Received 29 September 2022, accepted 4 October 2022, date of publication 6 October 2022,
date of current version 12 October 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3212726

RESEARCH ARTICLE

A Fair, Verifiable and Privacy-Protecting Data Outsourcing Transaction Scheme Based on Smart Contracts

LINA LI[1], TINGTING ZHANG[2], GUODONG SUN[3], DEZHENG JIN[2], AND NIANFENG LI[1]
[1]College of Computer Science and Technology, Changchun University, Changchun 130022, China
[2]College of Cyber Security, Changchun University, Changchun 130022, China
[3]Department of Internet-of-Things, Beijing Forestry University, Beijing 100083, China

Corresponding author: Nianfeng Li (linf@ccu.edu.cn)

This work was supported in part by the Natural Science Foundation of Jilin Province, China, under Grant YDZJ202101ZYTS191; in part by the Jilin Scientific and Technological Development Program under Grant 20210201083GX; and in part by the Industry Research Innovation Fund of the University, China, under Grant 2020HYB03002.

ABSTRACT The continuous production of large-scale data makes data outsourcing computation a trend. In order to ensure that data outsourcing transaction is trusted and fair, it needs the supervision and judgment of a third party. However, the third-party intermediary increases the transaction cost, and there is also subjective unreliability. Smart contracts allow trusted transactions without a third party, which are automatic, traceable and irreversible. Therefore, in this paper, we propose a framework for data outsourcing computation transaction based on the smart contract, in which the transaction is verifiable, fair and privacy protected. In this framework, by improved the replication-based verifiable computation technology, the data in the dataset is evenly distributed to each server, and at least one data in the sub datasets of the adjacent servers is the same, only the results from the duplicate data are verified, so as to implement the verifiable transaction with the low transaction cost. Meanwhile, a punishment mechanism is adopted to solve the fairness of the transaction in the smart contract. In addition, the oblivious transfer protocol is used to implement the privacy protection of transaction data. We deploy the data outsourcing computation transaction framework in the simulation environment of the Ethereum blockchain, and the experimental results show that our proposed scheme is effective and has low overhead. Specifically, the data returned by the server is verifiable, and the data privacy of the client is protected. When the client and servers are dishonest, they will be punished so that the transaction is fair. Moreover, the cost of using the smart contract in transactions is almost negligible.

INDEX TERMS Blockchain, smart contract, data outsourcing computation, replication-based verifiability, oblivious transfer protocol, privacy protection.

I. INTRODUCTION

With the arrival of the big data era, data is growing explosively, and the demand for data analysis and data mining is becoming more and more common. For large-scale data, the data outsourcing computation becomes very important. At present, outsourcing non-core IT business and its business processes to professional service providers has become the first choice for enterprise users to improve their business professionalism [1], [2]. In the data outsourcing computation transaction, there are mutual distrust problems between the transaction parties, including the return of wrong results by the service provider and the non-payment of enterprise customers to the service provider, which affect the universality of the data outsourcing computation [3]. In order to solve the above problem, people restrict the behavior of both parties to the transaction by concluding a contract agreement. Usually, a third party is introduced to supervise the execution of the contract. However, this mode increases the transaction cost, and the process is complex and time-consuming when resolving disputes. In addition, in the settlement of disputes, there is the subjectivity of the third party, which leads to the

The associate editor coordinating the review of this manuscript and approving it for publication was Rahim Rahmani.

unfairness and injustice of the judgment results. Meanwhile, there is a problem of data privacy disclosure in the transaction [4]. Therefore, in the data outsourcing computation, it has become a top priority to make both parties compute honestly and return the correct results while ensuring the privacy of all data in the transaction.

Blockchain technology is developing rapidly nowadays, which is a distributed shared ledger technology, and has the characteristics of decentralization, tamper-proof, and traceability [5], [6]. These characteristics provide thoughts for solving problems in data outsourcing computation transactions. In the case of false results, the client must verify the correctness of the computation output of the server. Clients can verify in two ways: (1) the proof-based verifiable computation and (2) the replication-based verifiable computation [7]. In the first way, the server sends the correct output, and the client verifies the results and accepts the output if it is correct. In the second way, the client outsources the same task to multiple clouds and compares the results to obtain the correct output. For the client, the goal is to get the right results while minimizing the cost. The existing proof-based verifiable computation technologies are all based on cryptography and then result in a high overhead. For example, Zhang et al. [8] adopted proof-based verifiability, in which the collision resistance of hash functions and the validity of elliptic curve digital signature algorithm (ECDSA) are used for verification. Cloud computing is based on a pay-per-use model, where servers charge for the resources they use. The proof-based verifiable computational tasks mean that the client must pay for the overhead imposed by the encryption algorithm or protocol. The replication-based verifiable computation avoids complex cryptographic protocols and only needs to verify whether the data results are consistent. However, the replication-based verifiable method increases the repeatability. For example, the researchers [9] proposed the verifiable computation based on multi-worker replication, which means the cost is increased by at least three times. Although Dong et al. [3] proposed a game-theoretic proof to distribute data to two workers for verifiable computation, there are also some repeated costs. So we want to get the right guarantee at a low cost in this paper.

The smart contract is based on blockchain technology and can automatically execute some pre-defined rules and terms [10]. Currently, researchers are trying to use smart contracts to solve various types of problems, and apply them in various fields, such as insurance [11], healthcare [12], electronic voting [13], cloud computing [14] and Internet of Things [15]. For third-party supervision in contracts, the smart contract is a programmable computer program. The characteristic of the smart contract is that, once it is deployed to the blockchain, it can only be accessed and cannot be changed [16]. Smart contracts can be executed automatically without a third party. The most important aspect of data is data privacy [17]. Researchers have proposed a data transaction model based on oblivious transfer protocol. The oblivious transfer technology can protect the privacy of buyers and sellers during the transaction process [18]. In this paper, we solve the trust and privacy issues in data outsourcing computation based on smart contracts.

To solve the above problems, this paper proposes a data outsourcing computation transaction scheme with verifiable, fair and privacy protection. In this scheme, a data outsourcing computation transaction framework is established based on blockchain. This framework consists of computational verification, reward and punishment mechanism, and privacy protection. The smart contract is the core of outsourcing computation transaction, which meets the transaction requirements together with oblivious transfer protocol and replication-based verifiable technology. In the computing verification stage, the transaction data is first distributed to multiple servers, and one of the allocated data of two adjacent servers is the same, so as to reduce the cost while the data results are verifiable. Then, the data of each transaction is recorded in the blockchain and used as evidence for dispute resolution. Moreover, in order to implement the fairness of data computation transactions, a punishment mechanism is established. The smart contract will punish the fraudster by confiscating the deposit to protect the interests of the other party, so that the data outsourcing computing transaction is fair. In the privacy protection mechanism, the transaction data is stored in multiple servers. Each server only knows part of the data it receives, but knows nothing about the rest of the data. The main contributions of this paper are as follows.

(1) Aiming at the problems of distrust, unfairness and privacy leakage in the process of data transaction, we propose a data outsourcing computation transaction scheme based on the smart contract. Both sides of the transaction can complete transactions efficiently, fairly and autonomously without involving any third-party entity supervision.

(2) In order to protect the privacy of the data, we divide the transaction data into multiple sub datasets and allocate them to different servers. The data set is encrypted based on the oblivious transfer protocol, and each server only knows part of the data computed by itself, but does not know the entire transaction data, so as to ensure the confidentiality and privacy of the data.

(3) For the result verification of transaction data, the replication-based verifiable technology is improved. By allocating one identical data (unknown to other servers) to two adjacent servers, the honesty of the server can be verified when these data computation results are returned, thus the cost waste caused by duplication can be reduced.

(4) A penalty mechanism is used in the smart contract for the fairness of outsourcing computation. The client and servers must store their deposit before the transaction starts. When one party is dishonest, the deposit will be deducted to protect the interests of the other party.

This paper is organized as follows. Section II reviews existing work on outsourcing computation techniques, section III introduces definitions and models, section IV describes our transaction scheme, section V describes experiments and related results, and the work of this paper is summarized and the future work is prospected in Section VI.

II. RELATED WORKS

In this section, we discuss the research related to the blockchain and the smart contract in the field of data computing transactions, and we also elucidate the related work of replication-based verifiable technology and oblivious transfer technology.

A. BLOCKCHAIN AND SMART CONTRACTS

Blockchain is widely used in the field of data transactions and the exchange of physical objects [19], [20]. With the prevalence of blockchain in data sharing, some problems have emerged, such as fairness of transactions and data copyright protection [21], [22]. Xiang et al. [23] proposed a flexible Ethereum-based scheme to automatically guarantee the fairness of data sharing and proactively protect data copyright by using smart contracts. Reniers et al. [24] recorded and tracked the process of file registration and access request by smart contracts. When one party raised a dispute, the dishonest party can be identified through the smart contract. Liu et al. [25] and Xiong and Xiong [26] both conducted data transactions based on the smart contract. Liu used punishment to hold violators accountable through the votes of external auditors, while Wei introduced deep learning, and uses similarity learning to deal with disputes about data availability in data transactions. In order to incentivize data sharing, Xuan et al. [27] proposed the smart contract data sharing incentive model based on evolutionary game theory. The smart contract mechanism can dynamically and collaboratively improve the incentive parameters and continuously encourage users to participate in the data sharing program. Wang et al. [28] proposed an auditable fair payment and physical asset delivery protocol based on the smart contract, which enables reliable and fair payments among merchants, consumers and logistics companies.

In addition to the application in data transactions, Blockchain is also widely used in data outsourcing computation to reduce the transaction cost. Wang et al. [29] designed a blockchain-based public cloud storage audit fair payment smart contract, which proposed a concept of non-interactive provable data ownership, thereby the data is effectively transacted. In outsourcing computation, there is also the trust problem between outsourcers and workers. Chen et al. [30] introduced a third-party trust problem in the outsourcing computation model, and proposed a new fair-condition payment scheme to solve the trust problem. Carbunar and Tripunitara [31] proposed a unified trust framework, relying on offline bank generation and redemption payment mechanisms, the correct participation will be economically rewarded, while outsourcing computation is efficiently validated and effectively rewarded. Kupcu [32] combined cryptography with elements of game theory and mechanism design, and proposed to increase rewards to the honest cloud based on blockchain to help detect cheating clouds. The fairness of transactions is also crucial in the process of computing transactions. Lin et al. [33] proposed an optimized blockchain-based fair payment (OBFP) system model for outsourcing computations, and addressed fairness and privacy issues with blockchain-based zero-knowledge proof technology. Similarly, Zhang et al. [34] also introduced a blockchain-based fair payment framework for cloud computing outsourcing service BPay.

B. OBLIVIOUS TRANSFER PROTOCOL

Oblivious transfer protocol is a privacy-protected communication protocol between two parties. The receiver's privacy is not known by the sender, Oblivious transfer protocol allows the communicating parties to transmit messages in an optically ambiguous manner. The traditional protocol is 1-of-2 oblivious transfer protocol of Diffie-Hellman key exchange, Jain and Hari [35], Lou and Huang [36] and Mu et al. [37] have proposed a new effective k-out-of-n oblivious transfer protocol based on this, where the receiver can only receive k messages out of n messages by the senders, and the senders did not know which messages were received. Chu and Tzeng [38] and Lai et al. [39] improved this by using an effective two-round k-out-of-n oblivious transfer scheme, in which the receiver sent O(k) messages to the sender, and the sender sent O(n) messages to the receiver. Esmaeilzade et al. [40] adopted the concept of asymmetric homomorphic encryption and proposed a general structure to build a simple and efficient oblivious transfer protocol.

Oblivious transfer protocol can be used in privacy protection. Using the concept of blockchain technology, Yang et al. [41] adopted a privacy protection data transmission scheme and proposed a novel OTmn protocol to support secure ciphertext conversion, malicious user identification, and two-way privacy for both communicating parties. For data sharing, Shen et al. [42] proposed a privacy-preserving and untraceable scheme by using proxy re-encryption and oblivious random access memory. Privacy protection also applies to digital rights and verification situations. Jiang and Bo [4] proposed a privacy-preserving digital rights management protocol based on oblivious transfer theory, in which the license server can determine the number of licenses, but not guarantee the user's choice accurately. Kak [43] verified the correctness of the program without revealing the random numbers used by both parties through the oblivious transfer protocol. Whereas Damodaran and Rial [44] protected the privacy of access to the database while enforcing access control policies by using oblivious transfer.

C. REPLICATION-BASED VERIFIABLE COMPUTATION TECHNOLOGY

The traditional verifiable computations are proof-based and replication-based verifiable computation, respectively, which were described in depth by Dorsala et al. [9]. Kumaresan and Bentov [45] and Simunic [46] used a verifiable computation to implement currency transactions. Since the proof-based verifiable computation uses an encryption algorithm, the cost is large [47]. Therefore, replication-based verifiable computation is generally used. Replication-based verifiable computation technology mainly sends the task to multiple people. Both Canetti et al. [48] and Avizheh et al. [49] delegated the computation to multiple servers, and used the designed protocol to compare the results for verifiability to guarantee to obtain the correct answer. If any inconsistency was detected, a cheating cloud would be found after a round of delegation of the two clouds. For replication verification, both Belenkiy et al. [50] and Dong et al. [3] established a game theoretic framework, in which a reward and penalty mechanism based on random review strategies and hired multiple contractors was designed to perform the work for verification to keep rational clouds from colluding and cheating.

The above solutions for fairness, privacy, and verifiable implementation of data outsourcing computation are all either too costly or not fully realized. In this paper, our work is to achieve fairness, privacy, and verifiability of data computations by using smart contracts and oblivious transfer protocol and replication-based verifiable technology.

III. DEFINITIONS, MODELS AND GOALS

In this section, we will briefly describe the technology, the transaction and adversary models used, as well as the design goals.

A. REPLICATION-BASED VERIFIABLE COMPUTATION

Replication-based verifiable computation means assigning the same outsourcing computation to multiple servers instead of a single server, and verifies the correctness of the computation by comparing the returned computation results. In this paper, the computing data is divided into n sub datasets on average, and the adjacent sub datasets contain at least one same computing data. Different sub datasets are outsourced to multiple servers, and only the computation results of servers with the same data are compared and verified, and the servers that submit the correct results can get paid.

Definition 1. The replication-based verifiable computation involves one client and multiple servers as follows.

• $F(x)$: The client transmits the sub dataset x and the computation function $F(x)$ to the corresponding server.

• Compute$(F, x) \rightarrow y_i$. Each server receives F and x, and computes $F(x)$ and outputs $y_i = F(x)$, $1 \leq i \leq n$.

• Verify$(Sy_i, Sy_{i+1}) \rightarrow \{0, 1\}$. Compare the computation results of the same data in two adjacent servers, output 1 if and only if they are equal, otherwise, output 0.

Definition 2. For fair verifiable computation between two parties, the server and the client must provide the following guarantees:

• Fast Verification: The output of the work performed to verify correctness is less than the computational effort.

• Get reward: the server receives a reward from the client if and only if the server submits the correct computation output to the client.

B. TRANSACTION MODEL

In this paper, we propose a data outsourcing computation transaction model, as shown in Figure 1. In this model, four parts are mainly involved: users, blockchain, IPFS and the smart contract. Users include one client and n servers. The client owns the transaction data and wants to get the computation results of the data. The server side is responsible for computing the data and finally getting paid. Blockchain is the infrastructure for the data transaction, which is used to deploy the smart contract of data outsourcing computation transactions and support users to join and invoke the smart contract. The distributed file storage system IPFS is adopted to store the outsourcing transaction data. The client uploads the data to be computed and the computation function to IPFS, and obtains the hash address corresponding to the data. The address is encrypted and transmitted to the server through the smart contract. The server decrypts the data address, and then downloads the data from IPFS. The Smart contract is the core of the whole transaction model. At the beginning of the transaction, the client and server store a deposit in the smart contract. When there is a transaction problem, the punishment mechanism will be triggered to support the fairness of the transaction. In the process of transaction, the smart contract protects the privacy of data through the oblivious transfer protocol, and the correctness of the data is judged by the improved replication-based verifiable computation. At the end of the transaction, the smart contract ensures that the client gets the computation results and the server gets the rewards, and the whole transaction process is traceable.

C. ADVERSARY MODEL

In the data outsourcing computation transaction, there are some potential threats on both sides of the transaction. We classify these threats into the following three categories and adopt different strategies to solve them in the adversary model.

1. The client refuses to pay after receiving the data results. This will cause the server to not get the corresponding reward after completing the computation. For this behavior, the client will be required to store a deposit not less than the reward in the blockchain before the transaction starts. When the server refuses to pay, his deposit will be confiscated as the reward.

2. The server returns the wrong results or does not return the results. For this behavior, the server will be asked to store the deposit before launching the transaction. When the server returns the results, the improved replication-based verification computation method is used to determine whether it is correct. If the server returns the wrong results or does not return the results, the deposit will be confiscated.

3. The client and server tamper with the data during the transaction. Any party modifying the data without authorization will lead to transaction disputes. For this phenomenon, IPFS is used to store the transaction data. Once the data is

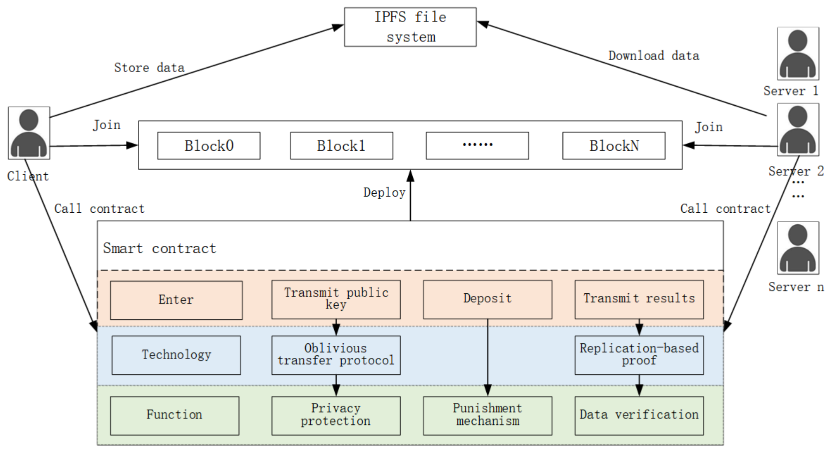


FIGURE 1. **The transaction model of the data outsourcing computation.**

modified, the corresponding data storage address will change. Since the transaction records are stored in the blockchain, one can find out which party has changed the data by viewing and comparing the data addresses.

D. DESIGN GOALS

In this paper, the design goals of the data outsourcing computation transaction scheme are as follows.

1. Fairness. After the transaction is completed normally, the client will get the data results, and the servers will get the corresponding reward. If there is any problem in the middle, the smart contract will deduct their deposit.

2. Autonomy. Transactions are executed automatically through the smart contract without the help and arbitration of any third-party entity supervision.

3. Privacy protection. In the transaction process, each server only knew the data of their own subtask set, but did not know others and the complete data.

4. Time constraints. All transaction transactions must be completed within the specified time.

5. Verifiability. After the transaction results are returned from the servers, the results will be verified, and the transaction can end only after all servers are honest.

IV. DATA OUTSOURCING COMPUTATION TRANSACTION SCHEME

In this paper, we propose a scheme of data outsourcing computation transaction based on the smart contract, which can ensure the verifiability, fairness and privacy protection of the transaction. In this scheme, the improved replication-based verifiable computation, the punishment mechanism, and the oblivious transfer protocol are combined with the smart contract, and executed automatically in the smart contract. In order to reduce redundant workload, each server only computes independent sub datasets, and adjacent servers only add an identical data, and verify the correctness of the computation results through the improved replication-based verifiable computation technology. Meanwhile, the punishment mechanism is used to achieve the fairness of the transaction, and the oblivious transfer protocol for protecting the data privacy. Next, we will describe the implementation details and process of the whole scheme. The main notations used in the proposed scheme are listed in Table 1.

TABLE 1. **The main notations used in the proposed scheme.**

Notations	Description
$Data$	The transaction data
x	The sub dataset data
$F(x)$	The computation function for data
Y	The server's data computation results
P	The remuneration amount for data
C	The compensation amount for data
n	The number of servers
PK_C, PK_S	The client and server's public keys
SK_C, PK_S	The client and server's private keys
D_C,D_S	The client and server's deposits
S_R	The computation value of the same data
M_R	The comparison result of the computation values
C_R	The verification result of the server's honesty

A. DATA TRANSFER PROCESS

In the data outsourcing computation transaction, the client first divides the data into n sub datasets, and the adjacent servers at least one same data in the sub dataset. Then, the transaction data will be transmitted through IPFS. The data transmission process is shown in Figure 2, and the processing steps are as follows.

1. Client transfers the data to be computed and the data computation function into the IPFS system.

2. Client obtains the hash address corresponding to the data from IPFS.

3. Client encrypts data through oblivious transfer protocol and uploads the data to the blockchain.

4. Server gets the encrypted address from the blockchain and decrypts the data to obtain the Hash address.

5. Server requests the data corresponding to the hash address from IPFS.

6. Server downloads the data uploaded by the client from IPFS.

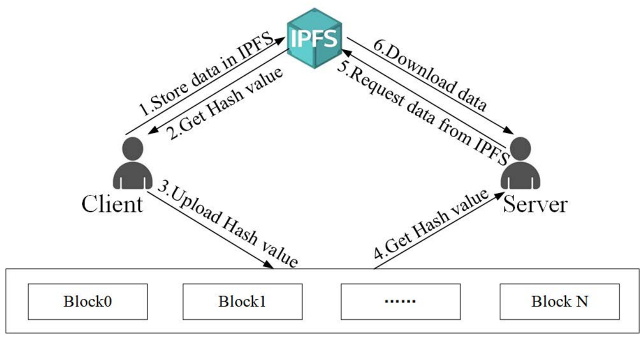


FIGURE 2. The data transfer process between the client and the server through IPFS and blockchain.

B. DATA OUTSOURCING COMPUTATION PROCESS

Based on the smart contract, the main process of data outsourcing computation transaction is shown in Figure 3.

1. The client sets the computation award and the amount of data to be computed by the server, and then makes the deposit setting for the client and servers.

2. The client stores the deposit into the smart contract according to the set deposit amount. If the deposit amount is not greater than the remuneration, the smart contract cannot be executed. After storing the deposit, the information set in the transaction is published in the blockchain.

3. The server intending to conduct data outsourcing transactions joins the blockchain and calls the smart contract, and sends the deposit to the smart contract.

4. The client generates n pairs of public and private keys, and sends the public key to the server through the smart contract.

5. The server selects a random number, encrypts the random number with the received public key, and sends the encrypted data to the smart contract.

6. The client decrypts with the private key, encrypts the data address with the random number, and then transmits it.

7. After receiving the encryption result, the server decrypts it with its own random number, obtains the data address, downloads the data and performs computation, and returns the computing results.

8. The client verifies the computation results through the smart contract. If the result is correct, the transaction is completed, and the client pays a fee to the server and the smart contract returns the deposit. Otherwise, the transaction is abnormal, and the contract will deduct the corresponding deposit and starts a new transaction.

The whole data outsourcing computation transaction process is mainly described in Algorithm 1, which includes the transaction data generation, data transmission, data computing, computation result verification, remuneration reward, deposit deduction. When the transaction fails, the smart contract should not only deduct the deposit from the server, but also restart a new round of transaction, that is, repeat Algorithm 1 until the transaction succeeds.

Algorithm 1 Data Outsourcing Computation Transaction Algorithm (DOCT)

```
Input: $Data$, $D\_C$, $D\_S$
Output: The transaction results (Success or failure)
1:  The client joins the blockchain, generates the transaction subtask sets $x$ from $Data$ and their corresponding $F(x)$, and stores $x$ and $F(x)$ in IPFS.
2:  The client sets the transaction data values, and the deposit threshold $\theta$ for the client and servers, and guarantees that $\theta$ is greater than $D\_C$ and $D\_S$.
3:  The client sends the deposit to the smart contract.
4:  The server joins the blockchain, and sends the deposit to the smart contract.
5:  The client generates $n$ pairs of public and private keys, PK_S and SK_S, and sends the public keys to each server.
6:  The server selects the public key to encrypt the random number, then sends the encrypted result to the smart contract.
7:  The client uses private key to decrypt the encrypted data, and uses the random number to encrypt the Hash address, then sends it to the smart contract.
8:  The server uses the random number to decrypt and get the Hash address to download the data from IPFS, and returns the computation results $Y$ and $S\_R$ to the client.
9:  The client calls the result validation algorithm (Algorithm 3) with $S\_R$ as parameters, and assigns the return value to $C\_R$.
10: if $C\_R ==$ True then
11:     The client pays the server award and returns deposit.
12:     return The transaction is successful.
13: else
14:     The client deducts the deposit and will start a new transaction.
15:     return The transaction failed.
```

C. TIME CONSTRAINTS ON TRANSACTIONS

In order to achieve a normal and efficient fair transaction, both parties reached a consensus and set a series of time limits in the smart contract. Some transaction operations must be completed within the specified time limit, otherwise the transaction will be terminated. The specific time constraints are as follows.

T_0: During the transaction process, it is necessary that the client sets the deposit amount of the client and servers, the computation reward and the computing amount of the server, stores the transaction sub datasets and the computation functions to IPFS, and stores their deposit in the smart contract before time T_0. Otherwise, the transaction will be terminated and the deposit will be returned to the client. At the same time, relevant transaction information is published in the blockchain so that interested servers can join the transaction.

T_1: After the server joins the blockchain and invokes the smart contract, it must store the deposit before T_1 and the deposit is not lower than the specified threshold. If any

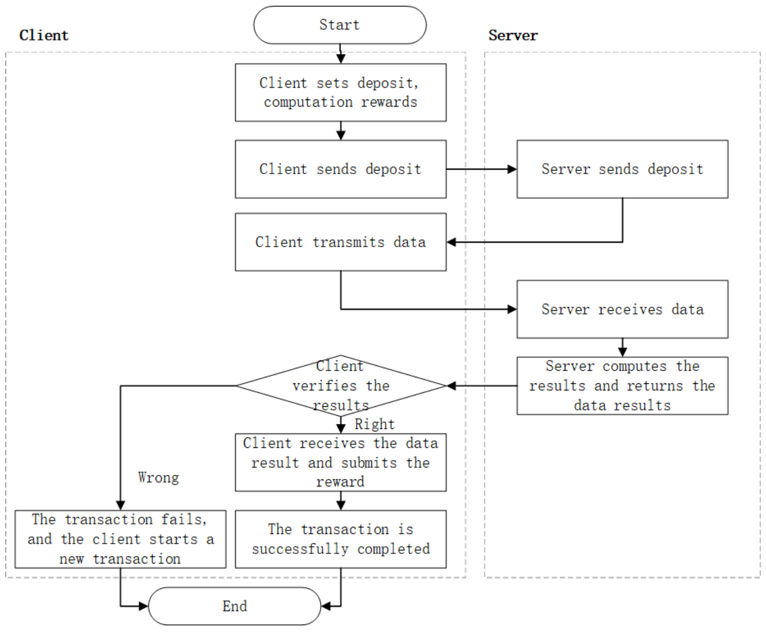

Client
Start
Server
Client sets deposit, computation rewards
Client sends deposit
Server sends deposit
Client transmits data
Server receives data
Client verifies the results
Server computes the results and returns the data results
Right
Client receives the data result and submits the reward
Wrong
The transaction fails, and the client starts a new transaction
The transaction is successfully completed
End

FIGURE 3. The main process of data outsourcing computation transaction.

server does not store the deposit or the deposit is insufficient, the transaction will be terminated, the corresponding deposit will be refunded, and the new server will join the transaction.

T_2: Once the deposit of all servers is stored in the smart contract, the client must generate n pairs of public and private keys within T_2, and submit all public keys to the smart contract. Otherwise, the transaction will be terminated.

T_3: After receiving the public key from the smart contract, the server must select a random number within T_3, then encrypt it with the public key and submit it to the smart contract. Otherwise, the transaction will be terminated.

T_4: When the client receives the encrypted data, it must decrypt it with the private key within T_4, then encrypt the data address with the obtained random number, and transmit it to the smart contract.

T_5: After the server obtains the data encryption result and decrypts it, it obtains the data through IPFS and performs computation. The computing results must be transmitted back to the client within T_5. If the results are transmitted after T_5, the server is considered dishonest and its output is set to null. The returned results are verified by the client. If the computation results are correct, the server is honest. Otherwise, it is regarded as dishonest.

T_6: When the results of all servers are verified, the client must deliver the reward to the smart contract within T_6, and each honest server will receive the reward. If the client fails to pay or the reward delivered is insufficient, the deposit is deducted as the reward and the deposit of the server is refunded. Meanwhile, the dishonest server is deducted the deposit, and the computing task is outsourced to a new server.

T_7: We set the buffer time to T_7, so that all servers have enough time to receive rewards, and return the deposit to the client and servers. After T_7, the entire transaction must be successfully completed, and the client and servers exit from the transaction, or the transaction fails, and the client starts the next transaction.

D. SMART CONTRACT ANALYSIS

The data outsourcing computation transaction is implemented through the smart contract. The interaction between the server and the client in the smart contract is shown in Figure 4, which mainly completes data encryption transmission, data computation result return, server reward acquisition, deposit return and other operations. In this paper, the penalty mechanism, the data verification algorithm and the oblivious transfer protocol are adopted in the smart contract to achieve fairness, verifiability and privacy protection of transactions.

1) PENALTY MECHANISM

In order to ensure the fairness of the data outsourcing computation, the smart contract introduces a penalty mechanism. Before the transaction starts, the client will store the deposit not less than the remuneration into the smart contract, and the server will store the deposit not less than the payable amount into the smart contract. When the client fails to pay or the reward is insufficient, the penalty mechanism will be triggered to deduct its deposit as the reward to the server and return the deposit to the server. When the server does not return data or the data verification is inconsistent, it is determined that the server is dishonest, and the penalty mechanism is triggered to deduct the deposit of the server and return the client's deposit. In algorithm 2, we implement the client and server penalty mechanism.

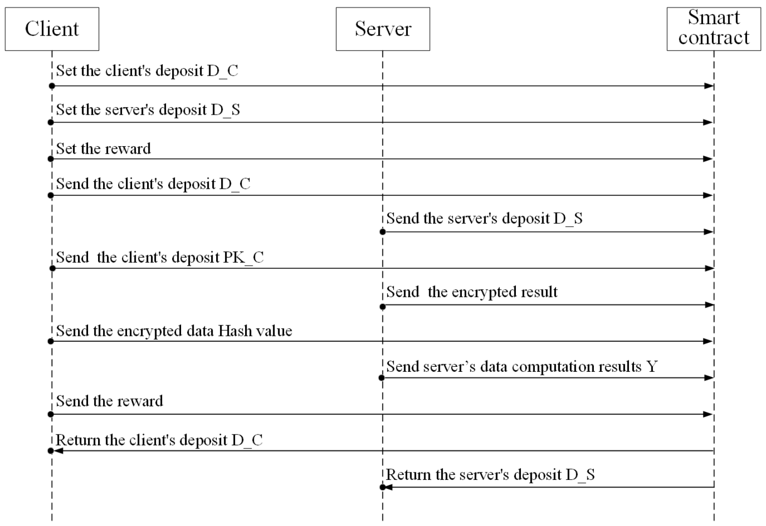

Client
Server
Smart contract
Set the client's deposit D_C
Set the server's deposit D_S
Set the reward
Send the client's deposit D_C
Send the server's deposit D_S
Send the client's deposit PK_C
Send the encrypted result
Send the encrypted data Hash value
Send server's data computation results Y
Send the reward
Return the client's deposit D_C
Return the server's deposit D_S

FIGURE 4. The interaction between the server and the client based on the smart contract.

Algorithm 2 Client and Server Penalty Algorithm (CSP)

```
Input: $User$, $D\_C$, $D\_S$
Output: $D\_C$, $D\_S$
1: if $User$ == Client then
2:     $D\_C = D\_C$-$P$ // Deduct the remuneration from the client's deposit as the reward of the server.
3:     $D\_S = 0$ // Return the deposit of the server.
4: else
5:     $D\_S = D\_S$-$C$ // Deduct the deposit from the server as compensation to the client.
6:     $D\_C = 0$ // Return the deposit to the client.
7: return $D\_C$, $D\_S$
```

2) RESULT VERIFICATION

In this section, we propose a computation result verification algorithm based on the improved replication-based verifiable technology, which is used to judge the correctness of the returned results of the server, so as to determine whether the server is honest. In the improved replication-based verifiable technology, in order to reduce the cost of repeated computation, we first divide the outsourcing computation data into n sub datasets, $n > 2$, and at least one data in the adjacent sub datasets is the same. Then, each sub dataset is assigned to one server. For the result data returned by each server i, it is only necessary to compare its computation results with that of the same data in adjacent servers $i-1$ and $i+1$, determine the server is honest or not by comparing the results. Each server has only two states: honest or dishonest. We assume that if the computation results of two servers are the same, they are both honest. Otherwise, one party is dishonest or both are dishonest. Therefore, the comparison result matrix between adjacent servers is shown in Table 2.

Based on the above matrix, we design a computation result verification algorithm. The specific verification process is described in Algorithm 3. First, we compute and record the comparison results of any two adjacent servers. If the results are the same, 1 is returned, otherwise, 0 is returned (lines 1-10). Then, according to the above comparison results, judge the honesty of all servers in a sequential cycle from the second server to the first server (lines 11-25). The judgment rules are as follows: (1) if the current value is 1 and the previous value is 1, the servers in the current and previous locations are honest; (2) If the current value is 1 and the previous value is 0, the server in the current location is honest and the server in the previous location is dishonest; (3) If the current value is 0 and the previous value is 1, the server at the current location is honest and the server at the next location is dishonest; (4) If the current value is 0 and the previous value is 0, the servers in the current, previous and next locations are all dishonest. At the end of this round, the computation tasks of the dishonest servers are assigned to the new servers for recomputation, and a new round of transactions is started.

TABLE 2. The comparison result matrix of adjacent servers.

	S_R_i =S_R_{i-1}	S_$R_i \neq$ S_R_{i-1}
S_R_i =S_R_{i+1}	server i-1 is honest server i is honest server i+1 is honest	server i-1 is dishonest server i is honest server i+1 is honest
S_$R_i \neq$ S_R_{i+1}	server i-1 is honest server i is honest server i+1 is dishonest	server i-1 is dishonest server i is dishonest server i+1 is dishonest

Complexity analysis. In algorithm 3, the result data similarity judgment compares the two adjacent results in turn, and the last result is compared with the first result. Then, the comparison results are stored in the array. This process takes O(n) time in total. Next, it takes O(n) time to traverse all the comparison results and judge the honesty of each server. The above two parts are executed independently. Therefore, the total time cost of the data validation algorithm is O(n). Since n is the number of servers, it is limited. Therefore, the cost of data verification algorithm is very small.

Algorithm 3 Computation Results Validation Algorithm (CRV)

```
Input: $S\_R_i(1 \le i \le n)$
Output: $C\_R_i$
1:  for $i = 1, \dots, n$ do
2:      $j = i \bmod n + 1$
3:      if $i == n$ then
4:          $k = 0$
5:      else
6:          $k = i$
7:      if $(S\_R_i == S\_R_j)$ then
8:          $M\_R_k = 1$
9:      else
10:         $M\_R_k = 0$
11: for $i = 1, \dots, n$ do
12:     if $i == 1$ then
13:         $k=n$
14:     else
15:         $k = i - 1$
16:     if $(M\_R_i == 1$ and $M\_R_{i-1} == 1)$ then
17:         $C\_R_k = 1, C\_R_i = 1, C\_R_{i+1} = 1$
18:     else
19:         if $(M\_R_i == 1$ and $M\_R_{i-1} == 0)$ then
20:             $C\_R_k = 0, C\_R_i = 1, C\_R_{i+1} = 1$
21:         else
22:             if $(M\_R_i == 0$ and $M\_R_{i-1} == 1)$ then
23:                 $C\_R_k = 1, C\_R_i = 1, C\_R_{i+1} = 0$
24:             else
25:                 $C\_R_k = 0, C\_R_i = 0, C\_R_{i+1} = 0$
26: return $C\_R$
```

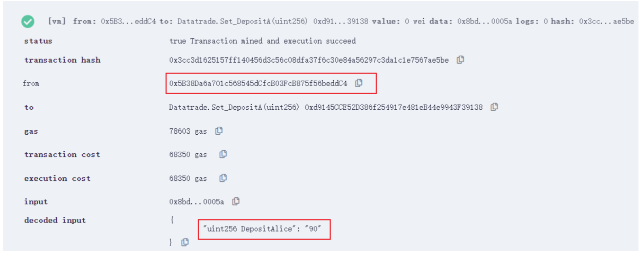
3) PRIVACY PROTECTION

In the data outsourcing computation transaction, we introduce the oblivious transfer protocol into the smart contract to protect the data privacy. Based on the oblivious transfer protocol, the server only knows the data it is responsible for computing, but cannot know the remaining data processed by other servers and the data repeated with adjacent servers, so as to achieve the privacy of the data. The traditional oblivious transfer protocol is mainly 1-2 protocol. That is, suppose A has two pieces of information, and B wants to select one. After the transmission through the oblivious protocol, B gets the piece of information he wants, but does not know the other piece of information, and A does not know which piece of information B selects. In this paper, the oblivious transfer protocol is improved, and the transaction confidentiality is realized by using 1-n oblivious transfer protocol. Compared with the classical 1-2 oblivious transfer protocol, 1-n oblivious transfer protocol is more secure, and the computational complexity is not high, which can improve the security of privacy protection.

The transaction flow of the 1-n oblivious transfer protocol is as follows. First, the client generates n pairs of public keys and private keys, and sends the public keys to the corresponding servers. After the server obtains the public key, it generates a random number, encrypts the random number with the public key, and then sends the encryption result to the client. After obtaining the encrypted data, the client decrypts the private key to obtain the random number. Then, the client encrypts the Hash value of the data address with a random number and sends the encryption result to the server. Finally, the server decrypts the ciphertext with random numbers to obtain the corresponding Hash value.

TABLE 3. Account addresses of all users.

User	Account address
Client(Alice)	0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
Server(Bob)	0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2
Server(Cindy)	0x4B20993Bc481177ec7E8f571ceCaE8A9e22C02db
Server(David)	0x78731D3Ca6b7E34aC0F824c42a7cC18A495cabaB



```
[vm] from: 0x5B3...eddC4 to: Datatrade.Set_DepositA(uint256) 0xd91...39138 value: 0 wei data: 0x8bd...0005a logs: 0 hash: 0x3cc...ae5be
status              true Transaction mined and execution succeed
transaction hash    0x3cc3d1625157ff140456d3c56c08dfa37f6c30e84a56297c3da1c1e7567ae5be
from                0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
to                  Datatrade.Set_DepositA(uint256) 0xd9145CCE52D386f254917e481eB44e9943F39138
gas                 78603 gas
transaction cost    68350 gas
execution cost      68350 gas
input               0x8bd...0005a
decoded input       {
                        "uint256 DepositAlice": "90"
                    }
```

FIGURE 5. The client and server deposit settings.

V. EXPERIMENTS EVALUATION

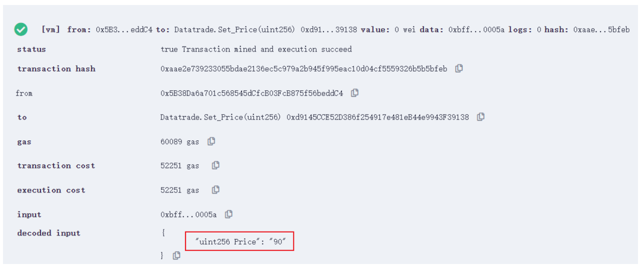
A. EXPERIMENTAL SETUP AND TESTING

In this paper, we have implemented a smart contract for data outsourcing computation transactions and verified the transaction scheme on the website "http://remix.ethereum.org/". Remix adopts the solidity language to build smart contracts, providing a test environment that supports the compilation, verification and deployment of any simulated Ethereum smart contracts. Solidity has built-in data types, which are used to build all the functions of smart contracts. Based on Remix and solidity, we implemented the punishment mechanism by building the deposit and return deposit functions, the oblivious transfer by sending the public key function and the encrypted data function, and the replication-based verifiability by the data verification function.

We use one client (Alice) and three servers (Bob, Cindy, and David) as test cases and set the Ethereum addresses of the client and servers, as shown in Table 3. All functional tests are executed by entities with specific Ethereum addresses. First, before launching the transaction, the client's deposit must be set to be no less than the reward return (In this paper, we assume 90wei), and the server deposit must be set to be no less than the due return, as shown in Figure 5.

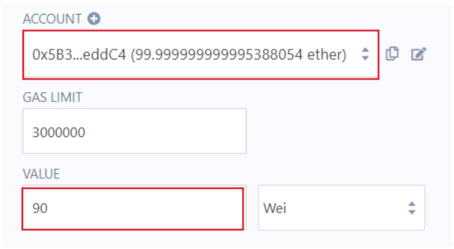
Alice must set the price and the number of blocks. As shown in Figure 6, the interested servers can join in, including Bob, Cindy, and David.

Then, the server and client must store the corresponding deposit according to the set deposit amount, otherwise the transaction will stop and remind users to store the correct



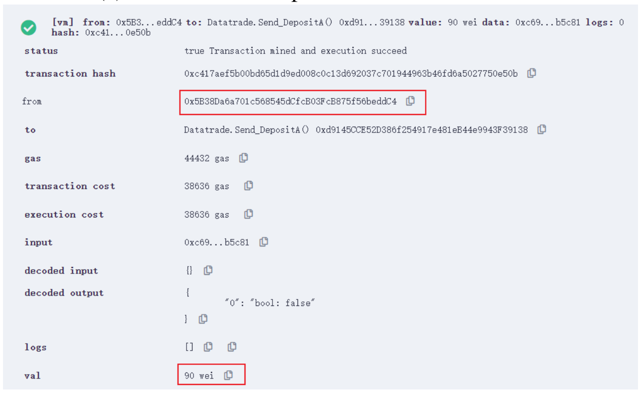
```
[vm] from: 0x5B3...eddC4 to: Datatrade.Set_Price(uint256) 0xd91...39138 value: 0 wei data: 0xbff...0005a logs: 0 hash: 0xaae...5bfeb
status              true Transaction mined and execution succeed
transaction hash    0xaae2e739233055bdae2136ec5c979a2b945f995eac10d04cf5559326b5b5bfeb
from                0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
to                  Datatrade.Set_Price(uint256) 0xd9145CCE52D386f254917e481eB44e9943F39138
gas                 60089 gas
transaction cost    52251 gas
execution cost      52251 gas
input               0xbff...0005a
decoded input       {
                        "uint256 Price": "90"
                    }
```

FIGURE 6. The client sets price and number of blocks.



```
ACCOUNT
0x5B3...eddC4 (99.999999999995388054 ether)
GAS LIMIT
3000000
VALUE
90    Wei
```

(a) Client sets the deposit amount of the client.



```
[vm] from: 0x5B3...eddC4 to: Datatrade.Send_DepositA() 0xd91...39138 value: 90 wei data: 0xc69...b5c81 logs: 0
hash: 0xc41...0e50b
status              true Transaction mined and execution succeed
transaction hash    0xc417aef5b00bd65d1d9ed008c0c13d692037c701944963b46fd6a5027750e50b
from                0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
to                  Datatrade.Send_DepositA() 0xd9145CCE52D386f254917e481eB44e9943F39138
gas                 44432 gas
transaction cost    38636 gas
execution cost      38636 gas
input               0xc69...b5c81
decoded input       {}
decoded output      {
                        "0": "bool: false"
                    }
logs                []
val                 90 wei
```

(b) Client sends the deposit.

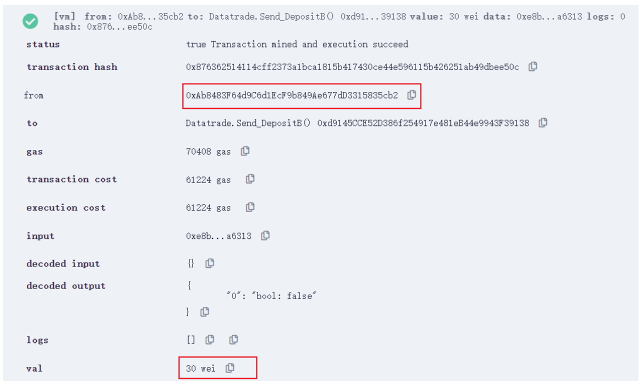
FIGURE 7. The deposit process of the client.

amount. The balance of the client (Ethereum address: 0 × 5B38Da6a701c568545dCfcB03FcB875f56beddC4) before the test is 99.999999999995355054 *ether*, and the deposit amount is set to 90 *wei*, which is deposited into the contract from the client, as shown in Figure 7. Then, the 30 *wei* deposit is deposited into the smart contract from the server Bob (Ethereum address: 0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2), and Figure 8 shows the log triggered by the server depositing the 30 *wei* deposit.

Next, Alice sends the public key of the oblivious transfer protocol to the transaction, and the transaction begins, as shown in Figure 9.

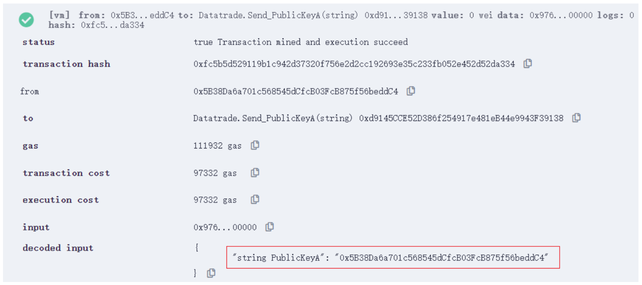
Then, Alice encrypts the data address with the random number returned by the server Bob, and sends the address to the server Bob, as shown in Figure 10.

After receiving the encrypted address, the server Bob decrypts to obtain the data address. Then, Bob obtains data from IPFS according to the address, performs the computation, and then returns the computation result to the smart contract. The smart contract needs to verify the correctness



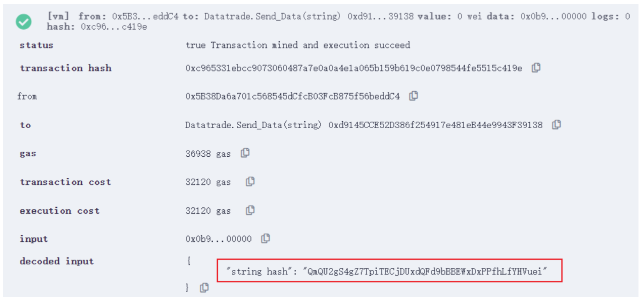
```
[vm] from: 0xAb8...35cb2 to: Datatrade.Send_DepositB() 0xd91...39138 value: 30 wei data: 0xe8b...a6313 logs: 0
hash: 0x876...ee50c
status              true Transaction mined and execution succeed
transaction hash    0x876362514114cff2373a1bca1815b417430ce44e596115b426251ab49dbee50c
from                0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2
to                  Datatrade.Send_DepositB() 0xd9145CCE52D386f254917e481eB44e9943F39138
gas                 70408 gas
transaction cost    61224 gas
execution cost      61224 gas
input               0xe8b...a6313
decoded input       {}
decoded output      {
                        "0": "bool: false"
                    }
logs                []
val                 30 wei
```

FIGURE 8. The deposit process of the server.



```
[vm] from: 0x5B3...eddC4 to: Datatrade.Send_PublicKeyA(string) 0xd91...39138 value: 0 wei data: 0x976...00000 logs: 0
hash: 0xfc5...da334
status              true Transaction mined and execution succeed
transaction hash    0xfc5b5d529119b1c942d37320f756e2d2cc192693e35c233fb052e452d52da334
from                0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
to                  Datatrade.Send_PublicKeyA(string) 0xd9145CCE52D386f254917e481eB44e9943F39138
gas                 111932 gas
transaction cost    97332 gas
execution cost      97332 gas
input               0x976...00000
decoded input       {
                        "string PublicKeyA": "0x5B38Da6a701c568545dCfcB03FcB875f56beddC4"
                    }
```

FIGURE 9. The client sends the public key of the oblivious transfer protocol.



```
[vm] from: 0x5B3...eddC4 to: Datatrade.Send_Data(string) 0xd91...39138 value: 0 wei data: 0x0b9...00000 logs: 0
hash: 0xc96...c419e
status              true Transaction mined and execution succeed
transaction hash    0xc965331ebcc9073060487a7e0a0a4e1a065b159b619c0e0798544fe5515c419e
from                0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
to                  Datatrade.Send_Data(string) 0xd9145CCE52D386f254917e481eB44e9943F39138
gas                 36938 gas
transaction cost    32120 gas
execution cost      32120 gas
input               0x0b9...00000
decoded input       {
                        "string hash": "QmQU2gS4gZ7TpiTECjDUxdQFd9bBBEWxDxPPfhLfYHVuei"
                    }
```

FIGURE 10. The client sends the encrypted data address of the server.

of the results returned by the server Bob. The verification process of the server Bob is shown in Figure 11.

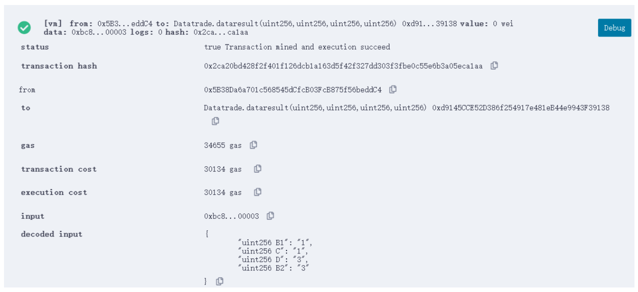
B. EVALUATION ANALYSIS

1) PERFORMANCE ANALYSIS

Privacy. During data transmission, Alice encrypts data with three random numbers only known to Bob, Cindy and David, respectively. Finally, Bob, Cindy and David can only decrypt the data they want to compute and do not know other servers' data, thus implementing the privacy protection of the transaction data.

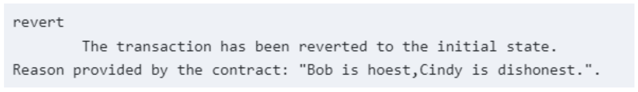
Verifiability. The smart contract compares the return results of the same data among Bob, Cindy and David. When all the comparison results are the same, all servers are honest. When Bob and Cindy have different comparison results, but Cindy and David have the same comparison results, Bob is regarded as dishonest, as shown in Figure 11.

Fairness. Before the transaction starts, both the client and the server store deposits, as shown in Figure 7 and Figure 8.



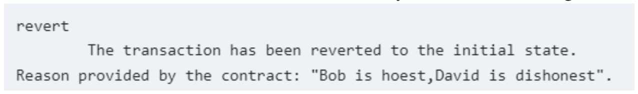
```
[vm] from: 0x5B3...eddC4 to: Datatrade.dataresult(uint256,uint256,uint256,uint256) 0xd91...39138 value: 0 wei
data: 0xbc8...00003 logs: 0 hash: 0x2ca...ca1aa
status              true Transaction mined and execution succeed
transaction hash    0x2ca20bd428f2f401f126dcb1a163d5f42f327dd303f3fbe0c55e6b3a05eca1aa
from                0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
to                  Datatrade.dataresult(uint256,uint256,uint256,uint256) 0xd9145CCE52D386f254917e481eB44e9943F39138
gas                 34655 gas
transaction cost    30134 gas
execution cost      30134 gas
input               0xbc8...00003
decoded input       {
                        "uint256 B1": "1",
                        "uint256 C": "1",
                        "uint256 D": "3",
                        "uint256 B2": "3"
                    }
```

(a) All servers return correct results.



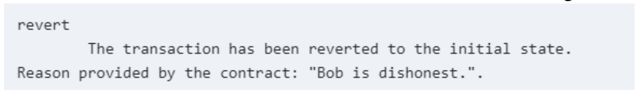
```
revert
        The transaction has been reverted to the initial state.
Reason provided by the contract: "Bob is hoest,Cindy is dishonest.".
```

(b) Bob returns the correct result, and Cindy returns the wrong result.



```
revert
        The transaction has been reverted to the initial state.
Reason provided by the contract: "Bob is hoest,David is dishonest".
```

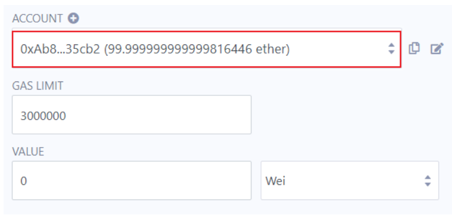
(c) Bob returns the correct result, and David returns the wrong result.



```
revert
        The transaction has been reverted to the initial state.
Reason provided by the contract: "Bob is dishonest.".
```

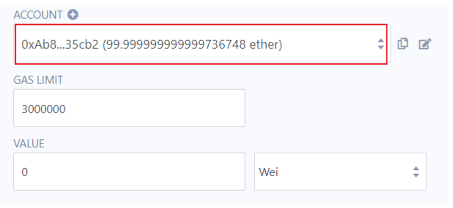
(d) All servers return incorrect results.

FIGURE 11. The verification process of the results returned by the server.



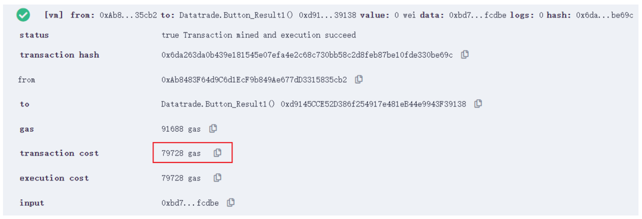
```
ACCOUNT
0xAb8...35cb2 (99.999999999999816446 ether)
GAS LIMIT
3000000
VALUE
0                Wei
```

(a) The balance of the server before the deposit is returned.



```
ACCOUNT
0xAb8...35cb2 (99.999999999999736748 ether)
GAS LIMIT
3000000
VALUE
0                Wei
```

(b) The balance of the server after the deposit is returned.



```
[vm] from: 0xAb8...35cb2 to: Datatrade.Button_Result1() 0xd91...39138 value: 0 wei data: 0xbd7...fcdbe logs: 0 hash: 0x6da...be69c
status              true Transaction mined and execution succeed
transaction hash    0x6da263da0b439e181545e07efa4e2c68c730bb58c2d8feb87be10fde330be69c
from                0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2
to                  Datatrade.Button_Result1() 0xd9145CCE52D386f254917e481eB44e9943F39138
gas                 91688 gas
transaction cost    79728 gas
execution cost      79728 gas
input               0xbd7...fcdbe
```

(c) Gas consumption of the deposit refund transaction process.

FIGURE 12. The deposit refund process of the server.

If the transaction is executed normally, Alice finally gets the data results, and Bob, Cindy and David get the data reward. In case of violation, the deposit of the violating party will be confiscated. When the client does not pay the reward, the deposit initially stored will be paid as the reward and the rest of the deposit refunded. We test the case of returning the deposit when the server Bob is honest. Before the test, the balance of Bob (Ethereum address: 0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2) is 99.999999999999816446 *ether*, then deducting the gas fee 79728 *gas* (1 *gas* = 1 *wei*) spent on returning the deposit transaction, and getting the contract returns the deposit amount of 30 *wei* to Bob. After the test, the balance of server Bob is 99.999999999999736748 *ether*, as shown in Figure 12. When the server returns an error result, the deposit stored by the server will be deducted and the rest will be returned. All operations of the distributed ledger log on the Ethereum blockchain will be recorded.

2) TIME CONSUMPTION ANALYSIS

In the data outsourcing computation transaction, each transaction stage is completed within a certain time constraint, so that the transaction is fair and timely. Since most transaction stages are based on smart contracts, the time spent in each step of the transaction will affect the efficiency of the transaction. Therefore, we will test the time consumption of several key steps in the smart contract, including storing the deposit, public key verification and returning of verification results, and give eight test results to verify the effectiveness of the time constraint. In the above steps, we only select a typical test object, and the specific implementation test time of each step is shown in Table 4.

TABLE 4. The time consumption of three transaction steps.

Test Steps	Eight Tests (s)							
	1	2	3	4	5	6	7	8
Store the deposit	1	2	1	1	1	2	1	2
Public key verification	3	2	3	2	3	3	2	3
Return verification results	2	3	2	2	3	2	3	3

3) GAS FEE ANALYSIS

Gas fee refers to the amount of computation required to perform transaction operations on the Ethereum blockchain network. The greater the amount of computation and energy consumption, the higher the transaction cost. In the data outsourcing computation transaction, all transaction operations are implemented by calling functions in the smart contract. Therefore, we compute the running costs of the four functions with the largest amount of computation, including set_deposit, send_deposit, send_PK, and verify_data, as shown in Table 4, as a measure of data outsourcing transaction costs. The transaction cost is calculated by converting the gas fee consumed by the function into US dollars (unit: US dollars). The computing formula is as follows. Cost $=$ Gas_fee$*10^{-9}*$312.02 (US dollars) (1 *gas* $=$ 1 *wei*, 1 *eth* $= 10^{-9}$ *wei*, 1 *eth* $=$ 312.02 US dollars). In Table 5, the gas fee costs for establishing and executing four transaction operations on the official Ethereum network are shown. As we can see, the cost of using the smart contract on Ethereum is very

TABLE 5. The gas cost of four functions.

Functions	Gas Used	Actual Cost	USD
set_deposit	68350	68350	0.021
send_deposit	38636	38636	0.012
send_PK	97332	97332	0.030
verify_data	30134	30134	0.009

TABLE 6. Comparison of our solution with existing solutions.

Property	[9]	[30]	[31]	[32]	Our solution
Smart contract based	Yes	No	No	No	Yes
Untamperable data	No	No	No	No	Yes
Verifiability	Yes	Yes	Yes	Yes	Yes
Fairness	Yes	Yes	Yes	Yes	Yes
Privacy	Yes	Yes	Yes	No	Yes

low, and the cost is roughly related to the computation and storage complexity of functions.

VI. DISCUSSION

In this section, we compare our data outsourcing transaction solution with other four existing solutions from relevant literatures in five important functional properties. If the solution has the feature, we mark it with 'Yes', otherwise with 'No'. The comparison results are shown in Table 6. With all comparison of the proposed solutions, all marks for properties of our solution are 'Yes'. That is, our solution meets all the attributes. Therefore, we can draw a conclusion that our solution has more advantages than other solutions in data outsourcing computation.

VII. CONCLUSION

This paper has proposed a data outsourcing computing scheme based on smart contract, which supports fair, privacy protection and verifiable transactions. In this scheme, the penalty mechanism based on smart contracts is used to implement automatic and low-cost transactions, and transaction data is recorded in the blockchain as evidence to resolve transaction disputes. The oblivious transfer technology is adopted to protect the data privacy of customers in transactions. Further, the improved replication-based verifiable technology is used to reduce transaction cost and computation cost. The smart contract of the scheme is written in solidity language, and deployed and tested in the Ethereum development environment Remix. The test case consists of one client and three servers to verify its effectiveness and efficiency. The simulation results show that the scheme is feasible, that is, when all parties are honest, the server gets paid, and the client gets the data results. Meanwhile, the transaction has fairness, verifiability and privacy protection. Particularly, the time consumption and cost consumption of smart contracts are very low and can be ignored. In future work, we will design more stable and secure smart contracts, and improve privacy protection to further ensure the privacy of transaction users.

ACKNOWLEDGMENT

(Lina Li and Tingting Zhang are co-first authors.)

REFERENCES

[1] W. Xiong and L. Xiong, "Data trading certification based on consortium blockchain and smart contracts," *IEEE Access*, vol. 9, pp. 3482–3496, 2021.

[2] A. Jyoti and R. K. Chauhan, "A blockchain and smart contract-based data provenance collection and storing in cloud environment," *Wireless Netw.*, vol. 28, no. 4, pp. 1541–1562, May 2022.

[3] C. Dong, Y. Wang, A. Aldweesh, P. McCorry, and A. van Moorsel, "Betrayal, distrust, and rationality: Smart counter-collusion contracts for verifiable cloud computing," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2017, p. 211.

[4] Y. Jiang and Y. Bo, "A privacy-preserving digital rights management protocol based on oblivious transfer scheme," *Int. J. Digit. Content Technol. its Appl.*, vol. 5, no. 5, pp. 337–341, May 2011.

[5] S. Wang, L. Ouyang, Y. Yuan, X. Ni, X. Han, and F.-Y. Wang, "Blockchain-enabled smart contracts: Architecture, applications, and future trends," *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 49, no. 11, pp. 2266–2277, Nov. 2019.

[6] A. Al Omar, A. K. Jamil, A. Khandakar, A. R. Uzzal, R. Bosri, N. Mansoor, and M. S. Rahman, "A transparent and privacy-preserving healthcare platform with novel smart contract for smart cities," *IEEE Access*, vol. 9, pp. 90738–90749, 2021.

[7] R. Gennaro, "Verifiable outsourced computation: A survey," in *Proc. ACM Symp. Princ. Distrib. Comput.*, 2017, p. 313.

[8] L. Zhang and R. Safavi-Nain, "Batch verifiable computation of outsourced functions," *Des., Codes Cryptogr.*, vol. 77, no. 2, pp. 563–585, 2015.

[9] M. R. Dorsala, V. N. Sastry, and S. Chapram, "Fair payments for verifiable cloud services using smart contracts," *Comput. Secur.*, vol. 90, Mar. 2020, Art. no. 101712.

[10] W. S. Park, H. Lee, and J.-Y. Choi, "Formal modeling of smart contract-based trading system," in *Proc. 24th Int. Conf. Adv. Commun. Technol. (ICACT)*, Feb. 2022, pp. 48–52.

[11] M. Sharifinejad, A. Dorri, and J. Rezazadeh, "Bis—A blockchain-based solution for the insurance industry in smart cities," 2020, *arXiv:2001.05273*.

[12] R. Kumar, W. Wang, J. Kumar, T. Yang, A. Khan, W. Ali, and I. Ali, "An integration of blockchain and AI for secure data sharing and detection of CT images for the hospitals," *Computerized Med. Imag. Graph.*, vol. 87, Jan. 2021, Art. no. 101812.

[13] M. Pawlak and A. Poniszewska-Marańda, "Trends in blockchain-based electronic voting systems," *Inf. Process. Manage.*, vol. 58, no. 4, Jul. 2021, Art. no. 102595.

[14] M. R. Dorsala, V. N. Sastry, and S. Chapram, "Blockchain-based solutions for cloud computing: A survey," *J. Netw. Comput. Appl.*, vol. 196, Dec. 2021, Art. no. 103246.

[15] M. Zhaofeng, W. Lingyun, W. Xiaochang, W. Zhen, and Z. Weizhe, "Blockchain-enabled decentralized trust management and secure usage control of IoT big data," *IEEE Internet Things J.*, vol. 7, no. 5, pp. 4000–4015, May 2019.

[16] W. Xiong and L. Xiong, "Anti-collusion data auction mechanism based on smart contract," *Inf. Sci.*, vol. 555, pp. 386–409, May 2021.

[17] A. Kumar, K. Abhishek, P. Nerurkar, M. R. Ghalib, A. Shankar, and X. Cheng, "Secure smart contracts for cloud-based manufacturing using ethereum blockchain," *Trans. Emerg. Telecommun. Technol.*, vol. 33, no. 4, p. 4129, Apr. 2022.

[18] T. Li, W. Ren, and Y. Xiang, "FAPS: A fair, autonomous and privacy-preserving scheme for big data exchange based on oblivious transfer, ether cheque and smart contracts," *Inf. Sci.*, vol. 544, pp. 469–484, Feb. 2021.

[19] T. Li, D. Li, and M. Wang, "Blockchain-based fair and decentralized data trading model," *Comput. J.*, vol. 65, no. 8, pp. 2133–2145, Aug. 2021.

[20] D. Hu, Y. Li, L. Pan, M. Li, and S. Zheng, "A blockchain-based trading system for big data," *Comput. Netw.*, vol. 191, May 2021, Art. no. 107994.

[21] Z. Ullah, B. Raza, H. Shah, S. Khan, and A. Waheed, "Towards blockchain-based secure storage and trusted data sharing scheme for IoT environment," *IEEE Access*, vol. 10, pp. 36978–36994, 2022.

[22] Z. Chen, Y. Tian, and C. Peng, "An incentive-compatible rational secret sharing scheme using blockchain and smart contract," *Sci. China Inf. Sci.*, vol. 64, no. 10, pp. 1–21, Oct. 2021.

[23] Y. Xiang, W. Ren, T. Li, X. Zheng, T. Zhu, and K.-K.-R. Choo, "A multi-type and decentralized data transaction scheme based on smart contracts and digital watermarks," *J. Netw. Comput. Appl.*, vol. 176, Feb. 2021, Art. no. 102953.

[24] V. Reniers, Y. Gao, R. Zhang, P. Viviani, A. Madhusudan, B. Lagaisse, S. Nikova, D. Van Landuyt, R. Lombardi, B. Preneel, and W. Joosen, "Authenticated and auditable data sharing via smart contract," in *Proc. 35th Annu. ACM Symp. Appl. Comput.*, Mar. 2020, pp. 324–331.
[25] K. Liu, H. Desai, L. Kagal, and M. Kantarcioglu, "Enforceable data sharing agreements using smart contracts," 2018, *arXiv:1804.10645*.
[26] W. Xiong and L. Xiong, "Smart contract based data trading mode using blockchain and machine learning," *IEEE Access*, vol. 7, pp. 102331–102344, 2019.
[27] S. Xuan, L. Zheng, I. Chung, W. Wang, D. Man, X. Du, W. Yang, and M. Guizani, "An incentive mechanism for data sharing based on blockchain with smart contracts," *Comput. Electr. Eng.*, vol. 83, May 2020, Art. no. 106587.
[28] S. Wang, X. Tang, Y. Zhang, and J. Chen, "Auditable protocols for fair payment and physical asset delivery based on smart contracts," *IEEE Access*, vol. 7, pp. 109439–109453, 2019.
[29] H. Wang, H. Qin, M. Zhao, X. Wei, H. Shen, and W. Susilo, "Blockchain-based fair payment smart contract for public cloud storage auditing," *Inf. Sci.*, vol. 519, pp. 348–362, May 2020.
[30] X. Chen, J. Li, and W. Susilo, "Efficient fair conditional payments for outsourcing computations," *IEEE Trans. Inf. Forensics Security*, vol. 7, no. 6, pp. 1687–1694, Dec. 2012.
[31] B. Carbunar and M. V. Tripunitara, "Payments for outsourced computations," *IEEE Trans. Parallel Distrib. Syst.*, vol. 23, no. 2, pp. 313–320, Feb. 2012.
[32] A. Küpçü, "Incentivized outsourced computation resistant to malicious contractors," *IEEE Trans. Dependable Secure Comput.*, vol. 14, no. 6, pp. 633–649, Nov. 2015.
[33] C. Lin, D. He, X. Huang, and K.-K.-R. Choo, "OBFP: Optimized blockchain-based fair payment for outsourcing computations in cloud computing," *IEEE Trans. Inf. Forensics Security*, vol. 16, pp. 3241–3253, 2021.
[34] Y. Zhang, R. H. Deng, X. Liu, and D. Zheng, "Outsourcing service fair payment based on blockchain and its applications in cloud computing," *IEEE Trans. Services Comput.*, vol. 14, no. 4, pp. 1152–1166, Jul. 2018.
[35] A. Jain and C. Hari, "A new efficient protocol for k-out-of-n oblivious transfer," *Cryptologia*, vol. 34, no. 4, pp. 282–290, Sep. 2010.
[36] D.-C. Lou and H.-F. Huang, "An efficient t-out-of-n oblivious transfer for information security and privacy protection," *Int. J. Commun. Syst.*, vol. 27, no. 12, pp. 3759–3767, Dec. 2014.
[37] Y. Mu, J. Zhang, and V. Varadharajan, "m out of n oblivious transfer," in *Proc. Australas. Conf. Inf. Secur. Privacy*, 2002, pp. 395–405.
[38] C. Chu and W. Tzeng, "Efficient k-out-of-n oblivious transfer schemes with adaptive and non-adaptive queries," in *Proc. Int. Workshop Public Key Cryptogr.*, 2005, pp. 172–183.
[39] J. Lai, Y. Mu, F. Guo, R. Chen, and S. Ma, "Efficient k-out-of-n oblivious transfer scheme with the ideal communication cost," *Theor. Comput. Sci.*, vol. 714, pp. 15–26, Mar. 2017.
[40] S. Esmaeilzade, N. Pakniat, and Z. Eslami, "A generic construction to build simple oblivious transfer protocols from homomorphic encryption schemes," *J. Supercomput.*, vol. 78, no. 1, pp. 72–92, Jan. 2022.
[41] H. Yang, J. Shen, J. Lu, T. Zhou, X. Xia, and S. Ji, "A privacy-preserving data transmission scheme based on oblivious transfer and blockchain technology in the smart healthcare," *Secur. Commun. Netw.*, vol. 2021, pp. 1–12, Sep. 2021.
[42] J. Shen, H. Yang, P. Vijayakumar, and N. Kumar, "A privacy-preserving and untraceable group data sharing scheme in cloud computing," *IEEE Trans. Dependable Secure Comput.*, vol. 19, no. 4, pp. 2198–2210, Jul. 2021.
[43] S. Kak, "Oblivious transfer protocol with verification," 2015, *arXiv:1504.00601*.
[44] A. Damodaran and A. Rial, "Unlinkable updatable databases and oblivious transfer with access control," in *Proc. Australas. Conf. Inf. Secur. Privacy*, 2020, pp. 584–604.
[45] R. Kumaresan and I. Bentov, "How to use Bitcoin to incentivize correct computations," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, Nov. 2014, pp. 30–41.
[46] S. Simunic, D. Bernaca, and K. Lenac, "Verifiable computing applications in blockchain," *IEEE Access*, vol. 9, pp. 156729–156745, 2021.
[47] J. Lee, C. Nicopoulos, G. Jeong, J. Kim, and H. Oh, "Practical verifiable computation by using a hardware-based correct execution environment," *IEEE Access*, vol. 8, pp. 216689–216706, 2020.
[48] R. Canetti, B. Riva, and G. N. Rothblum, "Practical delegation of computation using multiple servers," in *Proc. 18th ACM Conf. Comput. Commun. Secur. (CCS)*, 2011, pp. 445–454.
[49] S. Avizheh, M. Nabi, R. Safavi-Naini, and K. M. Venkateswarlu, "Verifiable computation using smart contracts," in *Proc. ACM SIGSAC Conf. Cloud Comput. Secur. Workshop (CCSW)*, 2019, pp. 17–28.
[50] M. Belenkiy, M. Chase, C. C. Erway, J. Jannotti, A. Küpçü, and A. Lysyanskaya, "Incentivizing outsourced computation," in *Proc. 3rd Int. workshop Econ. Networked Syst. (NetEcon)*, 2008, pp. 85–90.

LINA LI received the M.S. degree from the School of Computer Science and Technology, Harbin Institute of Technology, Harbin, in 2006, and the Ph.D. degree in computer system architecture from the College of Computer Science and Technology (CCST), Jilin University, Changchun, in 2019. She is an Associate Professor with the College of Computer Science and Technology, Changchun University. Her research interests include blockchain, deep learning, and cloud computing.

TINGTING ZHANG was born in Changchun, Jilin, China, in 1997. She is currently pursuing the master's degree with the College of Cyber Security, Changchun University, Jilin. She has participated in several provincial scientific research projects of the College of Computer Science and Technology, and relevant research results have applied for an invention patent and published an EI conference paper. Her research interests include blockchain and smart contract.

GUODONG SUN received the Ph.D. degree in computer science from the Harbin Institute of Technology, Harbin, China, in 2009. He was a Postdoctoral Researcher at Tsinghua University, Beijing, China. He is currently an Associate Professor with the Department of Internet-of-Things, Beijing Forestry University, Beijing. His research interests include machine learning, mobile computing, wireless algorithms, and the Internet-of-Things.

DEZHENG JIN was born in Anyang, China, in 1997. He is currently pursuing the postgraduate degree with Changchun University. He has a strong interest in the direction of blockchain and has conducted some research about smart contract and consensus mechanism in this field. He participated in a provincial scientific research project based on blockchain technology. He has published an EI conference paper and applied for an invention patent. His research interests include blockchain and deep learning.

NIANFENG LI received the Ph.D. degree in mechatronics engineering from the Changchun Institute of Optics, Fine Mechanics and Physics, Chinese Academy of Sciences. He is a Professor and the Dean of College of Computer Science and Technology (CCST), Changchun University, China. He is also the Vice Director of the Biomedical Engineering Research and Development Center, Changchun University. His current major research interests include image processing, somatosensory technology, rehabilitation training systems, and campus safety technology.