information

MDPI

Article

Decentralized Transaction Mechanism Based on Smart Contract in Distributed Data Storage

Yonggen Gu [1,2], Dingding Hou [2], Xiaohong Wu [1,*], Jie Tao [1] and Yanqiong Zhang [1]

[1] School of Information Engineering, Huzhou University, Huzhou 313000, China; gyg@zjhu.edu.cn (Y.G.); taojie@zjhu.edu.cn (J.T.); zhangyanqiong@zjhu.edu.cn (Y.Z.)

[2] College of Mathematics and Computer Science, Zhejiang Normal University, Jinhua 321004, China; dingding@163.com

* Correspondence: xhwu@zjhu.edu.cn

Received: 1 October 2018; Accepted: 14 November 2018; Published: 17 November 2018

Abstract: Distributed data storage has received more attention due to its advantages in reliability, availability and scalability, and it brings both opportunities and challenges for distributed data storage transaction. The traditional transaction system of storage resources, which generally runs in a centralized mode, results in high cost, vendor lock-in and single point failure risk. To overcome the above shortcomings, considering the storage policy with erasure coding, in this paper we propose a decentralized transaction method for cloud storage based on a smart contract, which takes into account the resource cost for distributed data storage. First, to guarantee the availability and decrease the storing cost, a reverse Vickrey-Clarke-Groves (VCG) based auction mechanism is proposed for storage resource selection and transaction. Then we deploy and implement the proposed mechanism by designing a corresponding smart contract. Especially, we address the problem of how to implement a VCG-like mechanism in a blockchain environment. Based on the private chain of Ethereum, we make the simulation for the proposed storage transaction method. The results of simulation show that the proposed transaction model can realize competitive trading of storage resources and ensure the safe and economic operation of resource trading.

Keywords: cloud storage; VCG (Vickrey-Clarke-Groves) mechanism; smart contract; security; erasure coding

1. Introduction

With the development of network technology, cloud storage technology developed with cloud computing has attracted more and more attention from enterprises and researchers [1–3]. The characteristics such as high reliability, availability and scalability have attracted more and more cloud users [4–7]. Many public cloud storage platforms are available in the market such as Google Storage, Amazon S3, Microsoft Azure, RackSpace CloudFiles, etc. However, these cloud storage platforms or applications suffered from different problems such as single-point failures and vendor lock-in [8–10]. For example, Microsoft Azure and Amazon S3 have experienced service outages for several times, causing huge economic losses to cloud storage service companies and users. In the result, the multi-cloud storage has been proposed, which places data on multiple could storage providers to achieve better quality of service by avoiding vendor lock-in and improving fault-tolerance. The multi-cloud storage is beneficial to applications which need to keep a large amount of data such as data backup, document archiving or electronic health recording.

In a multi-cloud system, data can be held by multiple storage providers to avoid vendor lock-in and improve fault-tolerant which obtains higher Quality of Service (QoS) in reliability, availability and scalability than single-cloud storage. Storage policy with erasure coding combined with a multi-cloud environment can not only improve the QoS in storage but also reduce the cost of data storage, and

it has attracted more studies [11–14]. With erasure coding, a demander's data are divided into k blocks, and then these blocks are used to generate $n-k$ encoded data blocks (n blocks in total, $n > k$). The numbers n and k are called as parameter (n, k) of erasure coding. These n data blocks will be placed to n cloud storage providers, and each provider holds just one data block respectively. Compared to full replication, the storage policy with erasure coding can decrease the backup size of the data. With this feature, a demander can choose the cheaper providers as well as keeping the QoS. Some works have used feature of erasure coding in multi-cloud storage system [15–17].

With the storage policy of erasure coding, how to select the cloud providers to store the data and execute the transaction securely is an important problem. Transaction mechanisms in cloud storage have been studied [17–22], some of which address the problem—how to choose the resources in competitive environment such as [18,19]. However, although these mechanisms investigate the distributed storage to avoid the disadvantage of centralized mode, they still adopt traditional centralized trading mode, without considering the transaction in decentralized environment. The weaknesses of centralized model still exist such as high cost, vendor lock-in, single point failure risk, etc. Recently, the blockchain technology has been applied in storage transaction such as Storj [23], N2C [24], but these applications did not investigate the cloud provider selection mechanism in the competitive environment.

Therefore, in this paper, to help the storage demander select suitable resource providers and execute transactions in decentralized environment, we construct a decentralized resource transaction mechanism with blockchain technology.

The contributions of our work are as follows.

First, to guarantee the availability and decrease the storing cost for storage policy with erasure coding, a reverse VCG (Vickrey-Clarke-Groves)-based auction mechanism is proposed for storage resource selection and transaction which can incentivize the cloud providers to report truthful cost in decentralized environment.

Second, we deploy the proposed mechanism by designing a corresponding smart contract. Especially, by dividing the whole transaction process into four stages: publishing requirement, submitting sealed bids, revealing sealed bids and auction, and payment. We present a solution for the problem—how to implement a VCG-like mechanism in a blockchain environment.

The remainder of the paper is organized as follows. The problem model and transaction mechanism are respectively introduced in Sections 2 and 3. In Section 4 we focus on the cloud storage resource transaction method based on smart contract. In Section 5 it is the experimental analysis in the private chain to show the correctness of transaction and finally we conclude and discuss the future work of the paper in Section 6.

2. Problem Model

2.1. Erasure Coding

Erasure coding is considered as a data protection method, which divides data into segments, and expands or encodes the redundant data blocks. As aforementioned, the parameter (n, k) of erasure coding means that a data file is split into n chunks $(n > k)$, where any subset including k chunks is sufficient to reconstruct a complete copy of the data. The rate $r = k/n (k/n < 1)$ of an erasure coding is the fraction of chunks required to rebuild the original data. In Figure 1, the original data can be rebuilt with the chunks stored at any three of the four cloud providers. Redundant data block is generated by other three data blocks using erasure coding algorithm.

Erasure coding allows to tolerate up to $n-k$ providers outage. Hence, even though $n-k$ providers fail to provision service at same time, which does not affect the demander downloading the complete data from erasure coding system. Furthermore, erasure coding provides finer granularity than full replication, making it easier to choose different providers for entire data in trading.

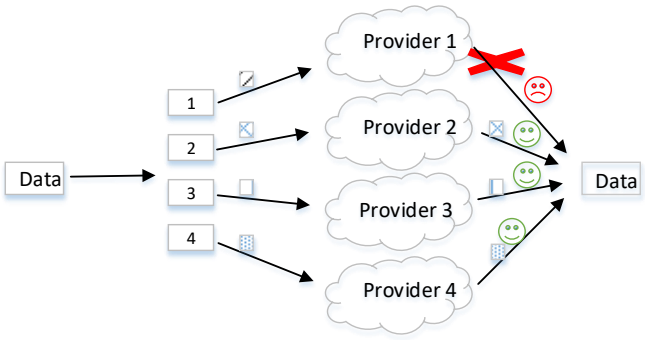

Figure 1. Erasure coding $(4,3)$: any 3-subset of the 4 chunks.

2.2. Cloud Storage Procurement Problem

In our cloud storage resource transaction, a demander is required to post his requirement. Providers will submit their bids according to the demander's requirement if they can provision enough storage resource. From the perspective of the demander, the goal is to select out a set of providers to place their data with minimal cost. When storage resource is sold, providers are required to provide storage services which include both storage and network bandwidth, to guarantee demander's data accessibility.

Figure 2 shows our model with blockchain. The demander posts requirement information, including the number of storage providers num and data block capacity $S_{capacity}$. We set the parameter (num, k) of erasure coding in our model, where k represents that demander's data have been divided into k blocks, and num represents how many providers that the demander wants to buy storage resource from. The provider submits the bid to the demander, and the winner will provision the storage resource and be paid the reward. Miners add past transaction records to the blockchain and validate a new block by the consensus protocol. Their jobs are to guarantee blockchain security and to earn transaction fees and mining rewards.

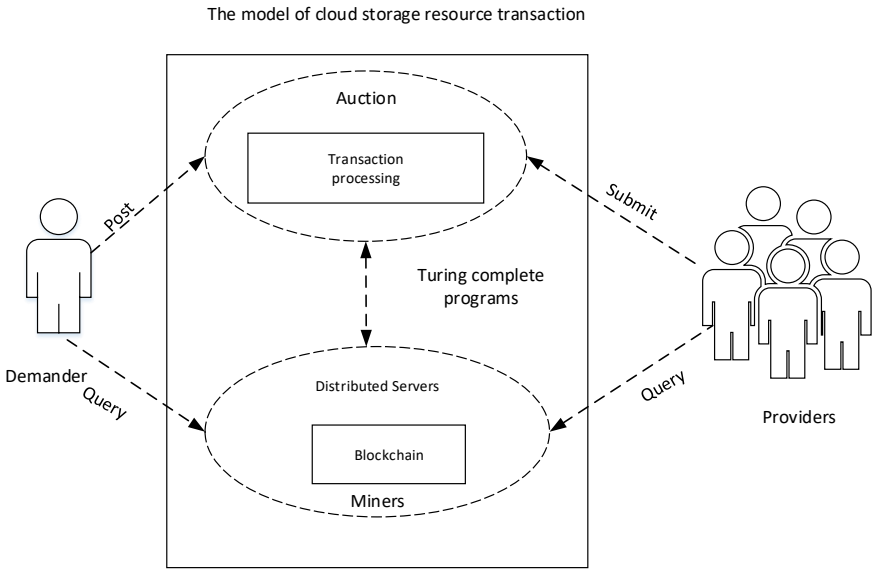

Figure 2. The model of cloud storage resource transaction.

In this paper, we propose a VCG-based mechanism of cloud storage transaction which allows providers to compete freely with each other for price to make benefits and increases the utilization of social resources.

If the demander publishes the demand information $\{num, S_{capacity}\}$ in a transaction, the total capacity of storage is $num \times S_{capacity}$. Let $\mathcal{N}$ denote the set of persons participating the mechanism, $\mathcal{N} = \{1, 2, ..., N\}$. Consequently, the demander receives a stream of bids, $B = (b_1, b_2, \cdots, b_N)$, and needs to make decision on each bid. The bid b_i means that the storage provider i would like to provision the resource with lowest price b_i. We assume the set of variables X to represent the result of the auction and the providers do not influence each other when providers participate in the auction. So $X = \{x_1, x_2, \cdots, x_N\}$ and satisfies

$$x_i = \begin{cases} 1, & if\ i\ is\ a\ winner \\ 0, & otherwise \end{cases} \tag{1}$$

Based on the above assumptions, the objective of this scenario is to minimize the total cost of the demander. That is

$$\begin{aligned} &\min \sum_{i \in \mathcal{N}} x_i c_i \\ s.t.\ &\sum_{i \in \mathcal{N}} x_i = num \\ &c_i \geq 0\ \forall i \in \mathcal{N} \\ &x_i = 0\ or\ 1,\ \forall i \in \mathcal{N} \end{aligned} \tag{2}$$

We have discussed the objective of the demander in this section. Incentive compatibility is an important property to encourage providers reporting truthful cost c_i, i.e., $b_i = c_i$. In the next section we will introduce the transaction mechanism in our system.

3. Storage Resource Transaction Mechanism

First, we introduce the definition of incentive compatible.

Definition 1 (Incentive Compatible, IC)**.** *A mechanism is incentive compatible, if for any provider i, regardless of the type reports of other providers, declaring a bid that reveals its true type can maximize its utility. Let b_i be the truthful cost of storage resource to provider i. Formally, given any cost profile of others b_{-i}, we have:*

$$u_i(f(b_i, b_{-i}), b_i) \geq u_i(f(b_i', b_{-i}), b_i)$$

To minimize the total storage cost, we should design an incentive compatible mechanism to make providers reveal their true bids.

Now we introduce reverse VCG auction mechanism. Let $\mathcal{X}$ denote the set of available allocation results. $X \in \mathcal{X}$ is a allocation result, $X = \{x_1, x_2, ..., x_N\}$. Given cost profile of all providers $c_1, c_2, \cdots, c_N$, a reverse VCG mechanism satisfies following rules:

(1) Allocation function can minimize the social cost: $f(c_1, c_2, \cdots, c_N) = arg\min_{X \in \mathcal{X}} \sum_{i=1}^{N} c_i x_i$

(2) Let $X^a = f(c_i, c_{-i})$, $X^b = f(c_{-i})$. The payment p_i for provider i is $p_i(c_1, c_2, \cdots, c_N) = \sum_{j \neq i} c_j x_j^b - \sum_{j \neq i} c_j x_j^a$.

The first rule means that the allocation of reverse VCG auction mechanism always minimizes the social cost. In the second rule, X^a is the allocation with minimal social cost, and X^b is the allocation with minimal social cost if i is not present. $\sum_{j \neq i} c_j x_j^b$ is the minimal social cost if i does not participate the mechanism, and $\sum_{i \neq j} c_j x_j^a$ is the total social cost of other participants excluding i under the allocation X^a. Therefore, the payment of i implies the saving social cost due to the participation of provider

i. The reverse VCG auction mechanism is incentive compatible, the proof process is similar to VCG mechanism [25].

Assume that the storage policy is based on erasure coding. In our work, we design a reverse VCG-based transaction mechanism to help the demander select cloud providers with different costs. Figure 3 explains these functions in detail.

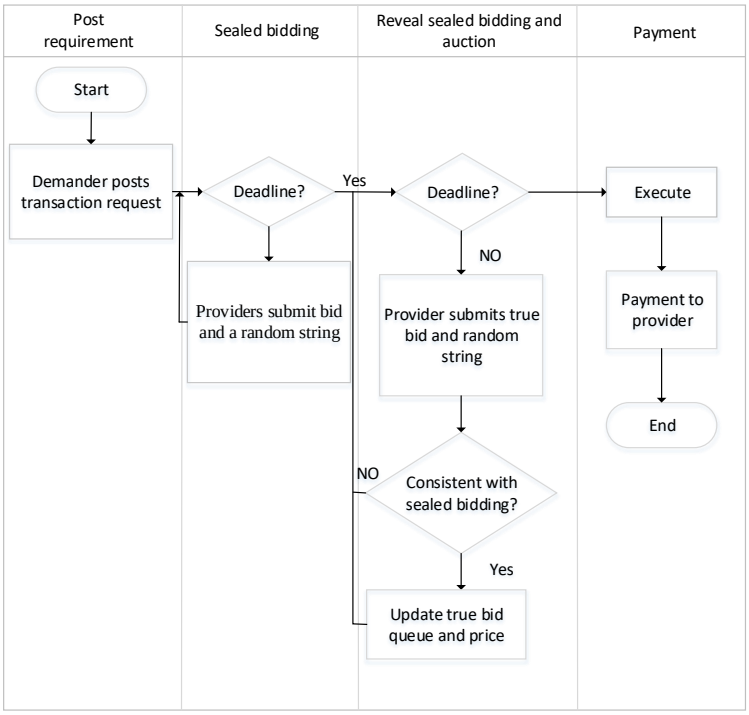

Figure 3. The diagram of transaction process based on smart contract.

The process in Figure 3 is as following:

Step 1 The demander posts requirement information, including the required number of cloud providers num, the availability of resource, required storage capacity $S_{capacity}$.

Step 2 Each provider who can provision storage resource and satisfy the requirement of availability submits the bid b_i.

Step 3 When the amounts of bidders who participate in the transaction is larger than num, we think that the transaction is successful. The allocation and payment are determined by the rule of VCG mechanism. Otherwise, the amount of provider participating in the transaction is less than num, the transaction is failed.

Step 4 Select num providers who can minimize the total costs of the demander as the winners. The payoff of the winner is $p_i = \sum_{j \neq i} c_j x_j^b - \sum_{j \neq i} c_j x_j^a$, where $X^a = \{x_j | j \in \mathcal{N}\}$ is the optimal allocation, and $X^b = \{x_j | j \in \mathcal{N}, j \neq i\}$ is the optimal allocation if provider i is not present.

In our mechanism, according to Step 4, the winners are the num providers with the lowest bids, and computed payoff of the winner equals to the highest bid of the losers. According to the above processes, we can get a truthful transaction mechanism, lowest social cost of the required resource. At the same time, our mechanism has some features of erasure coding such as high fault-tolerance and avoiding vendor lock-in.

4. The Method of Transaction Based on Smart Contract

In this section, we introduce blockchain and smart contract, and implement a distributed transaction method of cloud storage with smart contract in detail.

4.1. Blockchain and Smart Contract

Bitcoin (BTC) is the first digital currency based on the blockchain mechanism [26], and the Bitcoin network is the first large-scale and long-term digital currency system in history. The main features of the Bitcoin network are decentralization, anonymity and inflation prevention. The definition of blockchain is enlightened from the application of Bitcoin.

A continuous period of transaction is a collection of records in a file called "block", and each "block" contains a hash value of former "block". Thus the blocks form the hash chain which is called the chain block. Blockchain is analogous to a distributed database technology that maintains continuously growing, tamper-resistant data records by maintaining the blockchain structure. Users provide the computing resources they want to get the packaging rights for the blocks because the winners can win rewards and transaction fees. Blockchain technology has been used in many ways, such as storage systems, medical record sharing and video games.

The smart contract was first proposed by Nick Szabo in 1994 [27]. By definition, a smart contract is a set of promises defined digitally, including promises on which contract participants can execute those promises. The implementation of smart contracts requires a distributed, immutable, programmable security environment, which is provided by blockchain technology [28]. Currently, the blockchain platforms supporting smart contract are Ethereum and Hyperledger.

Ethereum is an open-source, public, blockchain-based distributed computing platform and operating system featuring smart contract functionality. Through the platform, developers can write any form of smart contract to publish and run on the Ethereum public chain, test chain or private chain. Ethereum provides a programming environment of Turing complete, and it supports smart contract by Solidity which is a language like other advanced languages in function. Smart contract is similar to a virtual user that can trade and transfer information with other users or smart contracts. Each user can act as a miner, to collect transactions or transmit information during the current period, or to run the relevant smart contract code and calculate the latest state of Ethereum based on the past block data.

The decentralization and completeness of the smart contract provide a very good technical support for realizing distributed transaction of storage resources. In next section we mainly introduce the method based on the Ethereum platform to realize the smart contract of resource transaction which is the core part of distributed transaction.

4.2. Smart Contract for Distributed Transaction

There are three rules for storage transaction to implement smart contract:

(1) All users can publish resources and participate in the auction.
(2) The provider's bid is confidential at the bidding stage.
(3) Contract execution results are settled automatically.

In this section, since the bid information submitted by each provider is also received by other providers, it cannot realize a sealed VCG-based mechanism directly in blockchain environment. To solve this problem, the transaction process is divided into four stages. There are four main functions which are corresponding to the four stages, including publishing requirement, submitting sealed bids, revealing sealed bids and auction, payment function. The following is the design of the smart contracts including the four functions.

4.2.1. Publishing Requirement

Each user can publish requirement in the transaction system. The smart contract records the publish request, which is visible to all users. The demander waits for the providers to submit the bid

information. t is the current time, t_{bid} is the deadline of biding stage, t_{rev} is the deadline of biding reveal stage, and t_{end} is the deadline of auction. $state_{contract}$ is the state of contract, which aims to avoid conflict among different demanders' requirement. After the requirement is published, $state_{contract}$ turns to false, and when requirement is ended, $state_{contract}$ turns to true. The post transaction can be described as Algorithm 1.

Algorithm 1: Publishing Requirement

Input: demander D, the deadline of biding stage t_{bid}, the deadline of biding reveal stage t_{rev}, the deadline of auction t_{end}, required number of providers W_{num}, data block capacity $S_{capacity}$

```
if state_contract = false then
    D fails to publish transaction;
else
    publish transaction with D, t_bid, t_rev, t_end, W_num, S_capacity;
    state_contract ← false;
```

4.2.2. Submitting Sealed Bids

It indicates that the provider uses a non-reverse-solvable, easily verifiable hash function to connect its true bid information by a series of random strings. In this way, the sealed bid not only contains the true bid information but also ensures that the information is not leaked to other providers. W is the set of providers who have submitted bids. The encrypting hash as a sealed bid is recorded in set $bid_{list}^{encryption}$ before the end of the sealed bid time. We use SHA-3 function to encrypt the bid information. $SHA-3(b_i, string_i)$ indicates the $SHA-3$ hash function which is a encryption algorithm. b_i represents the cost of the provider and $string_i$ is a user-defined random string. All the bids of providers are encrypted and recorded in the set $bid_{list}^{encryption}$. The Algorithm 2 presents the process.

Algorithm 2: The Process of Sealed Bidding

Input: provider i, bid b_i, random string $string_i$, the deadline of biding stage t_{bid}

Output: The set of sealed bids $bid_{list}^{encryption}$

```
if t > t_bid then
    i fails to submission;
else
    W ← W ∪ {i};
    bid^encryption ← SHA-3(b_i, string_i);
    bid_list^encryption ← {bid^encryption} ∪ bid_list^encryption;
    return bid_list^encryption;
```

4.2.3. Revealing Sealed Bids and Auction

In this stage, the provider needs to submit its own bid and random string before the deadline, and the smart contract will verify whether $SHA-3(b_i, string_i)$ is consistent with the sealed bid submitted by the provider. when provider i submits the bid without encryption, if it is consistent with the bid encrypted in stage sealed bidding, the bidding will be considered valid. Otherwise, the bidding is invalid information and $dealprice_i = 0$. When the bidding of provider i is valid, the following steps should be processed which are shown in Algorithm 3. In line 8, we use b_i^{rev} to represent the bid and record it in set $trueBid_{list}$. In lines 9–15, we sort the bids in set $trueBid_{list}$ in ascending order, and calculate the payment $dealprice$ which is the highest bid of losers according to the VCG-based payment rule. For each provider j, j is a winner if his b_j^{rev} is less then $dealprice$. Finally in line 16 we return

$dealprice_j$ of each provider j who participates in the auction. The Algorithm 3 has shown the process of reveal bids and auction.

Algorithm 3: Revealing Sealed Bids and Auction

Input: provider i, reveal bid b_i^{rev}, reveal string $string_i^{rev}$, t_{rev}, t_{bid}, W_{num}
Output: deal price $dealprice_i$

```
if t > t_rev and t < t_bid then
    Fail to reveal;
else
    trueBid_list ← ∅;
    bid_rev ← SHA-3(b_i^rev, string_rev^i);
    for bid_rev ∈ bid_list^encryption do
        if bid_rev == bid_rev^encryption then
            trueBid_list ← {b_i^rev} ∪ trueBid_list;
            sort trueBid_list in ascending order;
            for each j ∈ W do
                if bid_j^rev > dealprice then
                    dealprice_j ← 0;
                else
                    computing dealprice based payment rule;
                    dealprice_j = dealprice;
return all dealprice_j j ∈ W ;
```

4.2.4. Payment

There are two types of result for cloud providers. The losers obtain zero payment because they did not contribute their resources. The winners will receive a payment from the demander, and the function $sendReward(i, dealprice_i \cdot S_{capacity})$ will send reward $dealprice_i \cdot S_{capacity}$ to i. We describe the algorithm for transaction settlement in Algorithm 4.

Algorithm 4: Payment

Input: $dealprice_i$, t_{end}, $S_{capacity}$

```
if t ≥ t_end then
    for each i ∈ W do
        sendReward(i, dealprice_i · S_capacity);
    state_contract ← true;
```

5. Experimental Analysis

The main objective of our work is to design a distributed data storage transaction mechanism in blockchain environment. In order to test our proposed mechanism, we have implemented a software prototype on Ethereum and write complex codes by Solidity. The simulation used five demanders and 20 providers to construct auction scenario. Smart contract was executed on our proposed mechanism. We assume that all providers can provision storage resource and bid for demander's request.

In order to evaluate the correctness of the mechanism proposed in this paper, we publish the smart contract which realizes the resource transaction mechanism on the private chain of Ethereum. The experimental environment as a distributed transaction platform simulates the auction environment. We assume that there always exist demanders in the auction, and we consider providers

are independent identically. A demander publishes the number of suppliers num, data block size $S_{capacity}$ as shown in Table 1 and 20 providers participate in the auction. The bidding strategy of each provider is related to his total cost of resources, which mainly includes storage cost and bandwidth cost, etc. We suppose that the cost of each provider follows uniform distribution in $[b_{min}, b_{max}]$, and we choose $b_{min} = 1$ and $b_{max} = 10$. After the smart contract is deployed in Ethereum blockchain, we publish the bid information of a demander including num, $S_{capacity}$ and deadline of each stage. In our experiments, we set same time periods for each demander. The stage period of submitting sealed bids is within 5 min. A duration of 5–8 min is the stage period of revealing sealed bids and auction; the next 8–10 min is the payment period. We pre-deposit $10eth$ per account before starting the simulation, because the user has to pay a service fee when invoking the smart contract. For the sake of preventing DDoS attacks, providers are required to send Ether as deposit in the auction. Furthermore, the bid information of providers is encrypted and saved in data storage, thus no malicious attackers can decrypt and read them. By this way, the cloud storage resource transaction could be performed by smart contract rather than any unbelievably third agent.

Table 1. The test transaction data published by demanders.

Transaction	num	$S_{capacity}$
1	5	24
2	4	56
3	7	10
4	6	63
5	3	82

We used the Ubuntu 16.4 $x86\_64$ operating system to install the Ethereum private chain for testing environment. The stable version 1.82 of GETH was installed. And in order to operate smart contract easily, we installed the Ethereum wallet for smart contract deployment. After a series of complicate environments have been built, the Ethereum private chain for deploying smart contract can be officially started. Smart contract deployment is to pay attention to Solidity syntax, and it is better to perform simple operations on smart contract functions to avoid gas shortage which may lead to transaction failure.

First, we completed five transactions of experiments according to above procedures. In these experiments, each provider who can provision resources submits the two consistent bids in Stage 2 and Stage 3. Figure 4 shows the total cost of demander with each sample transaction. We compared the cost with that computed by MATLAB, and the results are consistent which implies the correct of our algorithm of auction mechanism. In addition, each provider can come or go freely in the auction, which does not affect the overall process of trading.

Furthermore, we select transaction 2 as an example to further check the correctness of functions in four stages. We show part of bidding information of providers in Table 2 because of page size limited. The cost and random string are combined to form sealed bid and revealed bid. For example, the bid "6.41edf" of provider i means that the bid is "6.41" and the encrypted random string is "edf". From the Table 2, we can get following information. According to sealed bidding information, $deal price$ is 2.04, but provider 7 submitted 3.43 in revealed bids, which is not consistent with bid 1.00 submitted in sealed bid. Therefore, our system considers the bidding of submitted by provider 7 as invalid bidding which cannot be allocated. Provider 5 submitted his bid in sealed bidding stage, but did not submit bid in revealing sealed bids stage, so his submission is also invalid. Provider 4 is also not allocated, even though his sealed bid is consistent with revealed bid, but random string is not consistent.

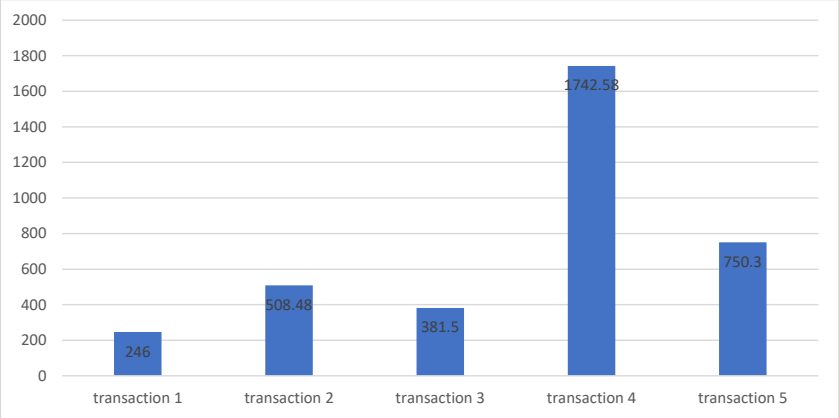

Figure 4. The total cost of demander in our mechanism.

Table 2. Bidding information and allocation results of providers in transaction 2.

Provider's Number	Sealed Bid	Revealed Bid	Allocation	Payment
1	6.41edf	6.41edf	0	0
2	3.72her	3.72her	0	0
3	1.89stb	1.89stb	1	127.12
4	2.27gjv	2.27guh	0	0
5	7.32rhy	–	0	0
6	1.58fdv	1.58fdv	1	127.12
7	1.00lfr	3.43lfr	0	0
8	8.69dev	8.69dev	0	0
9	2.04adg	2.04adg	1	127.12
10	1.27plg	1.27plg	1	127.12

6. Conclusions

With the continuous development and popularization of cloud storage technology, a secure, transparent and low-cost resource trading mechanism will become the inevitable direction of the development of cloud storage. This paper used the advantages of blockchain technology to solve the cloud storage transaction problem, and proposed a decentralized cloud storage transaction mechanism which was designed based on reverse VCG auction. In proposed decentralized trading method, the process of transaction is divided into four stages: publishing requirement, submitting sealed bids, revealing sealed bids and auction, payment which are implemented by four functions using the solidity language in blockchain environment. We have successfully deployed smart contract on the Ethereum private chain, and the smart contract can implement the actual platform to realize trading of cloud storage resource transaction.

Our work is a preliminary attempt to decentralized storage transaction with blockchain technology. Firstly, our system only implemented VCG-based mechanism with smart contract on private chain

taking into consideration the cost and security of storage transaction, and the other factors of transaction will be considered in future work such as reputation, QoS, etc. Secondly, in this paper we did not distinguish the value of availability, and each provider can participate the mechanism with same valuation only if his value of availability is higher than a predefined value. In future work we will investigate to how to select the providers with truthful mechanisms considering both availability and cost.

Author Contributions: Formal analysis, X.W.; Supervision, Y.G.; Validation, J.T.; Writing—original draft, D.H.; Writing—review & editing, X.W. and Y.Z.

Funding: This work has been supported by National Natural Science Foundation of China (No. 61170029), Zhejiang Provincial Natural Science Foundation of China (No. LY16F020015), Zhejiang Provincial Science and Technology Key Plan of China (No. 2017C02036), and Huzhou Science and Technology Key Plan of China (No. 2016ZD2011).

Acknowledgments: The authors appreciate the technical support from BASICS Lab of Shanghai Jiaotong University, and Electronic Commerce Research Institute of Huzhou.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Esposito, C.; Ficco, M.; Palmieri, F.; Castiglione, A. Smart Cloud Storage Service Selection Based on Fuzzy Logic, Theory of Evidence and Game Theory. *IEEE Trans. Comput.* **2016**, *65*, 2348–2362. [CrossRef]
2. Nabi, M.; Toeroe, M.; Khendek, F. Availability in the cloud: State of the art. *J. Netw. Comput. Appl.* **2016**, *60*, 54–67. [CrossRef]
3. Shahrad, M.; Wentzlaff, D. Availability Knob: Flexible User-Defined Availability in the Cloud. In Proceedings of the 7th ACM Symposium on Cloud Computing, Santa Clara, CA, USA, 5–7 October 2016; pp. 42–56.
4. Abdulatif, F.A.; Zuhiar, M. Cloud Security Issues and Challenges: Important Points to Move towards Cloud Storage. *Int. J. Sci. Res.* **2017**, *6*, 2105–2112.
5. Davoli, A.; Mei, A. Triton: A peer-assisted cloud storage system. In Proceedings of the First Workshop on Principles and Practice of Eventual Consistency, Amsterdam, The Netherlands, 13 April 2014; p. 4.
6. Zhao, J.; Chu, X.; Liu, H.; Leung, Y.W. Online procurement auctions for resource pooling in client-assisted cloud storage systems. In Proceedings of the 2015 IEEE Conference on Computer Communications (INFOCOM), Hong Kong, China, 26 April–1 May 2015; pp. 576–584.
7. Mager, T.; Biersack, E.; Michiardi, P. A measurement study of the Wuala on-line storage service. In Proceedings of the IEEE International Conference on Peer-To-Peer Computing, Tarragona, Spain, 3–5 September 2012; pp. 237–248.
8. Computer Sciences Corp. Available online: http://www.csc.com/ (accessed on 1 November 2018).
9. Fu, H.; Li, Z.; Wu, C.; Chu, X. Core-Selecting Auctions for Dynamically Allocating Heterogeneous VMs in Cloud Computing. In Proceedings of the IEEE International Conference on Cloud Computing, Anchorage, AK, USA, 27 June–2 July 2014; pp. 152–159.
10. Toka, L.; Dell'Amico, M.; Michiardi, P. Online Data Backup: A Peer-Assisted Approach. In Proceedings of the IEEE Tenth International Conference on Peer-To-Peer Computing, Delft, The Netherlands, 25–27 August 2010; pp. 1–10.
11. Xiang, Y.; Lan, T.; Aggarwal, V.; Chen, Y.F.R. Optimizing Differentiated Latency in Multi-Tenant, Erasure-Coded Storage. *IEEE Trans. Netw. Serv. Manag.* **2017**, *14*, 204–216. [CrossRef]
12. David Vorick, L.C. Sia: Simple Decentralized Storage. Available online: https://sia.tech/sia.pdf (accessed on 14 November 2018).
13. FUNATOZ. IPFS FIGTOO Storage. Available online: https://cdn.thiwoo.com/RedChain/reeed_white.pdf (accessed on 14 November 2018).
14. Storj Labs, Inc. Storj: A Decentralized Cloud Storage Network Framework. Available online: https://storj.io/storj.pdf (accessed on 14 November 2018).
15. Bessani, A.; Correia, M.; Quaresma, B.; Andr, F.; Sousa, P. DepSky: Dependable and Secure Storage in a Cloud-of-Clouds. In Proceedings of the European Conference on Computer Systems, EuroSys'11, Salzburg, Austria, 10–13 April 2011.

16. Chen, H.C.H.; Hu, Y.; Lee, P.P.C.; Tang, Y. NCCloud: A Network-Coding-Based Storage System in a Cloud-of-Clouds. *IEEE Trans. Comput.* **2014**, *63*, 31–44. [CrossRef]
17. Mu, S.; Chen, K.; Gao, P.; Ye, F.; Wu, Y.; Zheng, W. μLibCloud: Providing High Available and Uniform Accessing to Multiple Cloud Storages. In Proceedings of the 2012 ACM/IEEE 13th International Conference on Grid Computing, Beijing, China, 20–23 September 2012; pp. 201–208.
18. Lin, H.Y.; Tzeng, W.G. A Secure Erasure Code-Based Cloud Storage System with Secure Data Forwarding. *IEEE Trans. Parallel Distrib. Syst.* **2012**, *23*, 995–1003.
19. Papaioannou, T.G.; Bonvin, N.; Aberer, K. Scalia: An adaptive scheme for efficient multi-cloud storage. In Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis, Salt Lake City, UT, USA, 11–15 November 2012; p. 20.
20. Badanidiyuru, A.; Kleinberg, R.; Singer, Y. Learning on a budget: Posted price mechanisms for online procurement. In Proceedings of the 13th ACM Conference on Electronic Commerce, Valencia, Spain, 4–8 June 2012; pp. 128–145.
21. Zhao, D.; Li, X.Y.; Ma, H. How to crowdsource tasks truthfully without sacrificing utility: Online incentive mechanisms with budget constraint. In Proceedings of the IEEE International Conference on Computer Communications (INFOCOM'14), Toronto, ON, Canada, 27 April–2 May 2014; pp. 1213–1221.
22. Su, M.; Zhang, L.; Wu, Y.; Chen, K.; Li, K. Systematic Data Placement Optimization in Multi-Cloud Storage for Complex Requirements. *IEEE Trans. Comput.* **2016**, *65*, 1964–1977. [CrossRef]
23. Storj. Available online: https://storj.io (accessed on 1 November 2018).
24. N2 Technology Corp. Available online: http://www.n2yun.com/index.php/n2c (accessed on 1 November 2018).
25. Narahari, Y. *Game Theory and Mechanism Design*; World Scientific Publishing: Singapore, 2014.
26. Nakamoto, S. Bitcoin: A Peer-to-Peer Electronic Cash System; Technical Report, 2008. Available online: https://bitcoin.org/bitcoin.pdf (accessed on 14 November 2018).
27. Szabo, N. Formalizing and Securing Relationships on Public Networks. *First Monday* **1997**, *2*, 9. [CrossRef]
28. Li, M.; Weng, J.; Yang, A.; Lu, W. CrowdBC: A Blockchain-based Decentralized Framework for Crowdsourcing. *IACR Cryptology ePrint Archive* **2017**, *2017*, 444. [CrossRef]