Cooperative Game Model of Delegation Computing: Verifier Separated from Calculators

1st Duo Zhang
State Key Laboratory of Public Big Data,
College of Computer Science and Technology,
College of Mathematics and Statistics
Guizhou University
Guiyang, China
email: sxzd0816@163.com

2nd Youliang Tian
State Key Laboratory of Public Big Data,
College of Computer Science and Technology,
Institute of Cryptography and Date Security
Guizhou University
Guiyang, China
email: youliangtian@163.com

3rd Linjie Wang
State Key Laboratory of Public Big Data,
College of Computer Science and Technology
Guizhou University
Guiyang, China
email: wanglinjie_66@hotmail.com

4th Sheng Gao
School of Information,
Central University of Finance and Economics,
Beijing, China
email: sgao@cufe.edu.cn

5th Jianfeng Ma
School of Cyber Engineering,
College of Computer Science and Technology, Xi'dian University
Xi'an, China
email: jfma@mail.xidian.edu.cn

***Abstract*—In cloud computing, the delegation computation service is required because end users usually are resource-constrained, and at the same time the correctness of computing results needs to be verified. However, all the existing and available technologies have a high cost to verify which is the main problem faced by the delegation computing under the cloud platform. In order to address this issue, based on the game theory and a smart contract, we construct three protocols. The client and the calculator sign the Prisoner's Protocol to incentivize correct computation by asking the calculator to pay a deposit upfront; The client and the verifier sign the Long-acting Mechanism Protocol to ensure the validity of the verification results; The calculator and the verifier sign the Collusion Protocol to make collusion the most profitable strategy for all colluding parties. Through the combination of these protocols, the work is realized by separation the computing task and the verification task, and the heavy verification task is avoided. Finally, the performance analysis of the results shows that the combination of all the protocols not only solves the problem of verification complexity in the traditional delegation computing, but also guarantees the benefit of the honest player.**

***Index Terms*—Cloud computing, Smart contract, Game theory, Rational delegation computing, Verification complexity**

I. INTRODUCTION

In the age of big data, a large amount of data needs to be computed and stored, which easily leads to a serious shortage of local resources [1]. The emergence of cloud computing technology has well solved this problem. In the cloud computing, resource-constrained users delegate their own computing and data to the platform provided by cloud services for processing and storage, thus bringing many benefits to users [2] [3]. However, delegation computing under cloud platforms urgently needs verifiability: the cloud service providers differ from users in that their benefit may point to different directions, and users cannot trust the cloud completely. For a variety of reasons, clients often need to verify the correctness of the computing results. At the same time, the privacy of users and the correctness of the results are facing serious security challenges. Therefore, it is extremely urgent and realistic significance to introduce rational players and realize verifiable delegation task at a reasonable cost through the game theory and the smart contract.

Over the last few decades, a lot of research have been done on the cloud platform security delegation technology. Initially, researchers wanted to design a general computing framework to implement delegation computing for all problems. Gentry [4] proposed a full homomorphic encryption (FHE), then Gennaro et al. [6] first proposed a general delegation computing scheme based on FHE and Encrypted Boolean Circuit [5], which not only can effectively protect user privacy, but also can verify the results of the server. In addition, Chung et al. [7] proposed an improved scheme, which reduced the complexity of the general delegation computing scheme. However, because the FHE algorithm contains extremely complex computing operations and large-scale circuit size, the schemes of Gennaro et al. and Chung et al. have very high computing complexities. Therefore, researchers began to turn to the delegation computing scheme for specific problems,

---

This work is supported by Key Projects of the Joint Fund of the National Natural Science Foundation of China under Grant No.U1836205; the National Natural Science Foundation of China under Grant No.61662009, No.61772008; the Guizhou Province Science and Technology Major Special Plan No.20183001; the Guizhou Provincial Science and Technology Plan Project under Grant No.[2017]5788; the Ministry of Education-China Mobile Research Fund Project under Grant No.MCM20170401; the Guizhou University Fostering Project No.[2017]5788; Research on Key Technologies of Blockchain for Big Data Applications under Grant No.[2019]1098; Foundation of Postgraduate of Guizhou Province (No.YJSCXJH2019015).

hoping to design practical and available schemes.

So far, there are a lot of security delegation computing schemes for specific problems. Atallah et al. [8] first proposed a secure delegation computing scheme for matrix operation. Blanton et al. [9] proposed an improved sequence security delegation scheme. Hohenberger et al. [10] proposed a secure delegation computing scheme for the modular exponential operation. However, these schemes need two non-collusive servers to implement secure the delegation computing, which cannot effectively resist collusive attacks among servers. To address this issue, Atallah et al. [11] proposed a secure delegation computing scheme for matrix multiplication based on Shamir's secret sharing technology [12]. The scheme has only one server and there is no collusion attack. However, the secret sharing technology makes the scale of the delegation matrix multiplication problem increase sharply which lead high communication load. Although there are many shortcomings in the above-mentioned secure delegation computing schemes, it points out the further research directions of security delegation computing: *Protected user's privacy*, *Verified the results of delegation computing* and *Resisted collusion attacks among servers*.

Recently, great efforts have been made in the design of the delegation computing scheme and some important results have been achieved. For example, Wang et al. [13] proposed a secure delegation computing scheme for linear programming under the cloud platform. Then, the focus of the follow-up research on secure delegation computing scheme has evolved to reduce the computing complexity of the scheme as much as possible [14] [15] [16] [17] [18]. Because large-scale computing problems are common in practical applications, delegation computing schemes for different problems have been continuously researched [19]. Other research work related to secure delegation computing on cloud platform mainly includes: Yao [5] proposed a secure multi-party computation, which makes multiple independent computing players get together to solve problems and ensures that input values are not leaked to other players [20] [21] [22]. Golle et al. [23] recognized the reliability of the cloud platform's delegated computing results by inserting some prior knowledge into the delegated computing problem. Du et al. [24] uses a grid computing to detect the cheating behavior of cloud platform. Zhang et al. [25] design a secure delegated storage scheme based on game theory, which can effectively reduce the probability of audit disputes between users and cloud platforms. In a word, some current research results related to the secure delegate computing cannot be directly applied to the delegate computing of large-scale computing problems.

A. Contribution

In this paper, we propose a rational delegation computation fair protocol. The contributions of this paper are as follow:

- We propose a rational delegation computing game model, separate the computing task from the verification task and construct a three-player game model, so that the client can use the calculator and the verifier to complete the delegation task.
- The smart contract is introduced to realize the economic incentive mechanism to generate the benefit contradiction and distrust between the two players, so as to prevent the two players from colluding to cheat the client under the rational choice and realize the reliability of the computation results.
- In the rational delegation computing scheme, we formally analyze the game generated by the sub-protocol, give the conditions for the existence of the Nash equilibrium solution, and prove the effectiveness of these protocols under reasonable assumptions in order to ensure the benefits of honest calculators.

B. Organization

The rest of the paper is organized as follows: Some preliminaries are given in Section 2. In Section 3, we propose the system ideal model, adversary model and the system architecture of rational delegation computation fair protocol. The rational delegation computation fair protocol are presented in Section 4. The performance comparison and theoretical analysis are discussed in Section 5. Finally, we give a brief conclusion.

II. PRELIMINARIES

In TABLE I , we present notations mainly used in this paper.

TABLE I
THE PARAMETERS USED IN THIS PAPER

Symbol	**Significance**
w	The amount that the client agrees to pay to the calculator for computing the task.
v	The amount that the client agrees to pay to the verifier for computing the task.
c	The cloud's cost for computing the task.
d	The calculator deposit a cloud needs to pay to the client in order to get the job.
t	The deposit the colluding parties need to pay in the collusion agreement.
b	The bribe paid by the calculator of the collusion to the verifier in the collusion agreement.
m	The verifier deposit a cloud needs to pay to the client in order to get the job.

- $u \leq w$, $w \geq c$, $u \geq c$, $c > b$
- $t < c + d$, $d > 2t$, $m > v + t + b$, $t < b$, $T_1 < T_2 < T_3$.

III. IDEAL MODEL AND ADVERSARY MODEL AND SYSTEM ARCHITECTURE

A. Adversary Model

In the rational delegation computing model, we consider the verifier to be an honest player, which provides the client with the correct verification results. In addition, the client is an honest player, which provides a meaningful computing task and strictly follow the protocol. However, the verifier and calculator are also interested to compute data belonging to other parties. From this point of view, we introduce a rational adversary P' in our model, i.e. the verifier is a rational player. The goal of P' is to gain the trust of the client and obtain the benefit of the calculator, his final results has the following capabilities:

- P' may eavesdrop all communications to obtain the computations;
- P' may compromise the calculator to guess the verification value of the all computation outsourced from the client;
- P' may compromise the verifier to guess the computation value sent from the client.

The rational adversary is, however, restricted from compromising both the client and calculator. We remark that such restrictions are typical in adversary models used in the delegation computing.

B. System Architecture of Rational Delegation Computation Fair Protocol

In fact, the client needs to call TTP because of resource constrained, but the cost of calling TTP is often high, so the client is less willing to call TTP. In order to address this issue, the client delegates the computation and validation tasks to the calculator P_1 and verifier P_2, respectively. It assumes that the P_1 and P_2 have the same computing power and can complete the computing tasks independently. However, the calculator P_1 often takes some measures to reduce the computing cost and bribe the verifier P_2, thus easily creating a prisoner's dilemma model between them. i.e. the calculator P_1 and verifier P_2 collude to calculate and get higher returns than the honest calculation, but they know that collusion is unstable. Because whichever side initiated the collusion was seen as a trap, the other side always deviated from it. In order to prevent this phenomenon, we combine intelligent contract and game theory to put forward a rational delegation computation fair protocol to ensure the reliability of calculation results. System Architecture of Rational Delegation Computation Fair Protocol. The system architecture of the rational delegation computation fair Protocol is shown in Fig.1 below.

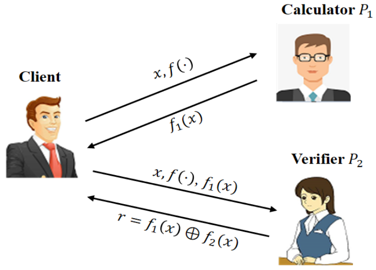

Fig. 1. System Architecture of Rational Delegation Computation Fair Protocol

The specific process is as follows.

- The client and P_1 conclude a contract, trying to stimulate the correct computing by requiring the P_1 to pay the deposit in advance. If the act of P_1 is honest, the deposit will be refunded; if the act is dishonest, the deposit will be owned by the client.
- The client and P_2 conclude a betrayal contract and tried to sabotage the conspiracy with P_1 through additional incentives and penalties. The purpose of this contract is not to encourage P_1 to deviate from collusion, but to encourage P_2 to report collusion.
- The P_1 concludes a collusion contract with P_2, and P_1 tries to encourage the conspiracy by paying a certain amount of bribes to P_2. The violating party is punished simultaneously.

Rational delegation computation fair protocol consists of three phases: the conclude contracts phase, the computing phase and the verification phase. The detailed process of the phases are as follows.

- Conclude contracts phase
  1) The client signs contracts π^1 and π^2 with the calculator P_1 and verifier P_2, respectively, which is public.
  2) The calculator P_1 and verifier P_2 conclude collusive contracts π^3 secretly.
- Computing phase
  1) The client sends computing task x, function $f(\cdot)$ to the P_1.
  2) The P_1 receives computing task x, function $f(\cdot)$. Then $f_1(x)$ is computed.
  3) The P_1 sends computing result $f_1(x)$ to the client.
- Verification phase
  1) The client sends computing task x, function $f(\cdot)$ to the P_2.
  2) The P_2 receives computing task x, function $f(\cdot)$, $f_1(x)$. Then $f_2(x)$ is computed.
  3) The P_2 sends verification results $r = f_1(x) \bigoplus f_2(x)$ to the client.
  4) The client pays corresponding fees to P_1, P_2 according the result from P_2.

IV. RATIONAL DELEGATION COMPUTING FAIR PROTOCOL

Protocol 1: Prisoner's contract π^1 and its implementation process:

- The client concludes the prisoner's Contract π^1 and the long-term mechanism contract π^2 with the calculator P_1 and the verifier P_2, respectively;
- The calculator P_1 concludes the collusion contract π^3 with the verifier P_2;
- The calculator P_1 agrees to compute function $f(\cdot)$ with input x, the verifier P_2 agrees to verify the result $f_1(x)$ computed by the calculator P_1;
- The time limit is $T_1 < T_2 < T_3$, which is agreed by the client, the calculator P_1 and the verifier P_2;
- The entrusting party agrees to pay to the calculator P_1 for ensuring that the result $f_1(x)$ is computed correctly and timely;
- The client agrees to pay to the verifier P_2 for ensuring that result $f_1(x)$ of the calculator P_1 is verified correctly and timely;
- As a constraint, both the calculator P_1 and the verifier P_2 must pay a deposit when signing the contract π^3, and the deposit is held by the smart contract;

- The calculator P_1 and the verifier P_2 must pay the deposit before T_1. If either party fails to do so, the contract will be terminated and deposit will be refunded;
- The calculator P_1 must pass the computing result $f_1(x)$ before time limit T_2;
- The client receives the computing result $f_1(x)$ from the calculator P_1 before T_2. The client performs the following operations:
    1) The client sends the input x along with the calculation functions $f(\cdot)$ and $f_1(x)$ to the verifier P_2; if the verification result is not delivered before T_3, the client will deduct the deposit and pay to the calculator P_1;
    2) If the verification result is delivered before T_3 and the verification result is $r = 0$, the client must pay v to the calculator P_1 and the verifier P_2, respectively, and refunded deposits;
    3) If the verification result is delivered before T_3 and the verification result is $r = 1$, the deposit d paid by the calculator P_1 shall be owned by the client, and the client shall reward $d/2$ to the verifier, and the contract π^1 shall be terminated.
- If the client does not receive the computing result $f_1(x)$ after T_2, the deposit d will be owned by the client and the contract π^1 will be terminated.

Actually, despite the high fines, the conspirators can still strike secret protocols to redistribute profits and punish those who stray first from the collusion. Of course, the client does not want a collusion protocol between the calculator P_1 and the verifier P_2. If a collusion protocol exists, the client would prefer that verifier P_2 uncover the calculator P_1 that initiated the collusion. Therefore, the client and the verifier P_2 also need to sign a long-term mechanism contract π^2.

Protocol 2: Long-term mechanism contract π^2 and its implementation process:

- The client concludes a long-term mechanism π^2 contract with the verifier P_2 before T_2;
- The verifier P_2 only concludes a collusion contract π^3 with the calculator P_1, and the client agrees to compensate the verifier P_2 for the loss of the collusion contract π^3 under appropriate circumstances;
- The verifier P_2 must deliver the verification result before T_3. If the result is not delivered, the client will deduct the deposit m;
- As a necessary condition for long-term cooperation, the client must pay a deposit $b + t + d/2$ to the verifier P_2, which is equal to the maximum amount of possible loss in the conspiracy contract plus incentives. The deposit is held by a smart contract;
- If the verifier P_2 delivers the verification result $r = 1$ before T_3, the client pays the reward $d/2$ in order to encourage the verifier P_2;
- If the verifier P_2 delivers the verification result $r = 0$ before T_3, the client pays v to the verifier P_2 and refunds the deposit m;
- When the client discovers that the verifier P_2 did not report or misreported conditions, besides deducting deposit m, he also broadcasts the dishonest behavior of the verifier P_2 on the block chain by using the smart contract technology, which makes it impossible for the verifier P_2 to accept computing or verification tasks in the future.

In order to report the collusion promptly, two procedures should be taken into account:

- The collusion contract π^3 is signed before the collusion is reported to the client.
- The collusion contract π^3 is signed only after the long-term mechanism contract π^2 is signed with the client.

Realistically, the collusion contract π^3 provides additional rules in addition to normal transactions, which will affect the remuneration of both parties and provide profitable strategies for the collusive verifier.

Protocol 3: Collusion contract π^3 and its implementation process:

- The contract π^3 is signed by the calculator P_1 and the verifier P_2, and the calculator P_1 is the initiator of the conspiracy;
- The verifier P_2 agrees to provide $r = 0$ as verification result of the protocol π^1;
- As a necessary condition, the calculator P_1 must pay $t + b$ to the verifier P_2, and the verifier P_2 must pay t before they sign the collusion contract π^3. This part of the deposit is held by the smart contract;
- The calculator P_1 and the verifier P_2 must pay the above amount before T_2. If any participant fails to do so, the contract π^3 will be terminated and the deposit paid will be refunded;
- After the completion of the prisoner's contract π^1, the balance in the contract π^3 will be treated as follows:
    1) If both the calculator P_1 and the verifier P_2 abide by the contract π^3 which means the verifier P_2 outputs the permanent result $r = 0$. Then the calculator P_1 pays the bribery b to the verifier P_2 and refunds the deposit of both parties;
    2) If only the calculator P_1 violates the contract π^3, which means the verifier P_2 outputs the permanent result $r = 1$, the calculator confiscates the deposit t paid by the verifier P_2;
    3) If only the verifier P_2 abides by the contract π^3 which means whether the calculator P_1 complies with the contract π^3 or not, the verifier P_2 still outputs the permanent result $r = 0$. Then the calculator P_1 pays bribery b to the verifier P_2 and the deposit is refunded to both parties;
    4) If both the calculator P_1 and the verifier P_2 violated the terms of the contract π^3, the deposit of both parties shall be refunded.

When the collusion contract π^3 is signed, the time result will be grasped, otherwise it is invalid. There are absolute advantages for the verifier P_2 in the process of contract execution. In the process of conspiracy, the verifier P_2 will receive an additional reward regardless of whether he betrays the calculator P_1 or not, but in the case of the client is least willing to see the verifier P_2. Consequently, the client and the verifier P_2 sign a long-term mechanism contract π^2 to ensure the reliability of the verification result. Furthermore, the equilibrium remains unchanged. The above execution process is shown in Fig.2 below:

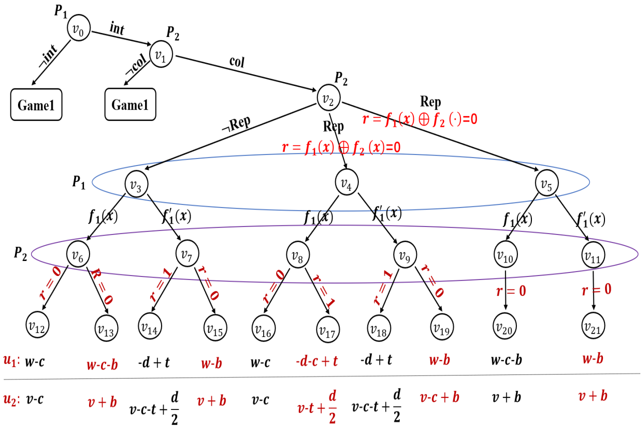

Fig. 2. The game induced by the prisoner's contract, the long-term mechanism contract and the collusion contract.

TABLE II
PAYOFF ANALYSIS OF GAME MODEL

Out	r	**P**	π^1		π^2		**Cost**	**Total**
			Clau	Pay	Clau	Pay		
v_{12}	$f_1 \bigoplus f_2 = 0$	P_1	$10b$	w	$5d$	0	c	$w-c$
		P_2		v		0	c	$v-c$
v_{13}	$f_1 \bigoplus f_1 = 0$	P_1	$10b$	w	$5c$	$-b$	c	$w-c-b$
		P_2		v		b	0	$v+b$
v_{14}	$f'_1 \bigoplus f_2 = 1$	P_1	$10c$	$-d$	$5b$	t	0	$-d+t$
		P_2		$v+d$		$-t$	c	$v-c-t+d/2$
v_{15}	$f'_1 \bigoplus f'_2 = 0$	P_1	$10b$	w	$5a$	$-b$	0	$w-b$
		P_2		v		b	0	$v+b$
v_{16}	$f_1 \bigoplus f_2 = 0$	P_1	$10b$	w	$5d$	0	c	$w-c$
		P_2		v		0	c	$v-c$
v_{17}	$f_1 \bigoplus f'_2 = 1$	P_1	$10c$	$-d$	$5b$	t	c	$-d-c+t$
		P_2		$v+d$		$-t$	0	$v-t+d/2$
v_{18}	$f'_1 \bigoplus f_2 = 1$	P_1	$10c$	$-d$	$5b$	t	0	$-d+t$
		P_2		$v+d$		$-t$	c	$v-c-t+d/2$
v_{19}	$f'_1 \bigoplus f'_1 = 0$	P_1	$10b$	w	$5a$	$-b$	0	$w-b$
		P_2		v		b	c	$v-c+b$
v_{20}	$f'_1 \bigoplus f_1 = 0$	P_1	$10b$	w	$5c$	$-b$	c	$w-c-b$
		P_2		v		b	0	$v+b$
v_{21}	$f'_1 \bigoplus f_2 = 0$	P_1	$10b$	w	$5a$	$-b$	0	$w-b$
		P_2		v		b	0	$v+b$

The rational delegation computing model and analysis triggered by three contracts is presented in Fig. 3. The participants are the calculator P_1 and the verifier P_2, that is, the verifier $P = \{P_1, P_2\}$. Although the client is also involved in the contract, he is honest and has only one deterministic strategy, he can be eliminated from the game model. The calculator P_1 and the verifier P_2 can communicate with each other. Action set $A = \{f_1(x), f'_1(x), f_2(\cdot)\}$, the first two means that the calculator P_1 sends $f_1(x)$ or $f'_1(x)$ before the deadline, and the last means any action that the verifier P_2 may take. The model has seven information sets: $I_{11} = \{v_0\}$ and $I_{11} = \{v_3, v_4, v_5\}$ belong to the calculator P_1, $I_{21} = \{v_1\}$, $I_{22} = \{v_2\}$, $I_{23} = \{v_6, v_7\}$, $I_{24} = \{v_8, v_9\}$, $I_{25} = \{v_{10}, v_{11}\}$ both of them belong to the verifier P_2. u_1 and u_2 represent the utility functions of P_1 and P_2, respectively. The remuneration of each party is listed below the terminal node. TABLE II shows the method of calculating the amount of payment.

Theorem: Let $m > v + t + b$, $d > 2t$, $b < c$, the calculator P_1 and the verifier P_2 are rational participants, then the implementation protocol terminates at $\{v_{12}\}$.

Proof: In the protocol, the calculator P_1 always acts first as a collusion initiator. If P_1 does not initiate an collusion or P_2 rejects the collusion, then π^1 is executed because collusion contract π^3 has not been signed. If P_2 agrees to collude with P_1, and there is $b < c$, they will enter different branches. Since the collusion contract π^3 has been signed at this time, the payment in this branch is totally different from that in the protocol π^1. In this branch, if P_2 does not report the collusion with P_1 to the client, it is sure that one of two things are going to happen, and that is that P_1 is going to make an honest calculation to get the maximum benefit of $w - c$ or collude with P_2 to get the maximum benefit of $w - b$. Obviously, the benefits of collusion are greater. At this time, the verifier's benefit $v+b$ or $v-c-t+d/2$ is greater than $v-c(d > 2t)$, but considering the deterrence of broadcast of personal reputation on the chain and the deposit $m(m > v + t + b)$, the verifier P_2 signed a protocol π^2 with the client. Thus, he chooses to calculate honestly, and end up at $\{v_{21}\}$. In this branch, if P_2 chooses to report his collusion with P_1 to the client, the best information $\{v_{17}\}$ benefit of P_2 is $v-c+d/2$, and the benefit of P_1 is $-d - v + t < 0$; P_2's best information $\{v_{20}\}$, $\{v_{21}\}$ benefits are $v + b$. However, for P_1, the benefit $w - c - b$ of information $\{v_{20}\}$ is less than that of honest calculation, while the honest calculation benefit of information $\{v_{21}\}$ is $w - b$. Similarly, the signatory verifier of the protocol π^2, P_2 will not choose this strategy.

Furthermore, from the standpoint of considering personal benefits and future credit development, we can know that in the case of long-term mechanism contract π^2 implementation, the verifier P_2 will not collude with the calculator P_1 or tell his collusion with the calculator P_1 to the client. In this case, even though the calculator initiates the collusion protocol π^3, it receives the best benefit $\{v_{13}\}$ and $\{v_{20}\}$, his income $w-c-b$ is less than that of the normal calculation. Therefore, for the sake of personal interests, it is impossible for the calculator P_1 to initiate collusion. In summary, if the calculator and the verifier are rational participant, the implementation of the protocol is fairness and will be terminated at $\{v_{12}\}$ and Nash equilibrium will be achieved.

V. PERFORMANCE ANALYSIS

A. Performance Comparison

Our work is closely related to [17] and [26], which tackle this issue of the verifiability of rational delegate computing. The contrast of information anti-collusion and complexity between the protocol in this paper and other schemes is presented in TABLE III. All protocols are implemented by constant rounds, but they cannot be implemented simultaneously in terms of information integrity and defense against collusion. However, the protocol in this paper ensures fairness through smart contract and reliability of verification results through long-term mechanism contract, and meanwhile reduces the cost of the client.

TABLE III
PROTOCOLS COMPARISON

	Comp	**Comm**	**Inf**	**ver**	**Anti-coll**
[17]	$O(nlogn)$	≥ 2	completion	✓	×
[26]	$O(n)$	≥ 2	completion	✓	✓
Ours	$O(n)$	2	incompletion	✓	✓

B. Theoretical Analysis

To implement the contracts, we will need to resolve the following challenges [17]: Privacy, Verifiability, Efficiency.

To address the issues, we use a suitable collision resistant hash function. Informally, a commitment scheme is a two-phase protocol. In the commitment phase, a committer commits to a value m by choosing a secret s to generate a commitment $Com_s(m)$. The commitment should be hiding, that is to say, it is infeasible to know m given only $Com_s(m)$ but not s; the commitment should also be binding, that is to say, it is infeasible to find $m' \neq m$ and $s' \neq s$ such that $Com_{s'}(m') = Com_s(m)$. Instead of using the plaintext, the implementation of the contracts needs to handle cryptographic values and the parties need to run some protocols.

The additional overhead incurred by cryptography is small. In every contract, each party at most 2 commitments need to be generated and verified. The size can be further reduced if point compression is used. As is all known, the cost is roughly related to the computational and storage complexity of the function. Therefore, the financial cost of using the smart contracts is low [17]. Furthermore, the proposed scheme not only reduces the clients verification task, prevents a collusion between the calculator and the verifier. A comparative summary between the three schemes is shown in TABLE III.

VI. CONCLUSION

Verifiability is a very important feature of the rational delegation computing. To illustrate the Verifiability of the rational delegation computing, we construct a protocol based on a smart contract. The calculation and verification tasks are assigned to a calculator and a verifier, respectively, and the game model is established between two rational players by using the smart contract in our protocol. The protocol prevents the players from collude with each other and returning back the wrong result, and ensure the benefits of honest calculation. The client does not need to verify the result of computing any more, but only pays the players' fees according to the result of the feedback from the verifier, which reduces the workload of the client a lot.

REFERENCES

[1] Y. Al-Dhuraibi, F. Paraiso, N. Djarallah and P. Merle, "Elasticity in Cloud Computing: State of the Art and Research Challenges," IEEE Transactions on Services Computing, vol.11, no.2, pp.430-447, April. 2018.

[2] M. Armbrust, A. Fox et al, "A View of Cloud Computing," Communications of the ACM, vol.53, no.4, pp.50-58, April. 2010.

[3] L. Zhang, "Editorial: Big Services Era: Global Trends of Cloud Computing and Big Data," IEEE Transactions on Services Computing, vol.5, no.4, pp.467-468, June. 2017.

[4] C. Gentry, "Fully Homomorphic Encryption Using Ideal Lattices," In Proceedings of the Annual ACM Symposium on Theory of Computing, vol.9, no.4, pp.169-178, June. 2009.

[5] A. Yao, "Protocols for Secure Computings (extensive abstract)," In Proceedings of Symposium on Foundations of Computer Science, 2010.

[6] R. Gennaro, C. Gentry, and B. Parno, "Non-Interactive Verifiable Computing: Outsourcing Computing to Untrusted Workers," Lecture Notes in Computer Science, vol.6223, no.3, pp.465-482, 2010.

[7] K. Chung, Y. Kalai, and S. Vadhan, "Improved Delegation of Computing Using Fully Homomorphic Encryption," In Proceedings of International Conference on Advances in Cryptology, 2010.

[8] M. J. Atallah and J. Li,Secure, "Outsourcing of Sequence Comparisons," International Journal of Information Security, vol.4, no.4, pp.277-287, October. 2005.

[9] M. Blanton, M. J. Atallah, K. B. Frikken, and Q. Malluhi, "Secure and Efficient Outsourcing of Sequence Comparisons," Lecture Notes in Computer Science, 2012.

[10] S. Hohenberger and A. Lysyanskaya, "How to Securely Outsource Cryptographic Computings," In Proceedings of International Conference on Theory of Cryptography, 2005.

[11] M. Atallah and K. Frikken, "Securely Outsourcing Linear Algebra Computings," In Proceedings of ACM Asia Conference on Communications Security, 2010.

[12] A. Shamir, "How to Share a Secret," Communications of the ACM, vol.22, no.11, pp.612-613, November. 1979.

[13] C. Wang, K. Ren, and J. Wang, "Secure and Practical Outsourcing of Linear Programming in Cloud Computing," In Proceedings of IEEE International Conference on Computer Communications, 2011.

[14] X. Lei, X. Liao, T. Huang, H. Li, and C. Hu, "Outsourcing the Large Matrix Inversion Computing to a Public Cloud," IEEE Transactions on Cloud Computing, vol.1, no.1, June. 2013.

[15] X. Lei, X. Liao, T. Huang, H. Li, and C. Hu, "Cloud Computing Service: The Case of Large Matrix Determinant Computing," IEEE Transactions on Services Computing, October. 2015.

[16] L. Zhou and C. Li, "Outsourcing Eigen-Decomposition and Singular Value Decomposition of Large Matrix to a Public Cloud," IEEE Access, 2016.

[17] C. Y. Dong, Y. L. Wang and A. Aldweesh, "Betrayal, Distrust, and Rationality: Smart Counter-Collusion Contracts for Verifiable Cloud Computing," In Proceedings of ACM SIGSAC Conference on Computer and Communications Security, November. 2017.

[18] X. Chen, X. Huang, J. Li, J. Ma, W. Lou and D. Wong, "New Algorithms for Secure Outsourcing of Large-Scale Systems of Linear Equations," IEEE Transactions on Information Forensics and Security, vol.10, no.1, pp.69-78, January. 2015.

[19] C. Luo, J. Ji, X. Chen, M. Li, L. Yang and P. Li, "Parallel Secure Outsourcing of Large-scale Nonlinearly Constrained Nonlinear Programming Problems," IEEE Transactions on Big Data, pp.13-24, March. 2018.

[20] R. Azaxderakhsh, D. Fishbein et al, "Fast Software Implementations of Bilinear Pairings," IEEE Transactions on Dependable and Secure Computing, vol.17, no.6, pp.605-619, December. 2017.

[21] J. Vaidya, "A Secure Revised Simplex Algorithm for Privacy-Preserving Linear Programming," In Proceedings of International Conference on Advanced Information Networking and Applications, May. 2009.

[22] O. Catrina and S. Hoogh, "Secure Multiparty Linear Programming using Fixed-Point Arithmetic," In Proceedings of European Symposium on Research in Computer Security, 2010.

[23] P. Golle and I. Mironov, "Uncheatable Distributed Computings," Topics in Cryptology-CT-RSA, April. 2001.

[24] W. Du, J. Jia, M. Mangal and M. Murugesan, "Uncheatable Grid Computing," In Proceedings of International Conference on Distributed Computing Systems, March. 2004.

[25] Y. Zhang, X. Li and Z. Han, "Third Party Auditing for Service Assurance in Cloud Computing," In Proceedings of EEE Global Communications Conference, December. 2017.

[26] MHR Khouzani, Viet Pham, and Carlos Cid, "Incentive Engineering for Outsourced Computing in the Face of Collusion," In WEIS 2014.