus, the total number of nodes must be larger than three times the number of the Byzantine failure nodes in the blockchain network. Formally, assume that the number of subchain validators is n_v and the number of Byzantine nodes is n_b. To achieve the BFT on zkCrowd, we have $n_v = 3n_b + 1$ where $n_b \geq 1$; that is, there must exist at least four subchain validators on each subchain. The determination of n_v should consider the following facts: 1) if the maximum number of answers Num for a private task is small, a few validators (e.g., four validators) are enough to deal with the transaction volume of the subchain; while if Num is large, more validators are necessary to handle the transaction verification; 2) a larger value of n_v indicates a stronger ability to resist Byzantine failure attack; and 3) as more validators enter a subchain, the communication overhead may increase dramatically, and the verification fee is also increased, leading to a higher cost to run the subchain. On the public chain, 21 validators are sufficient to verify all transactions for the whole system. Since the value of Num may not be too large, a small number of validators (e.g., four or seven) is enough. Besides, in the PBFT consensus, a leader node is elected periodically in the round-robin manner to update the subchain ledger and communicate with the public chain.

After the private task is completed, the subchain is released and its subchain validators become the alternative validators on zkCrowd. In other words, any subchain validator can reparticipate in the election of the public chain validators only after his subchain is released.

E. Answer Collection on Public Chain

During the procedure of answer collection, the smart contracts should ensure that in $Ans(T, ID_t, A_{ID^c_w}, ID^c_w, t_a, permit)$: 1) the answer is submitted by a permitted worker by checking permit; 2) the answer $A_{ID^c_w}$ has not yet been recorded in the

answer pool $Pool_a$; and 3) the answer is submitted on time, i.e., $t_a \leq t_2$. If the above-mentioned three conditions can be simultaneously satisfied, the answer is valid and is recorded into the answer pool. 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.

F. Answer Collection on Subchain

For each private task, the procedure of answer collection is similar to that on the public chain. The subchain validators only copy the transaction records from the public-chain ledger and record private task information in the subchain ledgers. The leader of each subchain can utilize ZK-proof to generate a commitment on all subchain blocks of a task and sends it to the public chain, and on the public chain, the smart contracts run ZK-SNARK verifier to check the task completion and records it into the public-chain ledger which will be described in Section V-G. Therefore, all private tasks cannot be revealed but can be verified on the public chain. In zkCrowd, the type of task determines whether the answers will be handled by the public chain or subchain. Particularly, for one task, different workers may require different privacy protection levels of their answers. Thus, for a better access control feature, an access list model can be added into the subchain smart contracts, which will be studied in our future work.

G. Zero-Knowledge Proof

The transaction information of the subchains is sent to the public chain using ZK-SNARK [19] attestation to verify that all the answers are submitted correctly by the subchain leaders for the workers. Within each block, every transaction can be tracked using Merkle root. When a task is completed, one or multiple blocks may be generated on the subchain. In order to enhance the verification efficiency, the leader extracts the Merkle roots from all the block headers to construct a new Merkle tree and computes the new Merkle root. Such a new root, denoted by $MT.Root(mt.root_{ID_t})$, means that all the answers are submitted for the private task with identity ID_t. Notice that every $2^n - 1$ blocks can form a full binary Merkle tree, in which n could be any positive integer. To make the frequency of answer submission on the subchains flexible, each leader can perform the above operation whenever receiving $2^n - 1$ blocks before the answer submission deadline t_2, where the value of n is determined by the platform. Furthermore, the overhead of computing a new Merkle root can be adjusted by setting the value of n.

Then, the leader counts the total number of answers $|A|$, finds the latest time t_c of all the answers, computes key (epk, esk) and a public ZK-SNARK parameter PP, and generates an attestation π_a. All these keys and public parameters can be computed off the blockchain (such as using an intel SGX) to reduce the on-chain overhead. The attestation together with esk are used as ZK-proof's witness uploaded to the public chain. The smart contracts on the public chain directly call *libsnark* to verify the validity of attestation [28]. Since the validators on the same subchain share the subchain ledger, these validators are able to verify the correctness of the attestation through PP, avoiding the leader's misbehavior.

H. Reward Distribution

Once a task is finished, the ledgers on the public chain and subchains are updated correspondingly, the reward of each answer and the verification fee of each validator/subchain leader are distributed via the execution of smart contracts. In this article, besides the reward of answers, the requesters are supposed to pay the verification fee. In reality, different task requesters may adopt the same or different reward distribution policies, and various ways could be utilized by the crowdsourcing platforms to charge validation fee. For examples, each worker's received reward may depend on the number of submitted answers or the quality of submitted data; each validator's received verification fee may be determined by the number of verified transactions; and each subchain leader also obtains an amount of the verification fee for his effort to help complete transaction verification and communicate with the public chain. After paying the rewards and validation fee, the rest of the deposits will be returned back to the requesters' accounts. But, if a requester is identified as malicious, his deposit will be distributed to the corresponding selected workers. The design of reward distribution mechanism is outside the scope of this paper and will be further investigated in our future work.

VI. Performance Analysis

In this section, we first analyze the advantages of zkCrowd from the aspects of the platform architecture, consensus mechanism, and cryptographic methodology. Then, we discuss the performance of zkCrowd to resist possible attacks and compare it with the state of the art.

A. Platform Architecture

Our zkCrowd has a hierarchical architecture of hybrid blockchains, in which diverse tasks (including public and private tasks) can be effectively processed with different levels of privacy protection and permission control. At the same time, zkCrowd well-integrates the characteristics of consensuses on the public chain and subchains. The subchain validators are directly selected from alternative validators with the highest number of votes, avoiding an extra voting process. That is, such a unified election mechanism does not add additional overhead to the blockchain system when selecting validators. Therefore, the subchain setup process can only use the idle computation resources of the public chain. Moreover, when a private task is completed, the subchain validators will be released back to the public chain. The flexibility of subchain establishment in zkCrowd can further reduce the network overhead and enhance the resource utilization.

B. Consensus Mechanism

1) DPOS Consensus: Compared with the POW blockchain (whose verification speed is 3.3–7 s per transaction [29]) and the POS blockchain (whose verification speed is $\frac{1}{12}$ s per

transaction [30]), the verification speed of DPOS can reach tens of thousands of transactions per second, which can be applied to enterprise-class crowdsourcing systems [21]. Thus, by adopting the DPOS consensus, the verification speed on the public chain can satisfy the needs of crowdsourcing.

The centralization problem of blockchain networks should be addressed. For examples, in the POW-based Bitcoin network, the total mining power of the top six mining pools has exceeded 80% of the entire mining power [31]; in POS systems, a voter with more stakes has a higher influence and rich people will become richer [32]. Although the DPOS consensus is controlled by 21 public validators, their accounting rights rotate every 3 s, and for each time cycle, the public validators must be revoted. This could avoid the DPOS-based systems becoming centralized.

2) PBFT Consensus: In the PBFT consensus on the subchains, the transactions are verified in a distributed manner. Compared with the existing hybrid blockchain [10] where the private chain is hosted by a single agent on public chain, PBFT consensus in zkCrowd elects the leaders in a round-robin manner, which can prevent the silent misbehavior and single point of failure problems caused by the private chain leaders. Also, our experiments in Section VII show that the PBFT consensus achieves a good performance in execution time, latency, and throughput, especially for small transaction volumes compared with Casper consensus in Ethereum. The results indicate that the PBFT consensus is very suitable for the subchain of zkCrowd.

C. Cryptographic Methodology

1) Certificate Authority: Before users join zkCrowd, TCA is employed to check the user's true personal identity and his public key and issue a digital certification. This can guarantee that the identity of each participant in the TCA's record must be unique. Thus, the adoption of TCA can prevent Sybil attack while ensuring anonymity.

2) Zero-Knowledge Proof: For the answers of private tasks on the subchain, an ZK-proof is utilized to ensure the answers are unrevealed on the public chain. Meanwhile, the public chain validators can verify the submission process of the task answers by verifying the commitment of ZK-SNARK. On any subchain, since all subchain validators share the same ledger, they can verify whether the commitment is created correctly by their leaders using the PP of ZK-SNARK.

The on-blockchain computation of ZK-SNARK is actually very light because the verification process is executed by only calling existing library $libsnark$, while the heavy computation of ZK-prover is done off blockchain. Therefore, ZK-SNARK can protect the private answers without introducing much computation overhead.

D. Attack Resistance

1) 51% Attack: In the traditional POW or POS-based blockchain networks, if a malicious attacker controls over half of the network computing power/stake, he can use it as a feature of competitive conditions to cancel the payment transactions that have already occurred. In zkCrowd, it adopts the DPOS consensus and the accounting right is divided into 21 public chain validators, thus, preventing the 51% attack.

TABLE II
COMPARISON OF ATTACK RESISTANCE

Attack	zkCrowd	MTurk [34]	Dynamo [35]	CrowdBC [4]	ZebraL [11]
51% Attack	✓	✓	✓	×	×
Sybil Attack	✓	×	✓	✓	✓
Privacy Leakage	✓	×	×	×	✓
Free-riding	✓	✓	✓	✓	✓
False-reporting	✓	×	×	✓	✓
Byzantine Failure	✓	×	×	×	×

✓: realized attack resistance; and ×: vulnerability to attack.

2) Sybil Attack: An attacker may create multiple identities in order to manipulate the reputation system and gain more benefit in the network. To solve this issue, the TCA in zkCrowd performs identity authentication and detects the misbehavior. That is, if an attacker owns multiple identities, it cannot get a valid certificate to join zkCrowd.

3) Privacy Leakage: Workers' private information may leak to the public when submitting their answers. In zkCrowd, the subchains with ZK-proof ensure the secure submission of answers without revealing private information on the public chain.

4) Free Riding and False Reporting: Free riding means that workers receive rewards without making real efforts and false reporting refers to that dishonest requesters try to repudiate the payments. These types of attacks cannot happen in zkCrowd due to the reason that a detailed smart contract with reward policy can ensure the correct process for a task.

5) Byzantine Failure: Some members in the system may make mistakes or send wrong information, which may lead to information corruption and different decisions made by the consensus, thereby destroying system consistency. In zkCrowd, both the public chain (DPOS) and subchain (PBFT) utilize the BFT consensus mechanisms, thus, guaranteeing different levels of BFT.

A comparison between zkCrowd and some existing crowdsourcing platforms is summarized in Table II.

VII. PERFORMANCE EVALUATION

In this section, extensive experiments are set up to assess the efficiency of DPOS and PBFT by comparing with Casper consensuses on Ethereum (based on POW and POS [35]).

A. Experiment Environment

Three private blockchains are built with the following experiment configurations: 1) DPOS-based EOS.io. The first private blockchain is on the EOS.io platform running DPOS consensus. The desktop with Intel i7-7700k, 32 GB RAM, 512 SSD hard drive is used. The private blockchain is built on EOS

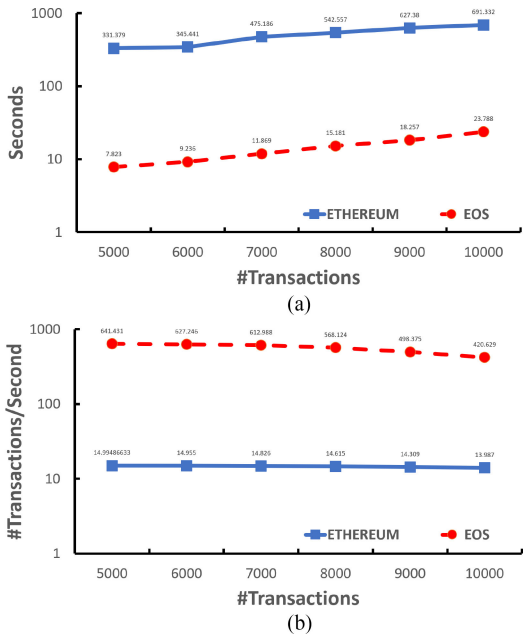


Fig. 3. Performance comparison between DPOS and Casper. (a) Semilog plot on execution time with 5000-10 000 transactions. (b) Semilog plot on throughput with 5000-10 000 transactions.

DAWN-v3.0.0, tested with txn_test_gen plugin on binaryen. One producer and one generator are connected to each other; 2) PBFT-based Hyperledger Fabric. The second one is on the Hyperledger Fabric platform using PBFT consensus. The desktop with the Ubuntu 16.04 operating system in VMWare is used, in which Hyperledger Fabric V1.4 is installed and Hyperledger composer is used to interact with the built-in private blockchain; and 3) Casper-based Ethereum. The third one is on the Ethereum platform using Casper consensus. The desktop with Intel i7-7700k, 32 GB RAM, 512 SSD hard drive is used. The Eth Geth 1.8.22-stable application is installed to build connections to Ethereum main network, and the solidity compiler is deployed locally to compile the smart contracts and create a private chain.

For each blockchain, we create two accounts to send and receive transactions, respectively. On the three chains, the Go language is exploited to deploy the smart contracts and a chaincode to accomplish multiple answer submission transactions in a row between different accounts. The performance metrics include: 1) execution time, which is the total time interval when all transactions are completed on the platform; 2) throughput, which refers to the number of successful transactions per second, indicating the verification capability of a blockchain network; and 3) latency, which is an average difference between a transaction completion time and its deployment time.

B. Result and Analysis

For the same transaction volume, each experiment is run 20 times, and the averaged results are presented in Figs. 3 and 4.

First, the comparison results in Fig. 3 reveal the performance of DPOS and Casper, where the transaction volume increases from 5000 to 10 000. Since the transaction volume on the public chain could be relatively large, we mainly focus on performance under the scenario with high transaction volume.

From Fig. 3(a), we can easily conclude that the execution time of both EOS.io and Ethereum is getting longer as the transaction volume increases. Particularly, when the transaction volume reaches 10 000, the execution time of Ethereum is about 30 times the execution time of EOS.io. The throughput is reported in Fig. 3(b). When the transaction volume reaches 10 000, EOS.io can still verify more than 400 transactions per second, while Ethereum only has over 10 transactions per second.

Therefore, the above results validate the advantages of the DPOS consensus on the public chain.

Then, the performance comparison between PBFT and Casper is presented in Fig. 4. On the subchains, the transaction volume is usually small, so in the experiments, the number of transactions is changed from 10 to 5000.

Fig. 4(a) shows the execution time of answer submission transactions when the transaction volume varies. From the results, we can observe that the execution time of both Hyperledger and Ethereum platforms is getting longer as the transaction volume increases. With the same transaction volume, the execution time of the Hyperledger is much shorter than that of Ethereum.

Since the subchain on zkCrowd is dynamically established based on task demand, usually the transaction volume is not very large on the subchains. Thus, the performance with small transaction volume should be paid more attention. As shown in Fig. 4(d), the results with the transaction volume less than 50 are reported. With the increase of the transaction volume, the execution time of the private chain on Hyperledger grows slowly, while the execution time of the private chain on Ethereum has slight fluctuations. The main reason lies in the randomness of the block producing rate in POW consensus mechanism. More specifically, from the results of Fig. 4(d), one can find that with small transaction volume, the execution time of Hyperledger is 3–5 s shorter than that of Ethereum.

From Fig. 4(b), we can conclude that Hyperledger outperforms Ethereum in terms of throughput. The throughput of both Hyperledger and Ethereum platforms is enhanced first, when the transaction volume grows up and then, is reduced when the transaction volume becomes larger than a certain value. Such a certain value to obtain the maximum throughput indicates the platform capacity, and performance bottleneck will appear when the transaction volume is larger than this certain value. In Fig. 4(b), Hyperledger achieves its maximum throughput when the transaction volume is around 1500, and Ethereum reaches its maximum throughput when the transaction volume is 1000, which implies that Hyperledger has a larger capacity.

Next, the throughput with the transaction volume smaller than 50 is analyzed. In Fig. 4(e), the results show that: 1) the throughput of both Hyperledger and Ethereum increases steadily with the increase of the transaction volume and 2) the throughput of Hyperledger is about three times the throughput of Ethereum.

Finally, Fig. 4(c) presents the latency of Hyperledger and Ethereum by verifying different numbers of transactions. As can be seen from the figure, when the transaction volume is increased, the latency of both platforms grows, and the latency of the Hyperledger is shorter than that of Ethereum. To obtain more

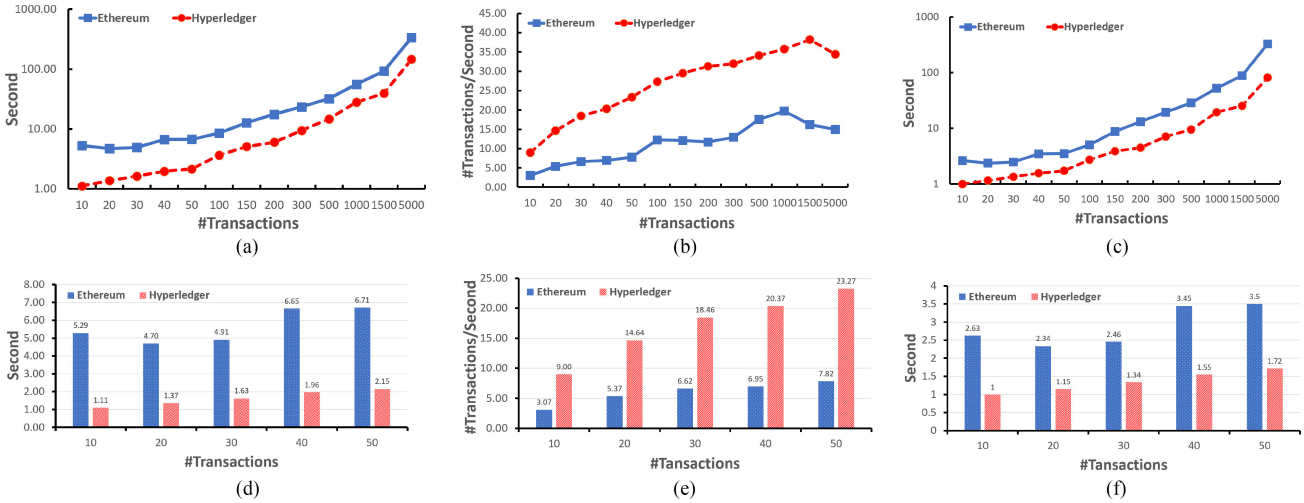


Fig. 4. Performance comparison between PBFT and Casper. (a) Semilog plot on execution time with maximum 5000 transactions. (b) Semilog plot on throughput with maximum 5000 transactions. (c) Semilog plot on latency with maximum 5000 transactions. (d) Execution time with maximum 50 transactions. (e) Throughput with maximum 50 transactions. (f) Latency with maximum 50 transactions.

insights, we also compare five sets of transactions in Fig. 4(f) and find that the latency of the Hyperledger is about half of Ethereum.

The above comprehensive and in-depth comparisons confirm that the PBFT consensus performs better than the current mainstream consensus and is suitable to be deployed on our zkCrowd to satisfy the needs of crowdsourcing.

Remarks: Our local experiment environment and hardware configuration may limit the performance of Hyperledger, Ethereum, and EOS.io. Nevertheless, the limited experiment settings do not influence the comparison results and conclusions.

VIII. Conclusion

In this article, a hybrid blockchain platform, named zkCrowd, was proposed for crowdsourcing. By integrating a public chain running the DPOS consensus and subchains running the PBFT consensus, our zkCrow can achieve nice features: 1) higher transaction throughput and less execution time compared with traditional POW/POS-based blockchain; 2) diverse privacy protection and access control for different crowdsourcing tasks by employing smart contracts and ZK-proof; and 3) resistance to severe attacks outperforming the state of the art. Finally, intensive experiments were performed to validate the effectiveness of zkCrowd via showing the superiority of DPOS and PBFT over Casper.

As our pilot work, the experiments in this article mainly focussed on the efficiency of consensuses used on zkCrowd. In our future work, the fundamental functions of zkCrowd and cross-chain communications will be realized, and the performance of the whole platform will be further improved.

References

[1] *Wikipedia Down? Current Status and Problems*. Accessed: Jan. 18, 2019. [Online]. Available: https://downdetector.com/status/wikipedia

[2] 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 Things J.*, vol. 2, no. 6, pp. 562–572, Dec. 2015.

[3] M. Isaac, K. Benner, and S. Frenkel, "Uber hid 2016 breach, paying hackers to delete stolen data," 2017. [Online]. Available: https://www.nytimes.com/2017/11/21/technology/uber-hack.html

[4] 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.

[5] Y. Wang, Z. Cai, Z. Chi, X. Tong, and L. Li, "A differentially k-anonymity-based location privacy-preserving for mobile crowdsourcing systems," in *Proc. Int. Conf. Identification, Inf. Knowl. Internet Things*, Shandong, China, 2017, pp. 28–34.

[6] 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, Aug. 2017.

[7] Y. Zhang and M. van der Schaar, "Reputation-based incentive protocols in crowdsourcing applications," in *Proc. IEEE Conf. Comput. Commun.*, Orlando, FL, USA, 2012, pp. 2140–2148.

[8] C. Tanas, S. Delgado-Segura, and J. Herrera-Joancomartí, "An integrated reward and reputation mechanism for MCS preserving users' privacy," in *Proc. 10th Int. Workshop Data Privacy Manage., Secur. Assurance*, 2015, pp. 83–99.

[9] 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.

[10] 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, 2018, Art. no. 3894.

[11] Y. Lu, Q. Tang, and G. Wang, "Zebralancer: Private and anonymous crowdsourcing system atop open blockchain," in *Proc. 38th IEEE Int. Conf. Distrib. Comput. Syst.*, Vienna, Austria, 2018, pp. 853–865.

[12] S. Matsumoto and R. M. Reischuk, "IKP: Turning a PKI around with decentralized automated incentives," in *Proc. IEEE Symp. Secur. Privacy*, San Jose, CA, USA, 2017, pp. 410–426.

[13] 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 *Proc. IEEE Symp. Secur. Privacy*, 2015, pp. 104–121.

[14] *Smilo: The Hybrid Blockchain Platform With a Conscience*. Accessed: Jul. 28, 2019. [Online]. Available: https://smilo.io/

[15] *Aergo: It's Not a Blockchain. It's the Blockchain for Business*. Accessed: Jul. 28, 2019. [Online]. Available: https://www.aergo.io/

[16] *Xinfin: Enterprise Ready Hybrid Blockchain For Global Trade And Finance*. Accessed: Jul. 28, 2019. [Online]. Available: https://www.xinfin.org/

[17] D. Chatzopoulos, S. Gujar, B. Faltings, and P. Hui, "Privacy preserving and cost optimal mobile crowdsensing using smart contracts on blockchain," in *Proc. 15th IEEE Int. Conf. Mobile Ad Hoc Sensor Syst.*, Chengdu, China, 2018, pp. 442–450.

[18] J. Zou, B. Ye, L. Qu, Y. Wang, M. A. Orgun, and L. Li, "A proof-of-trust consensus protocol for enhancing accountability in crowdsourcing services," *IEEE Trans. Serv. Comput.*, vol. 12, no. 3, pp. 429–445, May/Jun. 2019.

[19] A. E. Kosba, A. Miller, E. Shi, Z. Wen, and C. Papamanthou, "Hawk: The blockchain model of cryptography and privacy-preserving smart contracts," in *Proc. IEEE Symp. Secur. Privacy*, San Jose, CA, USA, 2016, pp. 839–858.
[20] G. Zyskind, O. Nathan, and A. Pentland, "Decentralizing privacy: Using blockchain to protect personal data," in *Proc. IEEE Symp. Secur. Privacy Workshops*, San Jose, CA, USA, 2015, pp. 180–184.
[21] K. Samani, "Delegated proof of stake: Features and tradeoffs," 2018. [Online]. Available: https://multicoin.capital/wp-content/uploads/2018/03/DPoS-Features-and-%Tradeoffs.pdf
[22] *EOS: The Most Powerful Infrastructure for Decentralized Applications*. Accessed: Jul. 14, 2019. [Online]. Available: https://eos.io/
[23] *Bitshares Blockchain: Industrial-Grade Decentralized Platform*. Accessed: Jul. 14, 2019. [Online]. Available: https://bitshares.org/
[24] 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 *Proc. 36th IEEE Symp. Rel. Distrib. Syst.*, Hong Kong, 2017, pp. 253–255.
[25] M. Fredrikson and B. Livshits, "Zø: An optimizing distributing zero-knowledge compiler," in *Proc. 23rd USENIX Secur. Symp.*, San Diego, CA, USA, 2014, pp. 909–924.
[26] B. Parno, J. Howell, C. Gentry, and M. Raykova, "Pinocchio: Nearly practical verifiable computation," *Commun. ACM*, vol. 59, no. 2, pp. 103–112, 2016.
[27] M. Castro and B. Liskov, "Practical byzantine fault tolerance," in *Proc. 3rd USENIX Symp. Oper. Syst. Des. Implementation*, New Orleans, LA, USA, 1999, pp. 173–186.
[28] E. Ben-Sasson, A. Chiesa, D. Genkin, E. Tromer, and M. Virza, "SNARKs for C: Verifying program executions succinctly and in zero knowledge," in *Proc. Adv. Cryptology - 33rd Annu. Cryptology Conf.*, Santa Barbara, CA, USA, 2013, pp. 90–108.
[29] K. Croman *et al.*, "On scaling decentralized blockchains (A position paper)," in *Proc. Int. Workshops Financial Cryptography Data Secur.*, Christ Church, Barbados, 2016, pp. 106–125.
[30] A. Hertig, "How will Ethereum scale?" Accessed: Feb. 4, 2019. [Online]. Available: https://www.coindesk.com/information/will-ethereum-scale
[31] A. Miller, A. E. Kosba, J. Katz, and E. Shi, "Nonoutsourceable scratch-off puzzles to discourage bitcoin mining coalitions," in *Proc. 22nd ACM SIGSAC Conf. Comput. Commun. Secur.*, Denver, CO, USA, 2015, pp. 680–691.
[32] G. C. Fanti, L. Kogan, S. Oh, K. Ruan, P. Viswanath, and G. Wang, "Compounding of wealth in proof-of-stake cryptocurrencies," 2018, *arXiv:1809.07468*.
[33] *Amazon Mechanical Turk: Access a Global, On-Demand, 24x7 Workforce*. Accessed: Aug. 4, 2019. [Online]. Available: https://www.mturk.com/
[34] N. Salehi *et al.*, "We are dynamo: Overcoming stalling and friction in collective action for crowd workers," in *Proc. 33rd Annu. ACM Conf. Human Factors Comput. Syst.*, Seoul, Korea, 2015, pp. 1621–1630.
[35] V. Buterin and V. Griffith, "Casper the friendly finality gadget," 2017, *arXiv:1710.09437*.

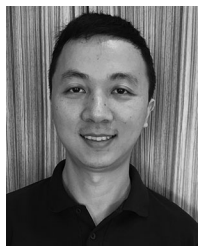
Saide Zhu received the B.S. degree in telecommunications from Nanjing University of Posts and Telecommunications, Nanjing, China, in 2014, and the M.S. degree in computer engineering from Boston University, Boston, MA, USA, in 2016. He is currently working toward the Ph.D degree at the Department of Computer Science, Georgia State University, Atlanta, GA, USA.

His research interests include blockchain, Internet of Things (IoT) security and privacy, and mobile computing.

Zhipeng Cai (SM'06) received the B.S. degree from the Department of Computer Science and Engineering, Beijing Institute of Technology, Beijing, China, in 2001, and the M.S. and Ph.D. degrees from the Department of Computing Science, University of Alberta, Edmonton, AB, Canada, in 2004 and 2008, respectively.

He is currently an Associate Professor with the Department of Computer Science, Georgia State University (GSU), Atlanta, GA, USA. Prior to joining GSU, he was a Research Faculty with the School of Electrical and Computer Engineering, Georgia Institute of Technology, Atlanta, GA, USA. His research interests include networking and big data.

Dr. Cai is the recipient of an NSF CAREER Award.



Huafu Hu received the B.S. degree from Hangzhou Dianzi University, Hangzhou, China, in 2007, and the M.S. degree from Georgia State University, Atlanta, GA, USA, in 2019, both in computer science.

He is currently a Software Engineer with Omega Bio-tech Inc. His research interest includes blockchain, machine learning, and data mining.

Yingshu Li (SM'04) received the B.S. degree from the Department of Computer Science and Engineering, Beijing Institute of Technology, Beijing, China, in 2001, and the M.S. and Ph.D. degrees from the Department of Computer Science and Engineering, University of Minnesota-Twin Cities, Minneapolis, MN, USA, in 2003 and 2005, respectively.

She is currently an Associate Professor with the Department of Computer Science, Georgia State University, Atlanta, GA, USA. Her research interests include Wireless networking, sensor networks, sensory data management, social networks, and optimization.

Dr. Li is the recipient of an NSF CAREER Award.

Wei Li (M'12) received the M.S. degree from the Beijing University of Posts and Telecommunications, Beijing, China, in 2011, and the Ph.D. degree from George Washington University, Washington, DC, USA, in 2016, both in computer science.

She is currently an Assistant Professor with the Department of Computer Science, Georgia State University, Atlanta, GA, USA. Her current research interests include the areas of security and privacy for the Internet of Things and cyber-physical systems, secure and privacy-aware computing, big data, blockchain, game theory, and algorithm design and analysis.

Dr. Li was the recipient of the Best Paper Awards in wireless algorithms, systems, and applications (WASA) 2011 and ACM MobiCom cognitive radio architectures for broadband (CRAB) 2013. She is a member of Association for Computing Machinery.