Decentralized IoT Resource Monitoring and Scheduling Framework Based on Blockchain

Dawei Li, Ruonan Chen, Qinjun Wan, Zhenyu Guan, *Member, IEEE*, Yu Sun, Qianhong Wu, Jiankun Hu, *Senior Member, IEEE*, and Jianwei Liu, *Member, IEEE*

***Abstract*—With the continuous advancement of edge intelligence, edge servers undertake more and more intelligent computing tasks. Nowadays, there are a large number of Internet of Things (IoT) devices in the network in an idle state. For instance, the mining process for the consensus of miners in blockchain such as Bitcoin causes a waste of computing resources and energy. A natural question arises: can we couple the idle computing resources of network devices to continuously and credibly share the burden of edge intelligent computing tasks in a secure manner? The answer of this article is yes. We propose a blockchain-based IoT resource monitoring and scheduling framework that supports resource management and trusted edge computing. We analyze the security threats in all phases of distributed edge computing, and utilize the trusted computing and public verifiability features of blockchain to ensure reliability and fairness in the trusted measurement of device computing power, the decomposition of intelligent computing tasks, the matching of task and computing power, and the verification of computing result. Finally, we implement a simulation on the edge network by performing a distributed machine learning task for weather prediction, and the simulation results demonstrate the availability of our scheme.**

***Index Terms*—Blockchain, edge intelligence, resource monitoring, resource scheduling.**

I. Introduction

AS VARIOUS technologies, such as sensor networks, big data analytics, and artificial intelligence continue to mature, the number of application tasks is exploding. According to the prediction, the data market is expected to reach U.S. $11 866.34 billion in 2025, with a compound annual growth rate (CAGR) of 9%. Large amounts of data put great pressure on storage and data processing. These data processing tasks are mainly computationally intensive. Edge computing [1], [2] is one of the mitigation methods for addressing this issue [3], [4], [5], [6], which provides cloud computing capability at the edge of the network near end users [7], [8], [9], [10]. The core idea is to move network functions and resources closer to end users, including communication capability, computing power, and cache resources.

Besides, crowdsourcing is an appealing choice to solve the issue above, which could reduce the computing cost of the entire system, and it is widely used in edge computing and Internet of Things (IoT) system [11], [12]. Most of the existing crowdsourcing platforms are centralized, which faces the risk of single point of failure and distributed denial-of-service attacks [13], [14]. Although many decentralized crowdsourcing systems have been proposed, many of them are unable to verify the correctness of computing results [15], [16], [17]. Therefore, it is natural to think of sharing computational tasks with nodes in the distributed network that have spare computing resources.

It is a challenging issue to reasonably schedule computing resources so that the tasks to be processed and computing resources are in a good condition. A single node/server cannot allocate tasks in accordance with its best advantage. The existing computing resource scheduling algorithms are basically based on cloud computing. There is no optimal computing resource scheduling algorithm available for the secure edge computing and distributed setting. Based on the investigation of previous work, in this article, we utilize the techniques of blockchain, resource scheduling algorithms, and smart contracts to construct a blockchain-based IoT resource monitoring and scheduling framework, which takes the advantage of nodes in the P2P network equipped with powerful computing resources, for solving the issue of the imbalance between the idle computing resources of network nodes and the heavy workload of tasks. For resource monitoring, computing nodes cannot deceive the network regarding their own resource conditions and task allocation.

Contributions: The contributions of our work are summarized as follows.

1) We introduce the concept of ResourceCoin and analyze the potential threats that are vulnerable to the edge computing network. A blockchain-based IoT resource monitoring and scheduling framework is proposed. This framework can effectively utilize idle resources dedicated to blockchain mining in addressing the problem of the sharp computing load. It can also address the revealed security threats involved in the threat model.

Manuscript received 26 August 2022; revised 1 November 2022; accepted 9 December 2022. Date of publication 13 December 2022; date of current version 7 December 2023. This work was supported in part by the National Key Research and Development Program of China under Project 2021YFB2700200; in part by the Natural Science Foundation of China under Project 62002006, Project 62172025, Project U21B2021, Project 61932011, Project 61932014, Project 61972018, Project 61972019, Project 61772538, Project 32071775, and Project 91646203; and in part by the Defense Industrial Technology Development Program under Grant JCKY2021211B017. *(Corresponding author: Zhenyu Guan.)*

Dawei Li, Ruonan Chen, Qinjun Wan, Zhenyu Guan, Yu Sun, Qianhong Wu, and Jianwei Liu are with the School of Cyber Science and Technology, Beihang University, Beijing 100191, China (e-mail: guanzhenyu@buaa.edu.cn).

Jiankun Hu is with the School of Engineering and Information Technology, The University of New South Wales, Sydney, NSW 2052, Australia.

Digital Object Identifier 10.1109/JIOT.2022.3228799

2) The fairness of outsourced computing in a distributed network has always been difficult to guarantee. First, it is necessary to prevent computing nodes from submitting random wrong results for defrauding rewards or simply copying the results submitted by others to reap without sowing. Besides, it is essential to prevent task publishers from not paying after obtaining the computing results. Our protocol utilizes the deposit strategy to prevent the task publisher from withholding payment. We protect the privacy of task results to prevent lazy computing nodes from copying results.
3) The traditional distributed task scheduling schemes always introduce a centralized trusted third party to distribute tasks to computing nodes in general. The assumption of a trusted third party is too strong to be realized in practice. Our protocol achieves decentralized task scheduling and can monitor the resource condition of the entire network.

II. Related Works

A. Blockchain-Based Task Scheduling

The task scheduling algorithm belongs to the NP-hard problem since some performance indicators are mutually opposed, which is impossible to optimize all indicators [18], [19], [20], [21], [22], [23], [24], [25]. Thus, the approximate optimal solution is generally selected to obtain a relatively balanced scheduling result. With the maturity of blockchain technology, more and more researchers use the excellent features of blockchain to improve the process of task scheduling. Baniata et al. presented a blockchain-assisted scheduling model, which used ant colony optimization (ACO) algorithm [26], [27], [28] to execute resource scheduling, and they reduced the latency by incorporating fog computing [29]. However, they did not verify the results of the computational tasks; thus, there may be dishonest miners defrauding task rewards with the wrong results. Li et al. [30] designed a decentralized cloud-fog-edge task scheduling model based on blockchain, and used Berger's fairness theory to achieve the fair task scheduling.

B. Crowdsourcing

Crowdsourcing has attracted the attention of many researchers since it was proposed [31], [32], [33], [34], [35], [36]. Crowdsourcing is a new business model which could help people distribute their tasks, seek solutions to difficult problems, or look for some ideas over the network, etc. Common crowdsourcing models are based on the centralized platform, where the crowdsourcing model mainly includes requesters, workers, and a centralized crowdsourcing platform.

Combining the crowdsourcing with blockchain could mitigate the defects of the centralized crowdsourcing model. Li et al. proposed a decentralized crowdsourcing framework called CrowdBC [33], where the task published by a requester is completed by a group of workers without a platform. The workers who are dedicated to completing the task would be paid a certain amount of rewards. Han et al. [34] proposed a crowdsourcing framework based on the blockchain, which achieves result verification by zero-knowledge proof. However, this framework does not protect the privacy of the solutions, which may allow some workers copy the submitted solutions for rewards without an effort. Lu et al. presented a decentralized crowdsourcing system called ZebraLancer [37], where the main contributions of their proposal are achieving the confidentiality of solutions and protecting the privacy of participants, respectively.

III. System Model and Threat Model

A. System Model

In this part, we describe the system model of the blockchain-based IoT resource monitoring and scheduling framework. The system model consists of four entities, namely, P2P network, task publishers, computing nodes, and blockchain. The details are as follows.

P2P Network: In the proposed scheme, all nodes P_i involved are in the P2P network. Our proposal takes the advantage of P2P network with scalability, decentralization, robustness, and load balancing.

Computing Nodes: The IoT devices with powerful idle computing resources act as computing nodes in the P2P network. The computing node is responsible for performing the published computational tasks and maintaining the stable operations of the whole blockchain.

Task Publishers: The peers in the P2P network with heavy computational tasks act as task publishers. They are willing to outsource these tasks to the computing nodes and pay for a certain amount of ResourceCoin as rewards.

Blockchain: Blockchain is a distributed ledger that can be used for data security storage, privacy protection, and trusted execution in distributed networks [38], [39]. The main operations in the scheme are executed by the smart contracts, including deposit payment, resource condition checking, task allocation, task results verification, and computational resource monitoring.

B. Threat Model

In this part, we discuss some potential attacks that are vulnerable to the blockchain-based IoT resource monitoring and scheduling framework.

1) *Spoof Attack:* Spoofing attack originates from IP spoofing. After receiving the calculation results, malicious task publishers may go offline instead of paying the promised ResourceCoin. Even the tasks $T_i = \{T_{\text{addr}}, T_{\text{ID}}, T_{\text{result}}\}$ released by the malicious publisher at the beginning are unsolvable, and they aim to maliciously occupy computing resources.
2) *Sybil Attack:* Sybil attack was proposed by John in 2002, which is a form of attacks that acts on P2P networks. The adversary $\mathcal{A}$ uses a single node to forge multiple fake identities $\{P_k\}_{1\leq k\leq n}$ in the P2P network, which could greatly weaken the redundancy of the network and reducing network robustness, even interfere with normal network activities.

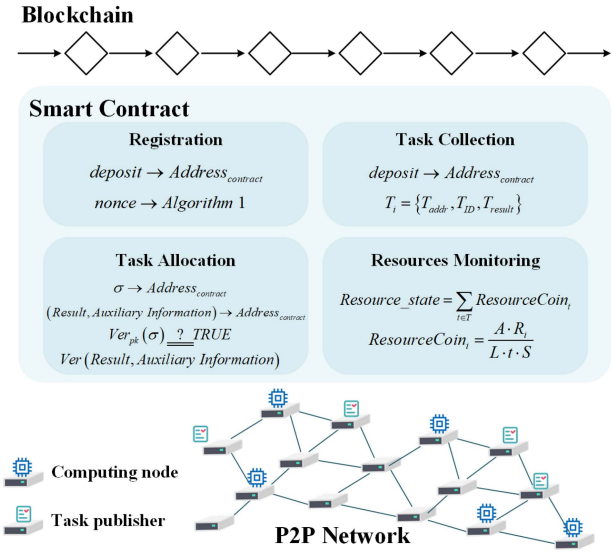


Fig. 1. Basic idea overview.

3) *Forgery Attack:* The adversary $\mathcal{A}$ has the ability to forge or randomly guess the result T_{result} of a computational task, which passes the verification algorithm and the adversary obtains the rewards successfully.
4) *Free-Riding Attack:* The adversary $\mathcal{A}$ has the ability to pass the verification for a task with the task results T_{result} of others and some related information.

IV. Construction

A. Basic Idea

In our proposal, we take the advantage of the powerful computing resources of peers in the P2P network to solve the computational tasks. They dedicate to complete the tasks to earn ResourceCoin. The task publishers are some resource-constrained peers in the P2P network, which are trapped by many expensive computational tasks. They publish the tasks to the public for seeking the results, and reward with the ResourceCoin. Moreover, the protocol could monitor the resources owned by computing nodes according to their possession of the ResourceCoin. The protocol mainly consists of four phases, namely, computing node registration, task collection, task allocation, and resource monitoring. The interaction between nodes and smart contracts is shown in Fig. 1.

B. Computing Node Registration

For a peer P_i in the P2P network who wants to invest its computational resource for the monetary reward, it first needs to register as a computing node. In order to resist Sybil attacks, the contract requires P_i to deposit a sum of coins as collateral to the contract address. Then, the smart contract measures the resource condition of the peer by testing its computing power, where the testing principle is similar to the Proof-of-Work (PoW) mechanism. Specifically, the smart contract challenges P_i with an analogous PoW puzzle, and evaluates the computing power of the peer by the resource condition checking algorithm. Before running the algorithm, the smart contract first computes an index value by $\text{Index} \leftarrow \phi(Tx_{de_i})$, $\text{Index} \in [1, 6]$, where Φ is a pseudo-random function, and the index value indicates the function that will be used in the following algorithm by P_i. The functions include the operations of the multiplication of floating-point number matrices (FPMM), *Modular Exponentiation*, SHA-1, SHA-256, SHA-512, and SM3, respectively. Randomly presetting different functions could prevent some Bitcoin proprietary circuits from being able to pass the algorithm easily, while in fact they have insufficient computing power to support the following task calculations.

Algorithm 1 Resource Condition Checking Algorithm

Input: The peer P_i, Pre_h, Tx_{de_i}, *Index*
Input: $Nonce_i$, the difficulty *diff*, the time interval $\triangle t$
Output: pass/reject

```
if Function_Index(Pre_h || Tx_de_i) < diff then
    Pre_h ++
end if
if Function_Index(Pre_h || Tx_de_i || Nonce_i) > diff ∨ △t < t_finish − t_deploy then
    return ⊥
end if
```

As shown in Algorithm 1, the algorithm takes the hash value Pre_h of the previous block, the identifier Tx_{de_i} of the deposit transaction by P_i, the index value Index as inputs, and presets the difficulty diff, the time interval $\triangle t$, which requires the peer P_i to compute a value Nonce_i such that $\text{Function}_{\text{Index}}(\text{Pre}_h \| Tx_{de_i} \| \text{Nonce}_i) < \text{diff}$, where $\text{Function}_{\text{Index}}()$ is a specific function randomly specified before the algorithm is executed. Besides, the algorithm requires that the peer should output a nonce which satisfies the condition within a certain time interval, i.e., $\triangle t > t_{\text{finish}} - t_{\text{deploy}}$. If P_i figures out a nonce value that satisfies the condition, as well as the execution time is within the prescribed time interval, then P_i passes the resource condition checking.

After that, if the peer passes the resource condition checking, the contract will return the collateral. Otherwise, it will be removed. After registration, the contract sends a small amount of initial ResourceCoin to the peer.

C. Task Collection

For the peers in the P2P network which are resource-constrained, they want to outsource their computational tasks and promise a certain amount of ResourceCoin as a reward. The process of task collection is shown in Algorithm 2. Before publishing the tasks, the peer (called task publisher) deposits the rewards into the smart contract for preventing the spoofing attack, i.e., $Tx_{\text{deposit}\rightarrow\text{Address}_{\text{contract}}}$. Then, the task publisher uploads the task information on the blockchain, where the task model is formalized as $T_i = \{T_{\text{addr}}, T_{\text{ID}}, T_{\text{result}}\}$. Specifically, T_{addr} denotes the address where the task is stored, T_{ID} denotes the identifier of the task, and T_{result} is the computation result of the task T_i, which is initialized to 0. Later, if the task result T_{result} returned from computing nodes passes the verification by the smart contract, i.e., $\text{Verification}(T_{\text{result}}) = 1$, then the

Algorithm 2 Task Collection Algorithm

Input: The task to be published $T_i = \{T_{addr}, T_{ID}, T_{result}\}$
Input: *deposit*
Output: $T_i.T_{result}$

```
if Tx_{deposit→Address_{contract}} = 0 then
    return ⊥
end if
if Verification(T_{result}) = 1 then
    T_i.T_{result} ← T_{result}
end if
```

Algorithm 3 Task Allocation Algorithm

Input: A task to be allocated $T_i = \{T_{addr}, T_{ID}, T_{result}\}$
Input: The computing nodes to be allocated
Output: The allocation result

```
Parse T_i = {{T_{serial_j}}_{1≤j≤s}, {T_{parallel_k}}_{1≤k≤p}}
while TRUE do
    if T_{il} ∈ {T_{serial_j}}_{1≤j≤s} then
        call MET
    end if
    if T_{il} ∈ {T_{parallel_k}}_{1≤k≤p} then
        call ACO
    end if
    if Verification(T.result) = 1 then
        return ⊥
    end if
end while
```

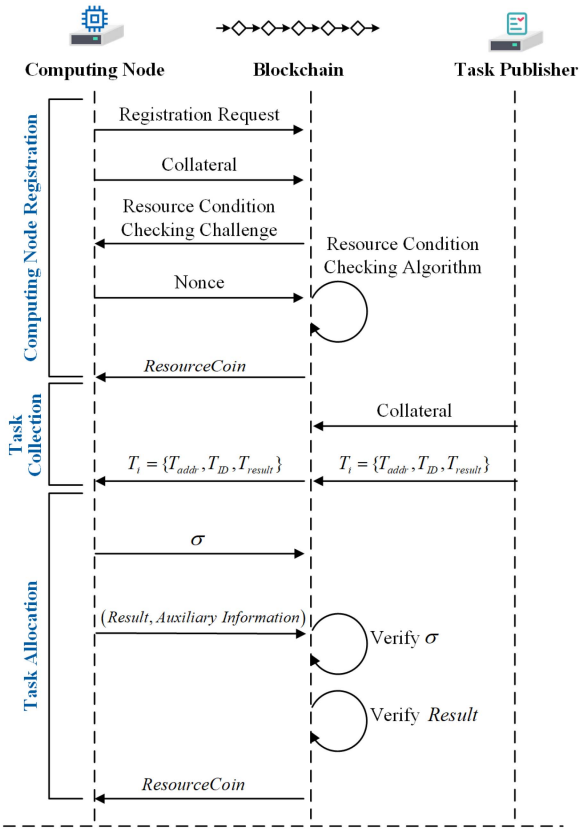


Fig. 2. Task management process.

contract fills the result into $T_i.T_{\text{result}}$, and the computing node obtains the reward released by the contract address. Otherwise, the computing node will get nothing.

D. Task Allocation

The process of task allocation is shown in Algorithm 3. When a task publisher successfully publishes a new task $T_i = \{T_{\text{addr}}, T_{\text{ID}}, T_{\text{result}}\}$ on the blockchain, the smart contract first parses it into a directed acyclic graph (DAG), and then uses the depth-first search (DFS) algorithm idea of the graph traversing to disassemble the task into serial tasks $\{T_{\text{serial}_j}\}_{1\le j\le s}$ and parallel tasks $\{T_{\text{parallel}_k}\}_{1\le k\le p}$. As the processing speed of the minimum execution time (MET) algorithm is relatively fast, our protocol uses the MET algorithm to schedule serial tasks $\{T_{\text{serial}_j}\}_{1\le j\le s}$. Meanwhile, as the limitation of a smart contract, the protocol uses the ACO algorithm [26], [27], [28], [40] to schedule parallel tasks $\{T_{\text{parallel}_k}\}_{1\le k\le p}$. When the computing node returns the computing result of the assigned task, it first returns the signature σ of the result, and then sends back the computing result T.result as well as the auxiliary information of the task, where the auxiliary information is the intermediate results during the computation. On receiving the computing result T.result, the smart contract first verifies the validity of the signature σ of the computing node. If the signature is invalid, i.e., Verification(T.result) $= 0$, then the smart contract rejects the result. Otherwise, the smart contract verifies the correctness of the computing result according to the auxiliary information, then sends the reward of the task to the computing node if the verification is passed. Otherwise, the reward is still locked in the contract address, and the computing will get nothing. The process of task management is shown in Fig. 2.

E. Resource Monitoring

As we mentioned above, when a new computing node wishes to join in the task computation, the resource condition checking algorithm first verifies the resource condition of the peer and then the computing node will get some ResourceCoin as a proof of its resource power if the peer passes the checking algorithm. After registration, the computing nodes could invest their computing resource and earn more ResourceCoin.

Therefore, the stability of a computing node in possession of ResourceCoin reflects the status of the node's computing resources. Thus, we use the amount of ResourceCoin during a period T to measure the resource power of the node. Specifically, we sum the ResourceCoin obtained in the last period of time, then we can get

$$\text{Resource\_state} = \sum_{t\in T} \text{ResourceCoin}_t.$$

Among them, ResourceCoin_t refers to the ResourceCoin obtained at time t, and Resource_state measures the execution state of the past tasks in the last period of time T.

By monitoring the ResourceCoin holding by the computing nodes in the P2P network, the resource condition of nodes can be estimated. If the ResourceCoin stability of a computing

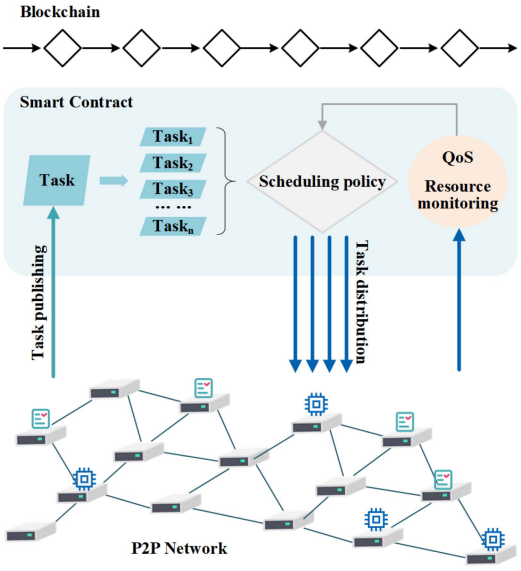


Fig. 3. Resource scheduling framework.

node has changed greatly within a period of time, then the smart contract will apply for a new round of resource condition checking for this node.

In the blockchain-based IoT resource monitoring and scheduling framework, ResourceCoin is an incentive measure. Nodes that want to publish tasks need to pay a certain amount of ResourceCoin as collateral, and the computing nodes can get ResourceCoin as a reward by executing tasks in the network. Therefore, the computing nodes in the network will actively performing the assigned tasks for earning more ResourceCoin. The process of resource scheduling is shown in Fig. 3.

V. Analysis and Simulation

In this section, we evaluate the performance of the scheme proposed in previous sections from two aspects. On the one hand, we analyze the security of the entire framework. On the other hand, we simulate the core parts of resource monitoring and task management in the protocol.

A. Security Analysis

Taking the security issues that the system may face into account, we analyze the impact of several attacks to the system.

Theorem 1: Assuming the existence of a secure smart contract, then the blockchain-based IoT resource monitoring and scheduling framework above is secure against the spoof attack.

Proof: In our protocol, task publishers are required to pay a deposit to the contract account after uploading their tasks. If the subsequent compute nodes calculate the result and upload it, the reward will be released directly from the contract account. The dishonest task publisher cannot obtain the computing result of a task without the payment. ■

Theorem 2: Assuming the existence of a secure smart contract, then the blockchain-based IoT resource monitoring and scheduling framework above is secure against the sybil attack.

Proof: In order to prevent malicious nodes from forging multiple identities and sending resource verification requests to the system at the same time, we charge a certain ResourceCoin as a deposit for each node that will perform resource verification. If the verification is passed, the deposit will be returned. Otherwise, the deposit will be deducted. ■

Theorem 3: Assuming the existence of a secure smart contract and an existentially unforgeable signature scheme, then the blockchain-based IoT resource monitoring and scheduling framework above is secure against the forgery attack and free-riding attack.

Proof: When the computing node returns the result, the smart contract verifies the computing result based on the returned auxiliary information. The malicious computing node cannot get rewards by randomly guessing or forging the result. Dishonest computing nodes cannot get rewards by learning the results of others for submission, since the computing nodes first sign their results when returning the results, and then submit the computing results. This reply order could not be disrupted. After that, the smart contract verifies the correctness of signed results. ■

B. Comparison With Existing Schemes

We compare features of our proposal with the existing works [29], [30], [33], [34], [35], [36], [41], [42], including spoof attack resistance, free-riding attack resistance, result verifiability, task decomposition, resource scheduling, and resource monitoring, which is shown in Table I. Result verifiability means that the correctness of the computing task results could be verified after the results have been submitted. Task decomposition refers to that computing tasks can be decomposed during the protocol process to improve the efficiency of task processing. Resource scheduling refers to that computing tasks are allocated according to certain rules and algorithms rather than a randomly selected or assigned during protocol. As we can see, the majority of them do not support spoof attack resistance and task decomposition. None of the literature considered resource monitoring. Our proposal achieves all of these features.

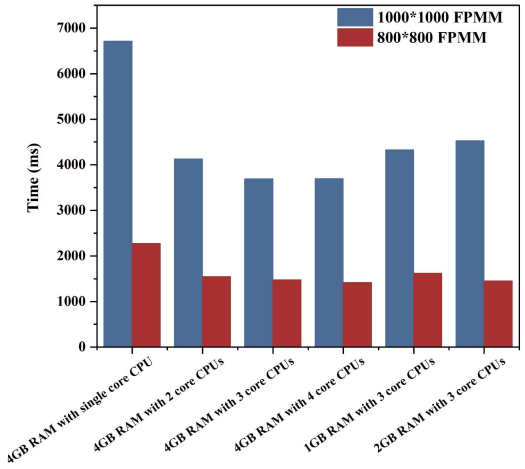
C. Implementation Performance

The implementation of the system is mainly divided into three parts, namely, the computing power testing for different operations, the simulation of the smart contracts in our proposal, and resource monitoring, respectively.

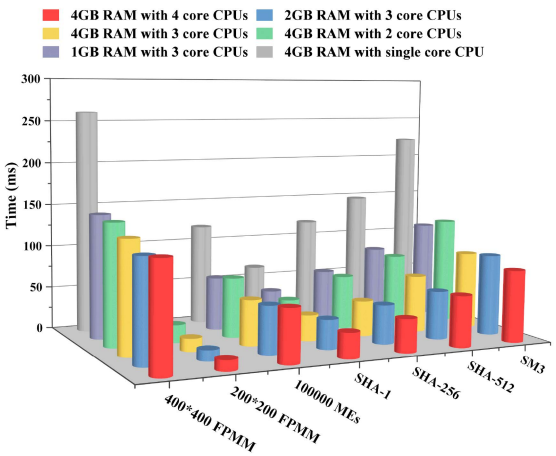
We test the latency time of computing node registration for different types of operations. The experimental parameter is shown in Table II. Specifically, we test the time consumption of a laptop with different memories and CPU cores to execute four different hash operations, modular exponentiation (ME), and FPMM of different specifications. The hash functions involved in the resource condition checking algorithm include Chinese SM3, SHA-512, SHA-256, and SHA-1. We test the time delay of ME for 100 000 times. The matrices used for floating-point number multiplication operations include $200 * 200$ matrix, $400 * 400$ matrix, $800 * 800$ matrix, and $1000 * 1000$ matrix. The experimental results are shown in

TABLE I
FUNCTIONAL COMPARISON WITH THE EXISTING SCHEMES

Ref.	Spoof Attack Resistance	Free-riding Attack Resistance	Result Verifiability	Task Decomposition	Resource Scheduling	Resource Monitoring
Baniata *et al.* [29]	✗	✗	✗	✗	✓	✗
Li *et al.* [30]	✗	✗	✗	✗	✓	✗
Nguyen *et al.* [41]	✗	✗	✗	✗	✓	✗
Li *et al.* [42]	✗	✗	✗	✗	✓	✗
Li *et al.* [33]	✗	✓	✓	✗	✗	✗
Han *et al.* [34]	✗	✗	✓	✓	✗	✗
Wang *et al.* [35]	✗	✗	✗	✗	✓	✗
Lin *et al.* [36]	✓	✓	✓	✗	✗	✗
Our Proposal	✓	✓	✓	✓	✓	✓



(a)



(b)

Fig. 4. Computing power testing. (a) Time cost of different device for FPMM. (b) Time cost of different device for different operations.

Fig. 4. In order to enhance the expressive effect, we put the figure of time cost of FPMM for 800∗800 matrix and 1000∗1000 matrix in Fig. 4(a), and other contents in Fig. 4(b). From the figure, we can see that SM3 has the longest time delay and SHA-1 has the shortest among the four hash functions.

TABLE II
EXPERIMENTAL PARAMETER SETTING

Memory	1GB, 2GB, 4GB
The Number of CPU Cores	1 core, 2 cores, 3 cores, 4 cores
The Type of Operations	FPMM, Modular Exponentiation, SHA-1, SHA-256, SHA-512, SM3

The time cost of four different FPMM increases successively, where the time cost of $1000 * 1000$ matrix FPMM for 1GB RAM with 3 core CPUs is 4327.52 ms and $200 * 200$ matrix FPMM takes 12.63 ms for 4-GB RAM with 4 core CPUs. Generally, under the same conditions, the more CPU cores, the lower the time delay.

We deploy the smart contracts on the Hyperledger Fabric platform on a desktop in the Cent OS 7 with a quad-core processor, 4-GB RAM, 20 GB. The functions in the smart contracts involved in our protocol consist of computing node registration, task collection, task allocation and resource monitoring. The time cost of running smart contracts is shown in Fig. 5. We test the performance of our protocol with a weather prediction task. Specifically, we deploy computing nodes and perform distributed machine learning tasks on the Occupancy Detection[1] data set, which is contributed by Luis Candanedo from the University of Mons. The main algorithm used in this task is the Naive Bayes algorithm.

First, the weather prediction task is divided into five subtasks, which is shown in Fig. 5(a). Subtask A is set to read and divide the data set. Subtasks B–D are mainly training processes, which are responsible for data statistic and composed of the Naive Bayes algorithm. Subtask E is making predictions for the training set. The smart contract assigns the serial subtask A and subtask E to the computing nodes with MET algorithm, and assigns the subtask B, subtask C, and subtask D to computing nodes with the ACO algorithm. The time cost of task allocation and task execution is shown in Fig. 5(b). From the figure, it can be seen that assigning the parallel tasks with the ACO algorithm takes 1.6656 ms, and assigning the serial tasks with MET algorithm takes 31.66 ms. If the task is

[1]https://archive-beta.ics.uci.edu/ml/datasets/occupancy+detection

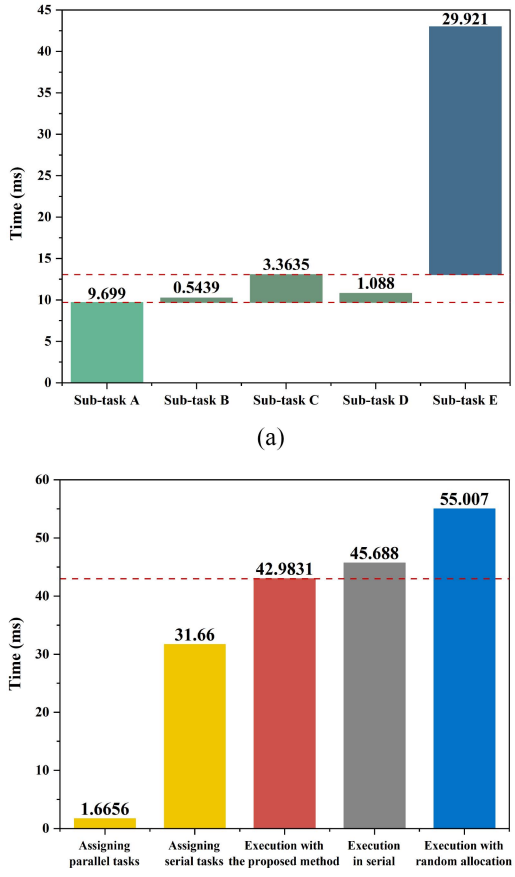


Fig. 5. Task allocation time cost. (a) Time cost of decomposed tasks. (b) Time cost of task allocation and execution.

assigned according to the method shown in Fig. 5(a), the total execution time of the task is 42.9831 ms, which can be seen in the red bar in the figure. If the task is executed serially, the total execution time is 45.688 ms, which can be seen in the gray bar. Additional, if the task is executed with random allocation, the total execution time is 55.007 ms, which can be seen in the blue bar. In general, the task execution with our proposed method is the fastest in these cases.

For the simulation of resource monitoring algorithms, we take the TensorFlow federated model as an example to simulate real-time resource monitoring in the application scenario of machine learning. We deploy computers as computing nodes and perform multiple distributed machine learning tasks on the MNIST data set. The value of ResourceCoin is determined by the key indicators of machine learning, namely, loss and accuracy. The specific formula is

$$\text{ResourceCoin}_i = \frac{A \cdot R_i}{L \cdot t \cdot S}.$$

In the formula above, ResourceCoin_i represents the ResourceCoin that the node i owns. A is the accuracy, and R_i is the number of rounds of machine learning. L is the loss, t is the operation time, and S is the learning rate. The experimental results are shown in Fig. 6.

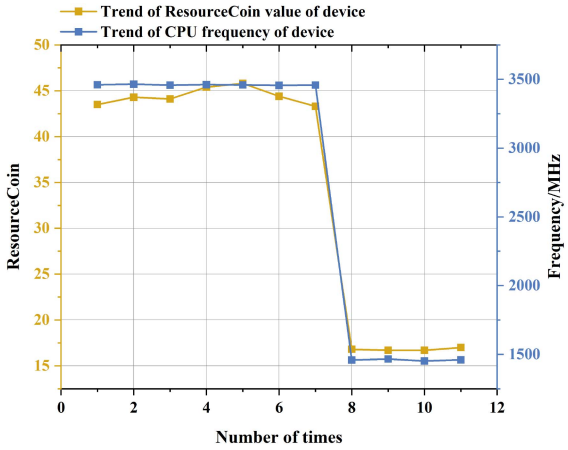


Fig. 6. Trend of resource fluctuation.

The implementation figure above shows the ResourceCoin value and the average frequency of the CPU when the computing node performs distributed machine learning 11 times. The learning rate is set to 0.02 and the number of rounds is 100. During the first seven executions, we did not impose any restrictions on the resources of the device, so that the device was operating at its best. Starting from the 8th execution process, we limit the resources of the device to halve its computing power. It can be seen from the figure that ResourceCoin is at a relatively high level and stable during the first 7 executions. Starting from the 8th time, it experienced a significant decline, and then maintained a stable operation at a low level. This trend is very close to the average frequency of actual CPU operations. The results show that ResourceCoin can fully reflect the actual resources of the devices. Once the device is affected by resource requirements from the tasks outside, the execution resources of the task can be reflected by ResourceCoin through the trend. If the time interval of a single task is set small enough or the tasks are subdivided, so that the execution time of a single task is shorter, and the resource status of the device can be reflected in real time.

VI. Conclusion

In this article, we proposed a blockchain-based IoT resource monitoring and scheduling framework, where the nodes with idle computing power in the P2P network share the computational burden with computing the computational tasks. The task publishers upload their heavy tasks and, later, the computing nodes devote their computational resources for rewards. We took advantage of blockchain for distributed resource scheduling and task management. Meanwhile, we achieved resource condition monitoring in the P2P network, which gives positive feedback to the protocol, and makes resource scheduling more reasonable and system operation more robust. Finally, we experimentally evaluated our proposal on the Hyperledger Fabric platform, and the results have shown the validity of our proposed framework.

References

[1] M. Satyanarayanan, "The emergence of edge computing," *Computer*, vol. 50, no. 1, pp. 30–39, Jan. 2017.

[2] F. Jalali, K. Hinton, R. Ayre, T. Alpcan, and R. S. Tucker, "Fog computing may help to save energy in cloud computing," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 5, pp. 1728–1739, May 2016.

[3] X. Zhou et al., "Edge enabled two-stage scheduling based on deep reinforcement learning for Internet of Everything," *IEEE Internet Things J.*, early access, May 30, 2022, doi: 10.1109/JIOT.2022.3179231.

[4] X. Zhou, X. Yang, J. Ma, and I.-K. Wang, "Energy efficient smart routing based on link correlation mining for wireless edge computing in IoT," *IEEE Internet Things J.*, vol. 9, no. 16, pp. 14988–14997, Aug. 2022.

[5] C. Ding, A. Zhou, X. Liu, X. Ma, and S. Wang, "Resource-aware feature extraction in mobile edge computing," *IEEE Trans. Mobile Comput.*, vol. 21, no. 1, pp. 321–331, Jan. 2022.

[6] Q. Li, S. Wang, A. Zhou, X. Ma, F. Yang, and A. X. Liu, "QoS driven task offloading with statistical guarantee in mobile edge computing," *IEEE Trans. Mobile Comput.*, vol. 21, no. 1, pp. 278–290, Jan. 2022.

[7] M. Qiu, S. Kung, and K. Gai, "Intelligent security and optimization in edge/fog computing," *Future Gener. Comput. Syst.*, vol. 107, pp. 1140–1142, Jun. 2020.

[8] K. Gai, K. Xu, Z. Lu, M. Qiu, and L. Zhu, "Fusion of cognitive wireless networks and edge computing," *IEEE Wireless Commun.*, vol. 26, no. 3, pp. 69–75, Jun. 2019.

[9] K. Gai, M. Qiu, Z. Xiong, and M. Liu, "Privacy-preserving multi-channel communication in edge-of-things," *Future Gener. Comput. Syst.*, vol. 85, pp. 190–200, Aug. 2018.

[10] L. Cui et al., "CREAT: Blockchain-assisted compression algorithm of federated learning for content caching in edge computing," *IEEE Internet Things J.*, vol. 9, no. 16, pp. 14151–14161, Aug. 2022.

[11] L. Han et al., "The impact of task abandonment in crowdsourcing," *IEEE Trans. Knowl. Data Eng.*, vol. 33, no. 5, pp. 2266–2279, May 2021.

[12] F. Ismailoglu, "Aggregating user preferences in group recommender systems: A crowdsourcing approach," *Decis. Support Syst.*, vol. 152, Jan. 2022, Art. no. 113663.

[13] H. Qiu, T. Dong, T. Zhang, J. Lu, G. Memmi, and M. Qiu, "Adversarial attacks against network intrusion detection in IoT systems," *IEEE Internet Things J.*, vol. 8, no. 13, pp. 10327–10335, Jul. 2021.

[14] Z. Shao, C. Xue, Q. Zhuge, M. Qiu, B. Xiao, and E. H.-M. Sha, "Security protection and checking for embedded system integration against buffer overflow attacks via hardware/software," *IEEE Trans. Comput.*, vol. 55, no. 4, pp. 443–453, Apr. 2006.

[15] L. Tan, H. Xiao, K. Yu, M. Aloqaily, and Y. Jararweh, "A blockchain-empowered crowdsourcing system for 5G-enabled smart cities," *Comput. Stand. Interfaces*, vol. 76, Jun. 2021, Art. no. 103517.

[16] C. Li, X. Qu, and Y. Guo, "TFCrowd: A blockchain-based crowdsourcing framework with enhanced trustworthiness and fairness," *EURASIP J. Wireless Commun. Netw.*, vol. 2021, no. 1, p. 168, Feb. 2021. [Online]. Available: https://doi.org/10.1186/s13638-021-02040-z

[17] L. Kong, Z. Wu, G. Chen, M. Qiu, S. Mumtaz, and J. J. P. C. Rodrigues, "Crowdsensing-based cross-operator switch in rail transit systems," *IEEE Trans. Commun.*, vol. 68, no. 12, pp. 7938–7947, Dec. 2020.

[18] X. Zhou, W. Liang, I. Kevin, K. I.-K. Wang, and L. T. Yang, "Deep correlation mining based on hierarchical hybrid networks for heterogeneous big data recommendations," *IEEE Trans. Comput. Social Syst.*, vol. 8, no. 1, pp. 171–178, Feb. 2021.

[19] A. F. Bashir, V. Susarla, and K. Vairavan, "A statistical study of the performance of a task scheduling algorithm," *IEEE Trans. Comput.*, vol. C-32, no. 8, pp. 774–777, Aug. 1983.

[20] L. Cai, X. Wei, C. Xing, X. Zou, G. Zhang, and X. Wang, "Failure-resilient DAG task scheduling in edge computing," *Comput. Netw.*, vol. 198, Oct. 2021, Art. no. 108361.

[21] C. Rottondi, F. Malandrino, A. Bianco, C. F. Chiasserini, and I. Stavrakakis, "Scheduling of emergency tasks for multiservice UAVs in post-disaster scenarios," *Comput. Netw.*, vol. 184, Jan. 2021, Art. no. 107644.

[22] Z. Ning et al., "5G-enabled UAV-to-community offloading: Joint trajectory design and task scheduling," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 11, pp. 3306–3320, Nov. 2021.

[23] F. Lumpp, S. Aldegheri, H. D. Patel, and N. Bombieri, "Task mapping and scheduling for OpenVX applications on heterogeneous multi/many-core architectures," *IEEE Trans. Comput.*, vol. 70, no. 8, pp. 1148–1159, Aug. 2021.

[24] M. Abdel-Basset, R. Mohamed, R. K. Chakrabortty, and M. J. Ryan, "IEGA: An improved elitism-based genetic algorithm for task scheduling problem in fog computing," *Int. J. Intell. Syst.*, vol. 36, no. 9, pp. 4592–4631, 2021.

[25] D. H. Kim, A. Abraham, and J. H. Cho, "A hybrid genetic algorithm and bacterial foraging approach for global optimization," *Inf. Sci.*, vol. 177, no. 18, pp. 3918–3937, 2007.

[26] M. Dorigo, V. Maniezzo, and A. Colorni, "Positive feedback as a search strategy," Dipartimento di Elettronica, Politecnico di Milano, Milan, Italy, Rep. 91-016, 1991.

[27] M. Dorigo and G. Di Caro, "Ant colony optimization: A new meta-heuristic," in *Proc. Congr. Evol. Comput.*, vol. 2, 1999, pp. 1470–1477.

[28] R. R. Delgado, L. G. B. Ruíz, M. P. Cuéllar, and M. C. Pegalajar, "An ant colony optimization approach for symbolic regression using straight line programs. application to energy consumption modelling," *Int. J. Approx. Reason.*, vol. 121, pp. 23–38, Jun. 2020.

[29] H. Baniata, A. Anaqreh, and A. Kertesz, "PF-BTS: A privacy-aware fog-enhanced blockchain-assisted task scheduling," *Inf. Process. Manage.*, vol. 58, no. 1, 2021, Art. no. 102393.

[30] W. Li, S. Cao, K. Hu, J. Cao, and R. Buyya, "Blockchain-enhanced fair task scheduling for cloud-fog-edge coordination environments: Model and algorithm," *Security Commun. Netw.*, vol. 2021, pp. 1–18, Apr. 2021. [Online]. Available: https://doi.org/10.1155/2021/5563312

[31] W. Feng, Z. Yan, L. T. Yang, and Q. Zheng, "Anonymous authentication on trust in blockchain-based mobile crowdsourcing," *IEEE Internet Things J.*, vol. 9, no. 16, pp. 14185–14202, Aug. 2022.

[32] J. Howe, *The Rise of Crowdsourcing*, vol. 14, Wired, San Francisco, CA, USA, Jan. 2006.

[33] M. Li et al., "CrowdBC: A blockchain-based decentralized framework for crowdsourcing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 30, no. 6, pp. 1251–1266, Jun. 2019.

[34] S. Han, Z. Xu, Y. Zeng, and L. Chen, "Fluid: A blockchain based framework for Crowdsourcing," in *Proc. Int. Conf. Manage. Data*, 2019, pp. 1921–1924.

[35] B. Wang, S. Fu, X. Zhang, T. Xie, L. Lyu, and Y. Luo, *Reliable and Privacy-Preserving Task Matching in Blockchain-Based Crowdsourcing*. New York, NY, USA: Assoc. Comput. Mach., 2021, pp. 1879–1888.

[36] C. Lin, D. He, S. Zeadally, N. Kumar, and K.-K. R. Choo, "SecBCS: A secure and privacy-preserving blockchain-based crowdsourcing system," *Sci. China Inf. Sci.*, vol. 63, no. 3, 2020, Art. no. 130102.

[37] 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.*, 2018, pp. 853–865.

[38] M. Qiu, H. Qiu, H. Zhao, M. Liu, and B. Thuraisingham, "Secure data sharing through Untrusted clouds with blockchain-enhanced key management," in *Proc. Conf. SmartBlock*, 2020, pp. 11–16.

[39] X. Wei, H. Guo, X. Wang, X. Wang, and M. Qiu, "Reliable data collection techniques in underwater wireless sensor networks: A survey," *IEEE Commun. Surveys Tuts.*, vol. 24, no. 1, pp. 404–431, 1st Quart., 2021.

[40] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant system: Optimization by a colony of cooperating agents," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 26, no. 1, pp. 29–41, Feb. 1996.

[41] T. Nguyen, T. Nguyen, Q. Vu, T. T. B. Huynh, and B. M. Nguyen, "Multi-objective sparrow search optimization for task scheduling in fog-cloud-blockchain systems," in *Proc. IEEE SCC*, 2021, pp. 450–455.

[42] L. Li, Y. Teng, F. R. Yu, M. Song, and W. Wang, "Blockchain based joint task scheduling and supply-demand configuration for smart manufacturing," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, 2021, pp. 1–6.