Journal of Cloud Computing:
Advances, Systems and Applications

RESEARCH Open Access

A rational delegating computation protocol based on reputation and smart contract

Juan Ma[1,2], Yuling Chen[1*], Ziping Wang[3], Guoxu Liu[3] and Hongliang Zhu[4]

Abstract

The delegating computation has become an irreversible trend, together comes the pressing need for fairness and efficiency issues. To solve this problem, we leverage game theory to propose a smart contract-based solution. First, according to the behavioral preferences of the participants, we design an incentive contract to describe the motivation of the participants. Next, to satisfy the fairness of the rational delegating computation, we propose a rational delegating computation protocol based on reputation and smart contract. More specifically, rational participants are to gain the maximum utility and reach the Nash equilibrium in the protocol. Besides, we design a reputation mechanism with a reputation certificate, which measures the reputation from multiple dimensions. The reputation is used to assure the client's trust in the computing party to improve the efficiency of the protocol. Then, we conduct a comprehensive experiment to evaluate the proposed protocol. The simulation and analysis results show that the proposed protocol solves the complex traditional verification problem. We also conduct a feasibility study that involves implementing the contracts in Solidity and running them on the official Ethereum network. Meanwhile, we prove the fairness and correctness of the protocol.

Keywords: Rational delegating computation, Smart contract, Nash equilibrium, Incentive contract, Reputation mechanism

Introduction

With the rapid development of cloud computing [1–3], more and more attention is paid to data processing and storage capabilities. Some of these devices are computationally weak due to various resource constraints. As a consequence, there are tasks, which potentially could enlarge a device's range of applications, that are beyond its reach. A solution is to delegate computations that are too expensive for one device, to other devices which are more powerful or numerous and connected to the same network [4]. The traditional delegating computation protocols generally assume that participants need to be honest or malicious. Honest participants always follow the rules of protocol during the execution process while malicious participants always violate the rules. A lot of participants are rational in the execution process, rational participants always choose the strategy to maximize their utilities. In addition, owing to the high cost of computation and communication complexity in the verification process of delegating computation, the efficiency of the protocol will be reduced.

Combining the game theory and traditional delegating computation, the rational delegating computation protocol is a part of rational cryptography. From the perspective of the participant's self-interest, the protocol utilizes the utility functions to ensure the correctness and reliability of the calculation results, and the clients do not need to verify the calculation results to improve the efficiency of the protocol. In recent years, many scholars have conducted researches on rational delegating computation. Kupcu et al. [5] designed an incentive mechanism

*Correspondence: ylchen3@gzu.edu.cn
[1] State Key Laboratory of Public Big Data, College of Computer Science and Technology, Guizhou University, Guiyang, China
Full list of author information is available at the end of the article

for rational delegating computation, which motivated the computing party to correctly executed the protocol and effectively prevented malicious computing parties from improper behavior. Zhang et al. [6, 7] combined game theory and safety entropy to design an entropy criterion model for rational secure two-party computation. This model selected the optimal utility function within the range of the safety entropy threshold through the relationship between safety entropy and utility function. Based on the fully homomorphic encryption scheme and a fully homomorphic signature scheme, Li et al. [8] proposed a new cloud data integrity verification scheme and a verifiable commissioned computing scheme, and proved the correctness and safety of these two schemes. Wang et al. [9] set a new incentive function by adding random values to the geometric reward function, which is used as a trade-off between fairness and incentive compatibility. Zhang et al. [10] introduced batch verifiable computation, which enables the simultaneous delegation of multiple functions. The authors constructed batch verifiable computation schemes that effectively reduce the requirement on cloud storage while preserving efficient client verification. But in the process of delegating computation, if the malicious participants do not follow rules in the protocol, then the benefits of honest participants will reduce. Therefore, how to ensure the fairness of the protocol and improve efficiency, which needs to be considered.

Recently, the fairness of delegating computation is one of the hot topics in current research, and the existing researches utilize a trusted third-party (e.g., bank [11], semi-trusted third-party [12, 13], trusted third-party [14, 15]) to overcome these issues. However, in the protocol process, with a third-party, the potential security problems will inevitably occur [16, 17], e.g., unreasonable Nash equilibrium, privacy leakage, and low efficiency. To eliminate the drawbacks, many researchers adopt smart contracts to realize the Peer-to-Peer (P2P) transaction between the clients and the computing parties [18]. Wang et al. [19] proposed an auditable fair payment protocol based on smart contract, which leverages the traceability and auditability of the blockchain to provide an efficient method for the assets in the entire transaction. Chen et al. [20] proposed a fair data exchange scheme based on blockchain, which guarantees transaction fairness and privacy protection when there is no trusted third-party, and realizes automatic exchange efficiently. Zhou et al. [21] combined the game theory with traditional delegating computation, established a game model with the reputation mechanism, and proposed a three-party game rational delegating computation protocol based on smart contracts. Dong et al. [22] combined game theory and smart contract to design a reasonable prisoner contract, collusion contract, and betrayal contract. In the contract, smart contracts replaced the trusted third-party to ensure the fairness of the payment process. Subsequently, to reduce the delegation computing overhead. Chen et al. [23] proposed a novel incentive-compatible rational delegation computing scheme. The lowest delegation overhead achieved by this scheme is only $n/(2n-2)$ of that achieved by Dong's scheme [22], where n means the number of servers. Besides, many scholars have studied the application of game theory in smart contracts [24–26], and discussed the security of data and models in detail [27]. However, in the process of delegation computation, different tasks or utilities have different impacts on the strategies of the computing parties, leading the client to get a different result. Consequently, it is an urgent task to select a reliable computing party by mining the difference in the preference of the computing party.

In the process of delegating computation, the reputation mechanism is designed to improve the reputation of honest participants and reduce the reputation of malicious participants. The reputation mechanism will reduce the probability of malicious participants being selected [28, 29]. Xiao et al. [30] designed a security system based on the behavior strategy of social norm and reputation system to motivate rational nodes, which motivated rational nodes to give up malicious behaviors for their interests. Zhao et al. [31] realized the confidentiality and reliability of data based on blockchain technology and reputation model. Wang et al. [32] proposed the impact of social cloud reputation and structure on rational computation, which ensures that a party with a good reputation means that they are likely to cooperate with others. The structure of the social cloud is not static. Instead, it evolutes when parties complete one round of computation. Jiang et al. [33] proposed a rational delegating computation based on reputation and contract theory, which ensures that the client selects a reliable computing party. Li et al. [34] proposed a blockchain-based trust mechanism for distributed IoT devices. In this mechanism, the trust level is quantified by regulating trust and risk degree. To solve the sparsity problem of social relationships and the additional risk of exposing the user's privacy in the current social network, Kou et al. [35] proposed a new link prediction method based on the Simhash technology. Li et al. [36] proposed a fair payment protocol based on bitcoin time commitment, which ensures the fairness of participants' payment by using bitcoin time commitment technology. However, in the process of selecting the computing party, the reputation of the computing party will be updated every round. Therefore, how to efficiently view the latest reputation of the computing party is a key issue that needs to be resolved.

To solve the aforementioned problems, we propose a rational delegating computation protocol based on reputation and smart contract, which realizes the optimal utility of all rational participants, and guarantees the correctness of the results and the fairness of the payment process. We analyze the strategies of participants and design a reasonable utility function. Then, we design incentive contracts to describe the motivations of participants, and construct a game tree to facilitate analysis of participants' behavior and utility. Besides, we design a reputation mechanism that has a reputation certificate identification and allows reputation to be measured from multiple dimensions. More specifically, the reputation mechanism quantifies the reputation of the computing party from different dimensions, and uses reputation to ensure the trust of the client in the computing party. Meanwhile, the reputation mechanism has a reputation certificate identification, and the reputation certificate is used to improve the efficiency of the computing party to view historical reputation. We design a reputation mechanism to allow client to choose high-quality computing parties and reduce the possibility of choosing malicious participants. Our main contributions are as follows.

1. According to the behavioral preferences of the participants, we define a utility function and design an incentive contract to motivate the participants to choose the strategy honestly, which reaches a reasonable Nash equilibrium result and ensures that honest participants obtain the maximum utility in the protocol.

2. Based on the smart contract, we propose a rational delegating computation protocol, which realizes the fairness of rational delegating computation. And we design a reputation mechanism for the client to choose high-quality computing parties, which can measure the computing party's reputation from different dimensions and improve the efficiency of the protocol.

3. We conduct a comprehensive experiment to evaluate the proposed protocol. The simulation results and analysis results show that the proposed protocol solves the complex traditional verification problem. In addition, we analyze our entire scheme and conclude that the overhead of the smart contract we design is extremely small. In a nutshell, our figures show that the total transaction cost for executing each contract is below $0.2.

The rest of this paper is organized as follows. In "Preliminaries" section, we introduce concepts such as delegation of computation, game theory, Nash equilibrium, and smart contract. "Incentive contract" section proposes an incentive contract based on a smart contract. "The proposed reputation scheme" section establishes a reputation mechanism that is convenient for the client to choose the computing party. "Protocol proof" section proves the fairness and correctness of the protocol. In "Simulations and results" section, we calculate the cost of the contract and analyze the performance of the protocol. Finally, we draw the conclusion of this work and discuss the future work in "Conclusion" section.

Preliminaries

In this section, we introduce the basic knowledge needed for our scheme. Firstly, we introduce the concept of the delegation of computation and standard game. Furthermore, we review the most important concept in game theory, i.e., the Nash equilibrium. Finally, we introduce some notations and a system model.

Delegation computation

Definition 1 (Delegation Computation). Delegation computation is also called verifiable computation, which refers to that the client C_d sends the delegation task X and function $f(\cdot)$ to the computing parties C, which receives it and uses its resources to calculate $f(X)$, and returns the result $Y = f(X)$ to C_d. At the same time, the calculation results obtained are verifiable. This verification process is much more efficient than the local calculation. Otherwise, it loses the sense of delegation of computation [37].

Game theory

Definition 2 (Standard Game). The standard form of a n-player game is composed of three elements: player set P, strategy space S and utility function u, denoted as $G = \{P, S, u\}$, where $P = \{P_1, \cdots, P_n\}$, $S = \{S_1, \cdots, S_n\}$, $u = \{u_1, \cdots, u_n\}$. Any specific strategy $s_i \in S_i$ indicates that strategy s_i is the key element of the strategy set S_i, and utility function $u : S \rightarrow R$ denotes the profits of the players i under different strategy profiles [38].

Definition 3 (Nash Equilibrium). A strategy profile $s^* = \{s_1^*, \cdots, s_n^*\}$ is a Nash equilibrium of game $G = \{P, S, u\}$, if $u_i(s_i^*, s_{-i}^*) \geq u_j(s_j^*, s_{-j}^*)$ holds for each player $P_i (i = 1, \cdots, n)$ and all $s_j \in S_i$. Obviously, if player $i \neq j$ complies with the strategy s_i^*, then the player will not deviate from the strategy s_j^*, as it will not benefit at all. In principle, there may be multiple Nash equilibrium in a game [38].

Smart contract

Definition 4 (Smart Contract). Cryptocurrencies are a type of digital currency used for decentralized network transactions. Cryptocurrencies are based on a new type of blockchain technology, which enables blockchain transactions to be conducted through the Internet without a trusted third party. With the development of the blockchain, the first decentralized cryptocurrency that emerged was Bitcoin, which began to be used for transactions on the blockchain. The rise of the blockchain has caused the value of Bitcoin to continue to rise. Later, the development of Ethereum appeared, using Ether for transactions, and also proposed smart contract to ensure the fairness of transactions [39].

Smart contracts are in the blockchain environment, allowing the definition and execution of contracts signed

Table 1 Notations

Symbol	Description
C	The set of computing party
C_h	The computing party that returns the correct result
C_f	The computing party that returns the wrong result
C_r	The computing party that returns the random result
C_d	The client
TTP	The trusted third-party
X	The calculation task

on the blockchain. It is the automated execution of the contract, and its essence is a piece of code written on the blockchain. The smart contract is the basis of the program-ability of blockchain, and each node does not rely on a third-party to automatically execute the contract. Broadly speaking, a smart contract is a set of rules encoded in a programming language. Once the execution requirements of the code are met, the script will be automatically executed to realize the operation, and this process does not require the participation of a trusted third-party [40].

Ethereum is an open-source public blockchain platform with smart contract functions. It provides a decentralized Ethereum virtual machine through its dedicated cryptocurrency Ether to process peer-to-peer contracts. In Ethereum, a transaction refers to sending a transaction from one account to another. Each transaction includes the sender's signature, the receiver's signature, and the amount of money sent. When users send a transaction, they need to pay a certain transaction fee (gas) for the execution of this transaction. The purpose is to prevent users from sending too many meaningless transactions on the blockchain. When the execution of a transaction requires complex and tedious calculation steps, the more gas the transaction consumes, after the transaction is executed, if the gas paid is not consumed, it will be returned to the account of the transaction initiator [41].

System model

In this part, we give a system model to describe our scheme. The system model considered in our construction comprises one client denoted by C_d and multiple computing parties denoted by $\{C_1, C_2, \cdots, C_n\}$. However, not all computing parties participate in the calculation, and the client selects two computing parties with high reputation to participate in the task based on the reputation. Before describing the system model in detail, we describe the parameters and concepts required in this solution. Some notions are explained in Table 1.

As illustrated in Fig. 1, the system model consists of eight steps. First, the client broadcasts the calculation task to the computing party, and includes the return time of the calculation task and the amount paid. The computing party views the calculation task, and the interested computing party will respond to the request. Next, the client will select two computing parties with a high reputation based on the reputation to perform the task, In short, the reputation (reputation evaluation scheme is given in The proposed reputation scheme) is the basis for the computing party to obtain the task, and the rational computing party must perform the task with an honest strategy to improve its reputation if it wants to obtain the task. Then, the client chooses the computing party and

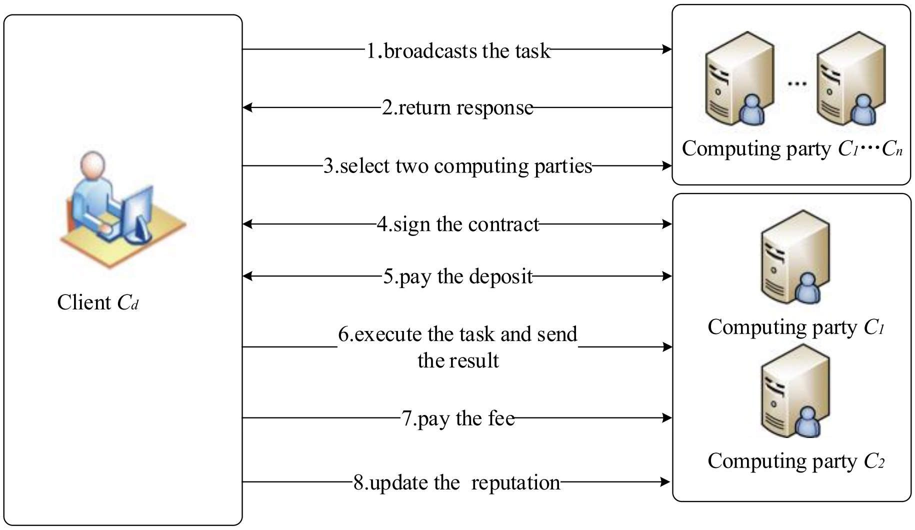

Fig. 1 The system model

signs a contract with them. According to the content of the signed contract (incentive contract design is given in Incentive contract), the client and the computing party respectively deposit the deposit stipulated in the contract into the smart contract. In other words, each participant must pay a deposit to join the contract, otherwise, the contract will be terminated if the deposit is not paid. Additionally, participants receive deposit rewards for being honest. Conversely, the deposit will be confiscated if participants behave dishonestly. What's more, depositing a deposit provides a guarantee for the following calculations, and then the computing party performs the task and sends the calculation result to the client. The client determines whether the computing party has performed the task correctly by cross-validation, and makes the corresponding response and payment. Finally, according to the interactive behavior of the computing party, the client updates and uploads the reputation of the computing party.

Incentive contract

In this section, we first give the deposit variables in the incentive contract. Then, we design the detailed process of delegating computation based on smart contracts and propose an incentive contract. Finally, we conduct a detailed analysis of the content of the incentive contract.

Deposit variables

We define some deposit variables required in the incentive contract in Table 2, and these deposit variables are all non-negative.

The following relations are obvious.

(1) $g - v > 0$. The computing party does not accept under-paid jobs. The calculation cost that the computing party needs to spend is c, and the computing party does not accept a job that is lower than the calculation cost.

(2) $r - 2c > 0$. The cost of calling the TTP by the client is higher than the cost paid to the two computing parties. Otherwise, the client will not choose the computing party to calculate the delegated task, and the client will choose to use TTP for calculation.

Table 2 Deposit variables

Value	Description
g	The client agrees to pay to a computing party for computing the task
v	The computing party's cost for computing the task
r	The deposit to invoke the *TTP*
c	The deposit a computing party pays to the client in order to get the task
$2g + r$	The deposit of a client in the smart contract

Contract content

Due to the characteristics of rational participants maximizing their utility, we analyze the strategies of rational participants and define the utility function. We introduce the specific content of the incentive contract, and we analyze each step in detail by studying the utility function of rational players. The content and steps of the contract are as follows.

Contract content

*Step1(Publish task): The client C_d publishes the task.

*Step1.1: C_d broadcasts the task X to all computing parties.

*Step1.2: If C decides to accept task will return response request P.

*Step2: C_d selects $C_i (i \in \{1, 2\})$ based on the reputation.

*Step3(Sign contract): C_d, C_1 and C_2 must sign the *contract* to start it, otherwise *contract* terminates.

*Step4(Pay deposit): C_d, C_1 and C_2 pay the deposit $(2g + r, c, c)$; otherwise *contract* terminates.

*Step5(Verification the result): The client checks the results returned by the computing party.

*Step5.1: If both C_1, C_2 deliver the results within the specified time, and the results are equal. Transferring C_d's deposit g to each C_i, returning C_i's deposit c.

*Step5.2: Otherwise, the TTP is invoked and one of the following steps is performed.

*Step5.2.1: If both C_1, C_2 are one party honest (return the correct result C_h) and one party malicious (return the wrong result C_f or return the random result C_r). Transferring C_d's deposit g to each C_h and C_f's (or C_r's) deposit c to C_d, and pays fee r to TTP.

*Step5.2.2: If both C_1, C_2 return wrong result. Transferring deposit c to C_d, returning C_d's deposit $2g$, and pays fee r to TTP.

*Step5.2.3: If both C_1, C_2 return random result. Transferring C_i deposit c to C_d, returning C_d's deposit $2g$, and pays fee r to TTP.

*Step5.3: For each C_i, if C_i fail to deliver the result within the specified time. Transferring deposit c to C_d, returning C_d's deposit $2g$.

*Step6: If the C_d deviates from the protocol, transferring C_d's deposit g to each C_i, returning each C_i's deposit c.

Contract analysis

In the protocol, if the participants meet the terms of the contract, they continue to execute the contract. Otherwise, the contract terminates. We analyze the behavioral strategies of the participants in the contract. Our contract analysis process is as follows.

Contract analysis

*Step1 means that C_d sends the tasks to C.

*Step2 means that C_d selects two computing parties $C_i (i \in \{1, 2\})$ based on the reputation.

*Step3 means that C_d, C_1 and C_2 must sign the contract to start it, otherwise *contract* terminates.

*Step4 means that C_d, C_1 and C_2 respectively pay deposit to the smart contract.

*Step4.1 C_d, C_1 and C_2 respectively pay deposit to the smart contract $(2g + r, c, c)$ and enter $\ast Step5$.

*Step4.2 otherwise *contract* terminates and the utility is $u(C_1, C_2) = (0, 0)$.

*Step5 represents the verification process of the client.

*Step5.1 means that C_1 and C_2 are honest, the utility is $u(C_1, C_2) = (g - v, g - v)$.

*Step5.2 means that one is honest and returns a wrong value, the utility is $u(C_h, C_f) = (g + c - v, -c - v)$.

*Step5.3 means that one is honest and one returns a random value, the utility is $u(C_h, C_r) = (g + c - v, -c)$.

*Step5.4 means that both C_1 and C_2 return random value, the utility is $u(C_r, C_r) = (-c, -c)$.

*Step5.5 means that both C_1 and C_2 return wrong result or time out, the utility is $u(C_f, C_f) = (-v - c, -v - c)$.

*Step6 means that C_d deviates from the protocol, the smart contract confiscates the C_d deposit $2g$ and transfers deposit g to C_1 and C_2.

Game tree

From the previous analysis, the computing party has different strategy choices in the calculation process, and different strategy choices will result in different utility functions. For this reason, we construct the game tree for the computing party, and the game tree is shown in Fig. 2.

We analyze the payment game model in the delegating computation process from the perspective of game theory. C_1 and C_2 represent two computing parties, which have three behavior strategies, namely $\{honest, dishonest, random\ value\}$. The *honest* behavior strategy means that the computing party honestly performs the calculation task and returns the result to the client within the specified time; *dishonest* behavior strategy indicates that the computing party maliciously performs the calculation task and returns to the client or fails to return the calculation result to the client within the specified time; *random value* behavior strategy means that the computing party returns a random value to the client without calculation. More specifically, the computing party chooses different strategies to obtain different utilities. For instance, if both C_1 and C_2 choose the honest strategy, C_1's utility is $u(g - v)$, and C_2's utility is $u(g - v)$. However, if C_1 chooses the honest strategy and C_2 chooses the dishonest strategy, C_1's utility is $u(g + c - v)$, and C_2's utility is $u(-c - v)$. In short, there are nine such situations (see Fig. 2 for details). The red line in the game tree represents the Nash equilibrium path. When C_1 and C_2 are honest behavior strategies, it reaches node n_6, that is, the game reaches the Nash equilibrium. The gray node represents the utility function of C_1 and C_2 to reach the Nash equilibrium. uc_1 represents the C_1 utility function, and uc_2 represents the C_2 utility function.

The proposed reputation scheme

In this part, the reputation mechanism mainly includes five stages. First, we define the format of the reputation certificate. Next, the client selects the appropriate computing party to compute the task according to the reputation. Then, the client evaluates the current reputation based on the computing party's honesty, and the client computes the global reputation. Finally, the computing party updates the reputation certificate.

Reputation certificate

In the delegating computation process, the client selects the appropriate computing party according to its task requirements. In a reputation mechanism with a reputation certificate, the computing party has a high-quality reputation as the basis for obtaining the task. Generally, the reputation of the computing party is higher, the probability of being selected is greater. The reputation certificate format is shown in Table 3.

Search of historical reputation

At this stage, the client publishes a task to the computing party, and the interested computing party returns a response request. At the same time, the computing party generates the latest reputation certificate to obtain the task. Next, for the response requests returned by many computing parties, the client will select two computing parties with high reputations to perform the task based on the reputation. In other words, if the computing party wants to obtain a task, the computing party must perform the task honestly to improve its reputation. The format of the new reputation certificate is shown in formula 1.

$$CRC_{C,n+1} = IDc||R_{n+1}||HRV_n||Sig_n(IDc||R_{n+1}||HRV_n) \tag{1}$$

Evaluation of current reputation

Usually, the computing party will choose different strategies according to different tasks and utilities, resulting in

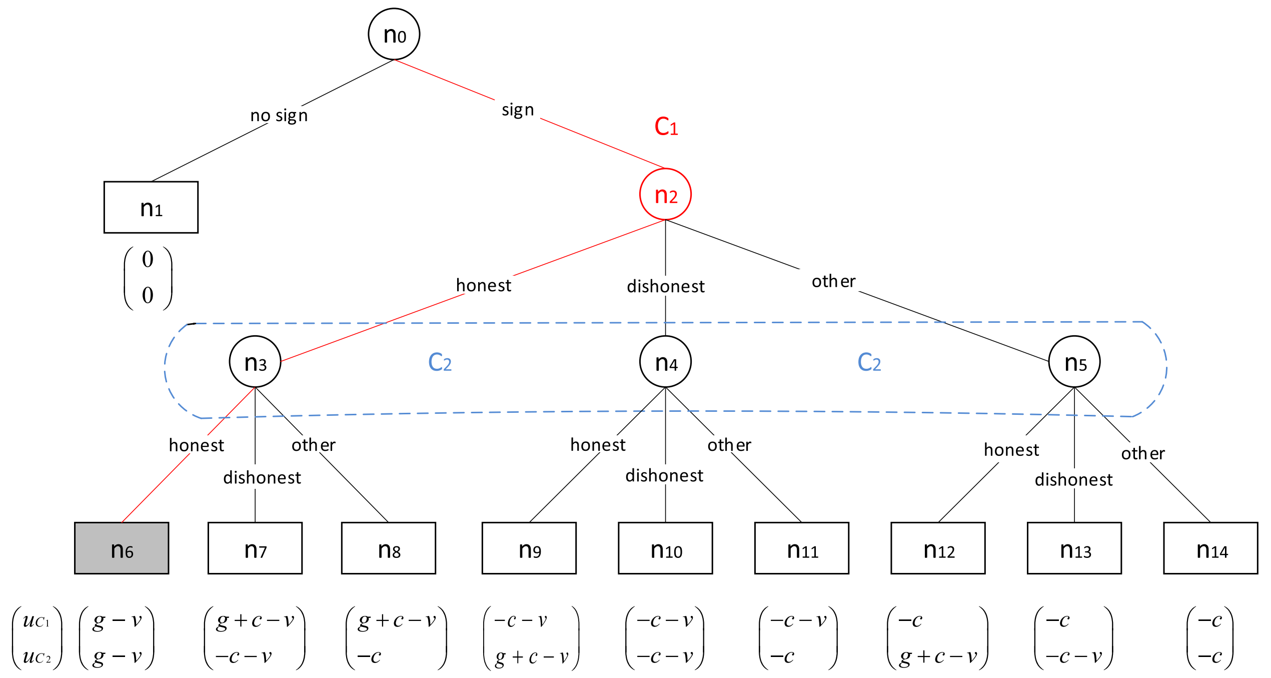
n0
no sign
sign
C1
n1
$\begin{pmatrix}0\\0\end{pmatrix}$
n2
honest
dishonest
other
C2
n3
n4
C2
n5
honest
dishonest
other
honest
dishonest
other
honest
dishonest
other
n6
n7
n8
n9
n10
n11
n12
n13
n14
$\begin{pmatrix}u_{C_1}\\u_{C_2}\end{pmatrix}$
$\begin{pmatrix}g-v\\g-v\end{pmatrix}$
$\begin{pmatrix}g+c-v\\-c-v\end{pmatrix}$
$\begin{pmatrix}g+c-v\\-c\end{pmatrix}$
$\begin{pmatrix}-c-v\\g+c-v\end{pmatrix}$
$\begin{pmatrix}-c-v\\-c-v\end{pmatrix}$
$\begin{pmatrix}-c-v\\-c\end{pmatrix}$
$\begin{pmatrix}-c\\g+c-v\end{pmatrix}$
$\begin{pmatrix}-c\\-c-v\end{pmatrix}$
$\begin{pmatrix}-c\\-c\end{pmatrix}$

Fig. 2 The game tree

them having different reputations. Reputation is whether the computing party's behavior is honest or not after each round of tasks. In general, the computing party has two reputation statuses: honest or malicious. Specifically, when the computing party's reputation in round $t+1$ is higher than that in round t, i.e. $L_{rep_{t+1}} \geq L_{rep_t}$, we consider the computing party to be honest. On the contrary, when the computing party's reputation in round $t+1$ is lower than round t, i.e. $L_{rep_{t+1}} \leq L_{rep_t}$, we consider the computing party to be malicious. The evaluation of reputation is shown in formula 2 below.

$$L_{rep_{t+1}} = \alpha_{i,t}\left[\frac{H}{S} * D(r) * A(r) * T(r)\right] \tag{2}$$

Let $\alpha_{i,t}$ be the client C_d interactive evaluation to the computing party C_i in round t, here, $\alpha_{i,t} \in \{0, 1, -1\}$. In order to reward the computing party who performed well in the execution of the task and punish the computing party who failed to complete the task as required, we propose formula 3 as follows.

$$\begin{cases} \alpha_{i,t} = \;\;\, 1, \; L_{rep_{i,t}} = \frac{H}{S} * D(r) * A(r) * T(r) \\ \alpha_{i,t} = \;\;\, 0, \; L_{rep_{i,t}} = 0 \\ \alpha_{i,t} = -1, \; L_{rep_{i,t}} = -\left[\frac{H}{S} * D(r) * A(r) * T(r)\right] \end{cases} \tag{3}$$

If and only if there is no any interaction between the client C_d and the computing party C_i, $L_{rep_{i,t}} = 0$. Obviously, for any C_i, when $t = 0$, $\alpha_{i,t} = 0$, then $L_{rep_{i,t}} = 0$. If and only if the computing party C_i honestly performs the delegation task in round t, $\alpha_{i,t} = 1$, then $L_{rep_{i,t}} = \frac{H}{S} * D(r) * A(r) * T(r)$. Conversely, if and only if the computing party C_i is behaves dishonestly in round t, $\alpha_{i,t} = -1$, then $L_{rep_{i,t}} = -\left[\frac{H}{S} * D(r) * A(r) * T(r)\right]$.

Where, $L_{rep_{i,t}}$ satisfies $-1 \leq L_{rep_{i,t}} \leq 1$, $L_{rep_{i,t}}$ is the reputation of the current round, H represents the number of honest calculations by the computing party, S represents the total number of calculations by the computing party, $D(r)$ represents the complexity coefficient of the delegating computation and satisfies $0 \leq D(r) \leq 1$, $A(r)$ represents the benefit coefficient of the delegating computation and satisfies $0 \leq A(r) \leq 1$, $T(r)$ represents the time to submit the result of the calculation.

(1) When $L_{rep_{i,t}} = -1$, it means that C_i is completely malicious in the process of delegating computation of round t.

(2) When $L_{rep_{i,t}} = 0$, it means that there is no interaction between C_d and C_i in the process of delegating computation of round t.

Table 3 Reputation certificate

Symbol	Description
ID_{cd}	ID of the client
ID_c	ID of the computing party
R_n	Number of rounds
Sig_{cd}	Signature of the client
Sig_c	Signature of the computing party
HRV_c	The historical reputation of the computing party

(3) When $L_{rep_{i,t}} = 1$, it means that C_i is completely honest in the delegating computation of round t.

Computation of global reputation

In this stage, after the client evaluates the reputation of the current round, the client computes the global reputation according to the historical reputation and the reputation of the current round. The computation of global reputation is shown in formula 4.

$$G_{rep_{n+1}} = G_{rep_n} + L_{rep_{n+1}} \tag{4}$$

$G_{rep_{n+1}}$ represents the global reputation of the computing party, and G_{rep_n} represents the global reputation of the computing party in the last round.

Update of reputation certificate

Finally, the client signs the global reputation of the computing party and uploads it to the blockchain. Then, the computing party updates the content of its reputation certificate, which facilitates the generation of the latest reputation certificate for the next round of tasks. The format of the latest reputation certificate is shown in formula 5 below.

$$CRC_{C,n+1} = CRC_{C-C_d,n+1} || Sig_n(CRC_{C-C_d,n+1}) \tag{5}$$

Protocol proof

In this section, we prove the protocol in detail from the two aspects of fairness and correctness of the protocol. More details are given in "Fairness proof" section and "Correctness proof" section.

Fairness proof

Theorem 1. *The rational delegating computation protocol based on smart contract is fair.*

Proof of Theorem 1. To ensure the fairness of the protocol, at the initial stage of the protocol, the client and the computing party respectively pay a deposit of $(2g+r, c)$ to the smart contract. In the payment process of delegating computation, there are four cases as follows:

Case 1: When the computing party is honest, the smart contract transfers the deposit g to the computing party's account.

Case 2: When the computing party is malicious, the smart contract confiscates the malicious computing party's deposit c and transfers deposit c to the client's account.

Case 3: When the client is malicious, the smart contract confiscates the deposit $2g$ and transfers deposit g to the two computing parties' account respectively.

Case 4: When the client is honest, the client can get the correct calculation result.

In order to avoid the situation that the third-party is dishonest or bought off, based on the smart contract technology, we construct a rational delegating computation protocol to achieve fairness.

Correctness proof

Theorem 2. *The rational delegating computation protocol based on smart contract is correct.*

Proof of Theorem 2. There are three cases for the proof as following:

(1) When C_1 chooses the *honest* strategy, if C_2 chooses the *honest* strategy, the utility is $u(g-v)$; else if C_2 chooses the *random value* strategy, the utility is $u(-c)$; else C_2 chooses the *dishonest* strategy, the utility is $u(-c-v)$; which is $u(g-v) > u(-c) > u(-c-v)$.

(2) When C_1 chooses the *dishonest* strategy, if C_2 chooses the *honest* strategy, the utility is $u(g+c-v)$; else if C_2 chooses the *random value* strategy, the utility is $u(-c)$; else C_2 chooses the *dishonest* strategy, the utility is $u(-c-v)$; which is $u(g+c-v) > u(-c) > u(-c-v)$.

(3) When C_1 chooses the *random value* strategy, if C_2 chooses the *honest* strategy, the utility is $u(g+c-v)$; else if C_2 chooses the *random value* strategy, the utility is $u(-c)$; else C_2 chooses the *dishonest* strategy, the utility is $u(-c-v)$; which is $u(g+c-v) > u(-c) > u(-c-v)$.

Similarly, C_2 also has the above three cases. According to the protocol analysis, only when participants choose the *honest* strategy, maximizing their utility and reach Nash equilibrium.

Simulations and results

In this section, we utilize the MATLAB software to simulate the computing party's reputation changes and the time cost of delegating computation. And, we simulate the smart contract on the Ethereum, then we analyze the overhead of the entire scheme and analyze the performance of the protocol.

Experiments

In the simulation process of computing party reputation, we assume that there are three computing parties (i.e., C_1, C_2, and C') and set their initial reputation to 60 points. As can be seen from Fig. 3, C' does not participate in the calculation, so the initial reputation is maintained. In rounds 1-4, when C_1 and C_2 choose the honest strategy to the same degree, their reputation increases at a similar rate. In the fifth round, with the increase of difficulty and interest, C_1 continues to be honest and C_2 chooses to cheat. In consequence, C_1's reputation increases while C_2's reputation decreases. In the later rounds, C_1 and C_1 are both honest, so their reputation keeps growing.

The experimental result reveals that the reputation change in Fig. 3. If the computing party's strategy choice is honest, the computing party's reputation will always keep

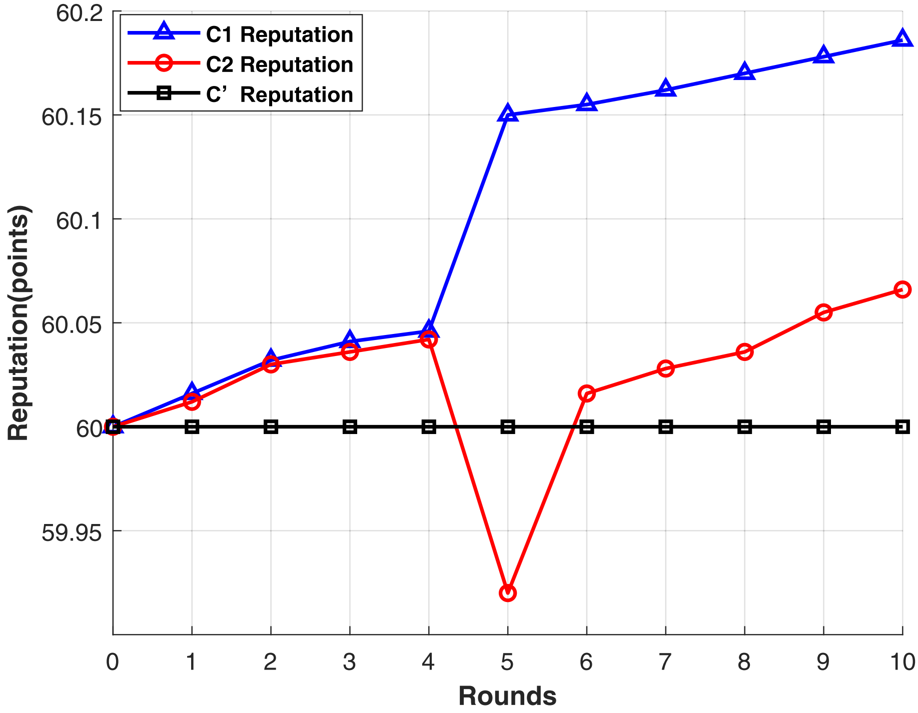

Fig. 3 Computing party's reputation changes

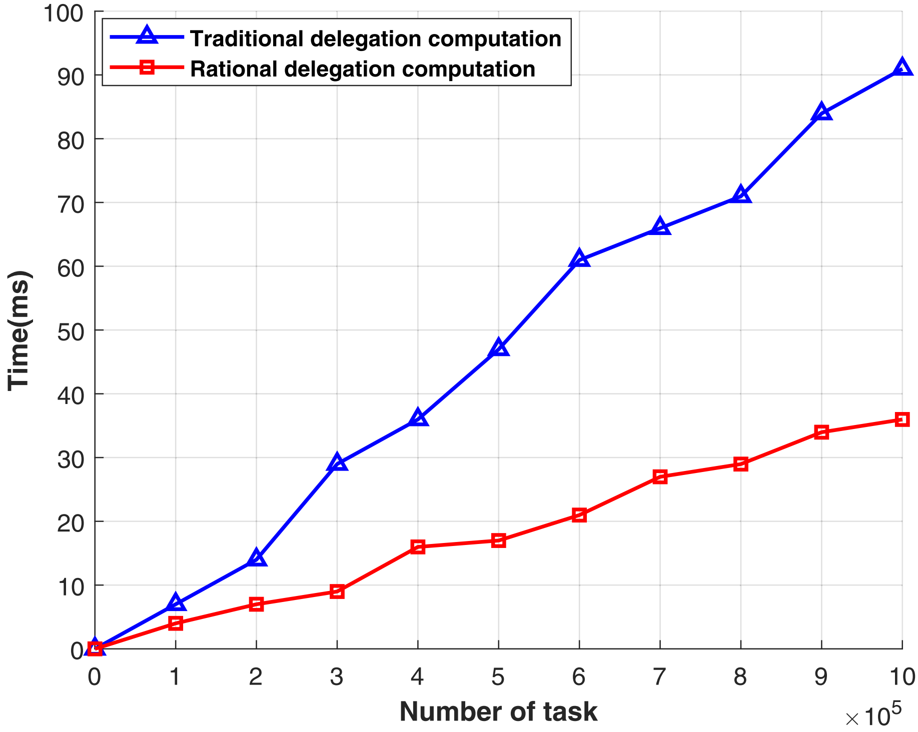

Fig. 4 Time cost of delegating computation

Table 4 Cost of using the smart contracts

Serial	Functions	Cost in Gas	Cost in USD
1	Initialize	410043	0.111
2	Create	10666	0.003
3	Select	384167	0.104
4	Sign	198988	0.054
5	Return	93084	0.025
6	Check	90084	0.024
7	Pay	295880	0.078

rising steadily. Conversely, if the computing party's strategy choice is malicious, the computing party's reputation will continue to decline, and this will affect the client's distrust of the computing party in the process of rational delegating computation.

We compare the time overhead required for different numbers of logarithmic operations using traditional delegating computation and our protocol. In the traditional delegating computation protocol, the client needs to verify the returned result, so it will consume a lot of verification time. However, the client does not need to verify the results in our proposed protocol. As shown in Fig. 4, we find that the rational delegating computation consumes less time than traditional delegating computation. Meanwhile, with the increase of the number of delegates, the gap between them increases and the computation efficiency of the rational delegating computation is higher than before.

Overhead

We show the cost of setting up and executing the contracts on the offical Ethereum network. Cost is measured in terms of gas in the smart contract. The gas price is 1 Gas = 1 Gwei (1×10^{-9} ether) in all transactions, where the current exchange rate is 1 ether = 270.16 USD. We show the cost of executing the functions in Table 4.

As we can see, the financial cost for using the smart contracts on the Ethereum network is low. The initialization ($0.111) is used to initialize the contract, we find other functions cost less. Our figures show that the total transaction cost for executing each contract is below $0.2.

Performance analysis

Table 5 shows the comparison of the rational delegating computation protocol in the proposed scheme and the existing delegating computation protocol. Comparing from the computational complexity, communication complexity, privacy, fairness of client, and fairness of computing party. Here, "✓" satisfies the performance and "X" dissatisfies the performance.

Xu et al. [12] proposed to use an honest but curious third-party to help verify the task results, which ensures the fairness of the client. Its computational complexity is $O(n)$ and communication complexity is $O(n)$ (n is the length of the result). In fact, the scheme uses a third-party to verify the calculation results, which is easy to leak the privacy of participants, and it cannot guarantee that the computing party can get the service fee, so the fairness of the computing party cannot be guaranteed.

Huang et al. [13] constructed a fair payment protocol based on bitcoin and commitment sampling technology. Its computational complexity is $O(1)$ and communication complexity is $O(1)$. The scheme introduced a semi-trusted third-party to help the client get back the deposit, but it is easy to leak the privacy of participants.

Yin et al. [15] designed a rational delegating computation protocol based on Micali-Rabin's random vector representation technique. Its computational complexity is $O(1)$ and communication complexity is $O(1)$. The protocol guarantees fairness in the form of deposits deposited by participants. However, there is a trusted third party in the protocol, which easily reveals the privacy of participants.

We propose a rational delegating computation protocol based on smart contract, the computational complexity is $O(1)$ and the communication complexity is $O(1)$. This protocol is based on smart contract to realize the fairness of delegating computation and protect the privacy of participants. We use the utility function to constrain the participants to execute the strategy honestly and ensure the correctness and reliability of the calculation results. In addition, we design a reputation mechanism that measures reputation from multiple dimensions. This reputation mechanism provides reputation certificate identification, which improves the communication efficiency of the protocol.

Conclusion

Combining game theory with smart contract, in this paper, we propose a rational delegating computation protocol based on reputation and smart contract. More

Table 5 The comparison of protocol

Literature	Computational complexity	Communication complexity	Privacy	Client's fairness	Computing party's fairness
Xu[12]	$O(n)$	$O(n)$	X	✓	X
Huang[13]	$O(1)$	$O(1)$	X	✓	✓
Yin[15]	$O(1)$	$O(1)$	X	✓	✓
Our protocol	$O(1)$	$O(1)$	✓	✓	✓

specifically, we analyze the strategies of participants, then we utilize smart contracts to substitute the third-party to ensure the fairness of protocol. What's more, we define a utility function and design an incentive contract to motivate the participants to choose the strategy honestly, which reaches a reasonable Nash equilibrium result. In other words, each participant must pay a deposit to join the contract, and participants will be rewarded for being honest. Conversely, the deposit will be confiscated if participants behave dishonestly. Also, the simulations show, it is feasible to use smart contracts to implement the incentive mechanisms, and the total cost of using smart contracts is extremely low. In addition, we design a reputation scheme, which can measure the reputation from different dimensions to ensure the client chooses a reliable computing party.

In this paper, we use a cross-validation method to verify the calculation results returned by the computing party, which causes the client to pay the calculation fee to the two computing parties, which is very expensive for the client. One future direction would be to combine the time prediction [42] with the measurement of the participant's behavioral uncertainty [43], to verify the computing party with a high reputation with a low verification probability, and reduce the computation cost to a computing party. Another future direction would be to extend our scheme to the rational delegation of computation like [23]. Our work will focus on reducing the cost of the computing party. We will try to reduce the delegation cost to a smaller amount.

Acknowledgements
We are thankful to State Key Laboratory of Public Big Data of Guizhou University for providing an environment for editing manuscripts and experiments.

Authors' contributions
Visualization, Y.C.; Conceptualization, Y.C. and Z.W.; Methodology, G.L.; Supervision, H.Z. All authors have read and agreed to the published version of the manuscript.

Authors' information
Juan Ma is currently a graduate student with the College of Computer Science and Technology, Guizhou University. Her research interests include data security and privacy protection, game theory and information security, and delegating computation.
Yuling Chen received the B.S. degree from Taishan University, Tai'an, China, in 2006, and the M.S. degree from Guizhou University, Guiyang, China, in 2009. She is currently an Associate Professor with the Guizhou Provincial Key Laboratory of Public Big Data, Guizhou University. Her recent research interests include cryptography and information security.
Ziping Wang is a professor and currently works at Blockchain Laboratory of Agricultural Vegetables, Weifang University of Science and Technology, Weifang, Shandong, China. His research interests include image processing, blockchain, and machine learning.
Guoxu Liu received the M.S. degree in electrical and electronics engineering from the Nanyang Technological University, Singapore, in 2014 and a Ph.D. degree in electronics engineering from the Pusan National University, Korea, in 2020. He is currently an assistant professor in the Weifang Key Laboratory of Blockchain on Agricultural Vegetables at Weifang University of Science and Technology. His research interests include image processing, computer vision, and machine learning.
Hongliang Zhu. He received his Ph. D. degree in information security from Beijing University of Posts and Telecommunications (BUPT). He is an associate professor in BUPT, meanwhile, serves as vice director for Beijing Engineering Lab for Cloud Security and Information Security Center of BUPT. His current research interests include network security and big data security.

Funding
This study is supported by Foundation of National Natural Science Foundation of China (Grant Number: 61962009); Major Scientific and Technological Special Project of Guizhou Province (20183001, 20193003); Foundation of Guizhou Provincial Key Laboratory of Public Big Data (No.2018BDKFJJ008); Foundation of Guangxi Key Laboratory of Cryptography and Information Security (GCIS202118); Science and Technology Support Plan of Guizhou Province ([2020]2Y011).

Availability of data and materials
The datasets generated during and/or analyzed during the current study are available from the corresponding author on reasonable request.

Declarations

Competing interests
The authors declare that they have no competing interests.

Author details
[1]State Key Laboratory of Public Big Data, College of Computer Science and Technology, Guizhou University, Guiyang, China. [2]Guangxi Key Laboratory of Cryptography and Information Security, Guilin University of Electronic Technology, Guilin, China. [3]Blockchain Laboratory of Agricultural Vegetables, Weifang University of Science and Technology, Shouguang, China. [4]Engineering Lab for Cloud Security and Information Security Center, Beijing University of Posts and Telecommunications, Beijing, China.

Received: 27 June 2021 Accepted: 7 September 2021
Published online: 25 September 2021

References

1. Xu X, Wu Q, Qi L, Dou W, Tsai S, Bhuiyan M (2021) Trust-aware service offloading for video surveillance in edge computing enabled internet of vehicles. IEEE Trans Intell Transp Syst 22(3):1787–1796. https://doi.org/10.1109/TITS.2020.2995622
2. Fu Y, Hou Y, Wang Z, Wu X, Gao K, Wang L (2021) Distributed scheduling problems in intelligent manufacturing systems. Tsinghua Sci Technol 26(5):625–645
3. Xu X, Liu X, Xu Z, Dai F, Zhang X, Qi L (2020) Trust-oriented iot service placement for smart cities in edge computing. IEEE Internet Things J 7(5):4084–4091. https://doi.org/10.1109/JIOT.2019.2959124
4. Xue R, Wu Y, Liu M, Zhang L, Zhang R (2015) Progress in verifiable computation. Sci Sin Informationis 45(11):1370–1388
5. Alptekin K (2017) Incentivized outsourced computation resistant to malicious contractors. IEEE Trans Dependable Secure Comput 14(6):633–649
6. Zhang X, Liu Y, Chen Y (2021) A new entropic criterion model in rational secure two-party computation. J Ambient Intell Humanized Comput 9:1–10
7. Wang Y, Yang G, Li T, Li F, Tian Y, Yu X (2020) Belief and fairness: A secure two-party protocol toward the view of entropy for iot devices. J Netw Comput Appl 161:102641
8. Li J (2016) Data verification and verifiable commissioned computing scheme based on fully homomorphic encryption. J Beijing Electron Sci Technol Inst 24(1):21–25
9. Wang Y, Yang G, Bracciali A, Leung H, Yu X (2020) Incentive compatible and anti-compounding of wealth in proof-of-stake. Inf Sci 530:85–94
10. Zhang L, Safavi-Naini R (2015) Batch verifiable computation of outsourced functions. Des Codes Crypt 77(2):563–585
11. Carbunar B, Tripunitara M (2010) Fair payments for outsourced computations. In: Proceedings of the Seventh Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks, SECON: 21-25 June, 2010. IEEE, Boston. pp 1–9

12. Xu G, Amariucai G, Guan Y (2017) Delegation of computation with verification outsourcing: curious verifiers. IEEE Trans Parallel Distrib Syst 28(3):717–730
13. Huang H, Chen X, Wu Q, Huang X, Shen J (2018) Bitcoin-based fair payments for outsourcing computations of fog devices. Futur Gener Comput Syst 78:850–858
14. Chen X, Li J, Susilo W (2012) Efficient fair conditional payments for outsourcing computations. IEEE Trans Inf Forensics Secur 7(6):1687–1694
15. Yin X, Tian Y, Wang H (2018) Fair and rational delegation computation protocol. J Softw 29(7):1953–1962
16. Tang J, Li R, Wang K, Gu X, Xu Z (2020) A novel hybrid method to analyze security vulnerabilities in android applications. Tsinghua Sci Technol 25(5):589–603
17. Azrour M, Mabrouki J, Guezzaz A, Farhaoui Y (2021) New enhanced authentication protocol for internet of things. Big Data Min Analytics 4(1):1–9
18. Li P, Li K, Wang Y, Zheng Y, Wang D, Yang G, Yu X (2020) A systematic mapping study for blockchain based on complex network. Concurr Comput Pract Experience 4:5712
19. Wang S, Tang X, Zhang Y, Chen J (2019) Auditable protocols for fair payment and physical asset delivery based on smart contracts. IEEE Access 7:109439–109453
20. Chen Y, Guo J, changlou L, Ren W (2019) Fade: A blockchain-based fair data exchange scheme for big data sharing. Future Internet 11(11):225
21. Quanxing Z, Qiuxian L, Meimei F (2020) Anti-collusion delegation computation protocol of three-party game based on smart contract. Comput Eng 46(8):124–131
22. Dong C, Wang Y, Aldweesh A, McCorry P, van Moorsel A (2017) Betrayal, distrust, and rationality: Smart counter-collusion contracts for verifiable cloud computing. In: Thuraisingham B, Evans D, Malkin T, Xu D (eds). Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS: 30 October 2017. ACM, Dallas. pp 211–227
23. Chen Z, Tian Y, Xiong J, Peng C, Ma J (2021) Towards reducing delegation overhead in replication-based verification: An incentive-compatible rational delegation computing scheme. Inf Sci 568:286–316
24. Song L, Li T, L W (2019) Application of game theory in blockchain. Chinese J Crypt 6(1):100–111
25. Li T, Wang Z, Yang G, Cui Y, Chen Y, Yu X (2021) Semi-selfish mining based on hidden markov decision process. Int J Intell Syst 36(7):3596–3612. https://doi.org/10.1002/int.22428
26. Li T, Chen Y, Wang Y, Wang Y, Zhao M, Zhu H, Tian Y, Yu X, Yang Y (2020) Rational protocols and attacks in blockchain system. Secur Commun Netw 2020(44):1–11
27. Wang Q, Liu X, Liu W, Liu A, Liu W, Mei T (2020) Metasearch: Incremental product search via deep meta-learning. IEEE Trans Image Process 29:7549–7564. https://doi.org/10.1109/TIP.2020.3004249
28. Liu Y, Wang F, Yang Y, Zhang X, Wang H, Dai H, Qi L (2021) An attention-based category-aware gru model for next poi recommendation. Int J Intell Syst 36(7):3174–3189
29. Wang F, Zhu H, Srivastava G, Li S, R M, Khosravi, Qi L (2021) Robust collaborative filtering recommendation with user-item-trust records. IEEE Trans Comput Soc Syst PP(99):1–11. https://doi.org/10.1109/TCSS.2021.3064213
30. Xiao L, Chen Y, Lin W, Liu K (2012) Indirect reciprocity security game for large-scale wireless networks. IEEE Trans Inf Forensics Secur 7(4):1368–1380
31. Zhao Y, Li Y, Mu Q, Yang B, Yu Y (2018) Secure pub-sub: Blockchain-based fair payment with reputation for reliable cyber physical systems. IEEE Access 6:12295–12303
32. Wang Y, Li T, Liu Q, Sun J, Liu Z (2015) The impact of social cloud reputation and structure on rational computation. J High Speed Netw 21(3):181–194
33. Jiang X, Tian Y (2020) Rational delegation of computation based on reputation and contract theory in the UC framework. In: Yu S, Mueller P, Qian J (eds). Security and Privacy in Digital Economy - First International Conference, SPDE:30 October 2020. Springer, Quzhou. pp 322–335
34. Li F, Wang D, Wang Y, Yu X, Wu N, Yu J, Zhou H (2020) Wireless communications and mobile computing blockchain-based trust management in distributed internet of things. Wirel Commun Mob Comput 2020(5):1–12
35. Kou H, Liu H, Duan Y, Gong W, Yanwei, Xu X, Qi L (2021) Building trust/distrust relationships on signed social service network through privacy-aware link prediction process. Appl Soft Comput 100(5):106942. https://doi.org/10.1016/j.asoc.2020.106942
36. Li T, Tian Y, Xiang K, Gao H (2020) A fair payment scheme based on blockchain in delegated computing. J Commun 41(3):80–90
37. Wahby R, Howald M, Garg S, Shelat A, Walfish M (2016) Verifiable asics. In: IEEE Symposium on Security and Privacy, SP: 22-26 May 2016. IEEE Computer Society, San Jose. pp 759–778
38. Fudenberg D, Tirole J (1992) Game theory. Economica 60(238):841–846
39. Christidis K, Devetsikiotis M (2016) Blockchains and smart contracts for the internet of things. IEEE Access 4:2292–2303
40. Wang Y, Wang Y, Wang Z, Yang G, Yu X (2020) Research cooperations of blockchain: toward the view of complexity network. J Ambient Intell Humanized Comput:1–14
41. Vivar A, Orozco A, García-Villalba L (2021) A security framework for ethereum smart contracts. Comput Commun 172:119–129
42. Jin Y, Guo W, Zhang Y (2020) A time-aware dynamic service quality prediction approach for services. Tsinghua Sci Technol 2(11):227–238
43. Bhardwaj N, Sharma P (2021) An advanced uncertainty measure using fuzzy soft sets: Application to decision-making problems. Big Data Min Analytics 4(2):94–103

Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.