TSINGHUA SCIENCE AND TECHNOLOGY
ISSN 1007-0214 08/10 pp964–972
DOI: 10.26599/TST.2021.9010069
Volume 27, Number 6, December 2022

Self-Renewal Consortium Blockchain Based on Proof of Rest and Strong Smart Contracts

Wenyu Shen, Xuebing Huang, Yunshan Fu, Yongwei Hou, and Li Ling*

Abstract: Focusing on the business alliance scenario in blockchains, this paper proposes a new consensus mechanism named proof of rest (PoR) and strong smart contracts. The block structure and logic of PoR consensus are described. And a consortium blockchain system supporting strong smart contracts is designed. We modify the difficulty value algorithm based on proof of work (PoW) and add adjustable parameters. The longer a node rests after creating a block, the less difficult it is to create another new block, hence the term PoR. The penalty for slack nodes, the joining and quitting of nodes, and the adjustment of the expected block creation time can all be accomplished using the strong smart contracts, so the consortium blockchain can realize self-renewal.

Key words: blockchain; consortium chain; proof of rest (PoR); smart contract; self-renewal

1 Introduction

The core technology of blockchains is essentially cryptographic algorithms and consensus mechanisms. The rudiments of these technologies have appeared as early as the last century and are gradually maturing.

This paper focuses on consortium blockchains and proposes a new consensus mechanism called proof of rest (PoR), which has certain authorization and access conditions. Blocks can be quickly created with a little competition introduced. Moreover, the system throughput is considerable, and the network requirement for reaching a consensus is not high, so it has some advantages in the business alliance scenario.

Nodes participating in a consortium blockchain are controlled by different organizations, which form an industry alliance to manage the blockchain. The industry alliance shall hold joint meetings on major issues for consultation and solution. The alliance may split if the agreement is not reached, which would be a hard fork for the blockchain. In this case, the alliance no longer exists, so this paper assumes that any issues can be resolved through negotiations within the alliance.

In general, the industry alliance is the highest decision-making group and actual controller of the consortium blockchain. However, the system cannot be regarded as a centralized system. The members of the industry alliance negotiate and vote with one another through joint meetings, and no single member can control the whole alliance. Thus, it is essentially a decentralized system. From the perspective of blockchains, nodes belong to different organizations that guard against one another. In fact, industry alliance and consortium blockchain with access threshold, multi-party negotiation, and certain precautions in the implementation can be considered a weakly centralized system.

On this basis, we have designed smart contracts to control the update of consensus mechanism parameters, so as to achieve the self-renewal of the blockchain and absolute control of the industry alliance for the consortium blockchain.

2 Overview

2.1 Cryptology theory

The basic theories of cryptography have been mature in the last century, mainly including symmetric encryption algorithms (e.g., the data encryption standard (DES)[1]

• Wenyu Shen, Xuebing Huang, Yunshan Fu, Yongwei Hou, and Li Ling are with the School of Information Science and Technology, Fudan University, Shanghai 200433, China. E-mail: wyshen18@fudan.edu.cn; xbhuang19@fudan.edu.cn; ysfu17@fudan.edu.cn; ywhou17@fudan.edu.cn; lingli@fudan.edu.cn.

* To whom correspondence should be addressed.
Manuscript received: 2021-04-15; revised: 2021-09-12; accepted: 2021-09-21

issued by the United States Federal Standard in 1977), asymmetric encryption algorithms (e.g., the Rivest-Shamir-Adleman (RSA)[2] algorithm proposed by R. L. Rivest, A. Shamir, and L. M. Adleman in 1978; the ellipse curve cryptography (ECC) proposed by V. S. Miller of IBM in 1985[3] and N. Koblitz of the University of Washington in 1987[4]; and the Elgamal[5] algorithm proposed by T. Elgamal in 1985), and one-way function encryption algorithms (e.g., the cyclic redundancy check (CRC)[6] algorithm published by W. W. Peterson an D. T. Brown in 1961; the message digest (MD)[7] algorithm published and upgraded by R. L. Rivest from 1989 to 1992; RACE integrity primitives evaluation message digest (RIPEMD)[8] algorithm proposed by H. Dobbertin, A. Bosselaers, and B. Preneel on the basis of MD4 in 1996; and the secure hash standard (SHA)[9] designed by the National Security Agency and published by the National Institute of Standards and Technology in 1995).

2.2 Classification of blockchains

Blockchain systems can be divided into the following categories based on the access permission: private blockchain that is fully controlled by a single person or organization, consortium blockchain with certain thresholds, and public blockchain that allows anyone to join.

All nodes of a private blockchain are completely controlled by an organization or institution. The owner of a private blockchain can easily manipulate the entire system, making the blockchain more like a distributed database. A private blockchain can choose various consensus mechanisms, but due to its private characteristics, there is generally no need for competition among nodes. Hence, a classical distributed consensus[10] is the most convenient way to complete state machine replication. Typical schemes include Paxos[11], practical Byzantine fault tolerance (PBFT)[12], and HotStuff[13].

A consortium blockchain is generally an alliance composed of multiple companies or organizations. Others need to be recognized by the existing members of the alliance when joining it. Within the consortium blockchain, nodes are still on guard against one another. From this point of view, it has no difference from the public blockchain, except that certain access permissions have been added. This condition makes it easier to exchange keys and the network topology becomes clearer, which cannot be performed in public blockchains. In the consortium blockchain, nodes actually belong to different controllers. Thus, it is still a decentralized system under the consensus mechanism, which is an advantage compared with the private blockchain. A consortium blockchain is generally set up with access permissions, which can be considered as an authorization consensus. Typical schemes include Hyperledger[14], DFINITY[15], and Pala[16].

The best known public blockchain systems are Bitcoin and Ethereum, which are motivated by virtual currencies. The design concept of a public blockchain is that anyone can join or quit unconditionally. Therefore, there is no trust between all nodes, which is often accompanied by a certain waste of resources. The consensus mechanism in a public blockchain is generally considered to be an unauthorized consensus mechanism[17]. In addition to proof of work (PoW)[18] used by Bitcoin and proof of stake (PoS)[19] used by Ethereum, there are also mixed consensus mechanisms of a single committee or multiple committees. A single-committee mixed consensus mechanism first uses the PoW or PoS to select some nodes as the committee, and then runs the distributed consensus algorithm similar to PBFT within the committee to complete block generation. Typical schemes include ByzCoin[20], AlgoRand[21], and Solida[22]. The multi-committee mixed consensus mechanism divides the network into multiple zones, each of which runs the internal committee processing transactions in parallel. Typical schemes include Elastico[23] and RapidChain[24].

In general, the consensus mechanism can be divided into permissioned consensus, permissionless consensus, and classical distributed consensus. The PoR designed in this paper belongs to permissioned consensus.

2.3 Network environment

In this paper, we assume that there will be no large-scale network interruption, the network topology is a complete graph, the IP addresses and ports of all nodes are known, and the public keys have been exchanged among the nodes. Moreover, RSA encryption cannot be cracked, and SHA256 is irreversible. We allow a certain number of evildoers to occur in the consortium, but by default more than half of the nodes are honest. Industry alliances can decide whether to increase or decrease members, or simply exclude evildoers.

3 PoR Consensus Mechanism

3.1 Proposal of PoR

In PoW, a system issues certain rewards to the nodes

that create blocks, but it does not want the rewards to be always obtained by certain nodes. Therefore, a higher degree of difficulty is set, such that all nodes compete with one another, resulting in a waste of computing power[18]. The consortium blockchain allows a certain amount of waste, but it is never willing to accept such a large internal loss. PBFT needs to forward data twice[12] to confirm whether the block creator has sent multiple distinct blocks to other nodes. This process increases the requirements for the network and should be avoided as much as possible. Fortunately, the simultaneous appearance of multiple latest blocks in PoW has become commonplace. On this basis, we propose a new consensus mechanism called PoR.

In the setting of our system, creating blocks is an obligation to join the consortium. Therefore, the consortium blockchain can be completely free of rewards, and there is no need to waste computing power. In this case, the difficulty requirement can be set relatively low to make it easier to create blocks. However, one node cannot control the entire blockchain by continuously creating blocks with its strong computing power. Under these conditions, we designed a new consensus mechanism called PoR. Compared with PoW, PoR has a great improvement in the setting of difficulty values, which fully meets the requirements of moderate competition and no monopoly. In terms of the formation and resolution of forks, the idea of PoW is still adopted. Multiple nodes work at the same time and publish their own blocks, but nodes will choose the longest fork. When the length is the same, the forks received earlier on the timeline are inclined. The basic idea of the PoR consensus is simple: to set a larger mining difficulty for nodes that have recently created a block and, on the contrary, set a smaller difficulty. The closer the block created last time, the more difficult it is to create a new one. If the node rests temporarily for a period of time, then the difficulty of creating a new block will be greatly reduced, so it is called PoR.

In Bitcoin, the difficulty of creating a block is determined by a four-byte field called "bits"[18]. The difficulty value at this stage depends on the computing power of the entire system and the block generation time of the last stage, which controls the time required to create a new block to approximately 10 min. Our blockchain system sets n nodes, numbered from 1 to n. The difficulty for node k to create block m is determined by a certain number of blocks before m. We define the sequence number difference of blocks on a blockchain as the distance. We stipulate that for block m, blocks within a distance of $2n$ will affect it, which are block $m-1, m-2, \ldots, m-2n$. The weight of each block is linearly distributed after taking the logarithm twice. In terms of hash collision, the probability exponentially decreases.

Taking into consideration the business requirements and actual computing power of the alliance, we designed the adjustable parameters Max and Min. The two values decide the maximum and minimum impacts of the historical block on the difficulty value of the latest block. Usually, we set $7 \leqslant Max \leqslant 8, 0 < Min < Max - 1$. If Min becomes larger, then Max should also be further increased.

$\lfloor x \rfloor$ represents the largest integer not exceeding x, and $\lceil x \rceil$ represents the smallest integer not less than x. We define the difficulty value of node k when creating block m as $DT_{(k,m)}$. And we stipulate that the creator of block i is represented by B_i. Then $DT_{(k,m)}$ is defined as follows:

$$Hash(block\ header) < DT_{(k,m)} \tag{1}$$

$$DT_{(k,m)} = \lceil 2^{W_{(k,m)}} \rceil \tag{2}$$

$$W_{(k,m)} = 256 - \sum_{i=m-2n}^{m-1} 2^{T_{m-i}} \cdot C_{(k,i)} \tag{3}$$

$$T_{m-i} = Max - (m-i) \cdot \delta \tag{4}$$

$$\delta = \frac{\left\lfloor \frac{Max-Min}{(2n-1)} \cdot fix\_ratio \right\rfloor}{fix\_ratio} \tag{5}$$

$$fix\_ratio = 10^{fix\_len} \tag{6}$$

$$fix\_len = \lceil \lg 2n \rceil \tag{7}$$

$$C_{(k,i)} = \begin{cases} 1, & k = B_i; \\ 0, & k \neq B_i \end{cases} \tag{8}$$

where $Min \leqslant T_{2n} < Max$. T_{2n} and Min are not completely consistent, and there is a slight error between them. However, this does not affect the basic idea that the weight of the historical blocks is linearly distributed after taking the logarithm twice. A consensus can still be reached if all nodes follow these algorithms. This error is meant to reduce the number of significant digits after the decimal point and the calculation amount to derive $DT_{(k,m)}$. fix_len and fix_ratio are meant to ensure that δ must not be 0. The so-called linearity is by no means exactly equal.

In other words, as long as a node can observe the creation of block m and the previous $2n$ block, it can calculate the difficulty value of B_m in creating block

m. Therefore, the four-byte difficulty value field can be omitted in the block structure to save storage space.

In this way, each node has a different difficulty value for creating the same latest block. The longer it is since the node created the block last time, the smaller its difficulty value is. Moreover, any node that receives the block can quickly determine the difficulty value of the current block creator by tracing upwards, and then check whether it meets the requirements.

3.2 Weight of historical blocks

A node needs to select a random number (nonce) when creating a block, and then computes the hash of the block header. If $hash < DT_{(k,m)}$, then the block is successfully created. This is actually an enumeration process. Every time the hash is computed, there is a certain probability of success, which is related to the difficulty value. We select different numbers of nodes and different Max and Min parameters to calculate the impact of each historical block on creating a new block. The probability of creating a block is

$$P = \frac{DT_{(k,m)}}{2^{256}} = \frac{\lceil 2^{W_{(k,m)}} \rceil}{2^{256}} \approx \frac{2^{256-\sum_{i=m-2n}^{m-1} 2^{T_{m-i}} \cdot C_{(k,i)}}}{2^{256}} = 2^{-\sum_{i=m-2n}^{m-1} 2^{T_{m-i}} \cdot C_{(k,i)}} \tag{9}$$

$$\lg P \approx -\sum_{i=m-2n}^{m-1} 2^{T_{m-i}} \cdot C_{(k,i)} \cdot \lg 2 \tag{10}$$

In this paper, when the total number of nodes is n, the historical block that has an impact is $2n$ blocks. If the computing power of each node is similar, then two or three historical blocks will have an effect on the difficulty value of each node. These historical blocks are basically unlikely to be close in distance. It can be roughly estimated that each node takes turns to create blocks.

Among the $2n$ blocks, the newer n blocks weigh much more than the older ones. The reason why $2n$ is chosen instead of n is that there is no guarantee that a node can create a new block every n blocks. A certain deviation is entirely possible. Only nodes that do not contribute to the system at all may fail to create blocks for $2n$ consecutive blocks. The consortium blockchain automatically punishes these nodes and removes them from the alliance. In the fourth part of this paper, we will explain how to use smart contracts to control the nodes in detail. Therefore, given n, Max, and Min, we only need to study the effect of the new n blocks on the difficulty value of the latest block.

The number of nodes, and the values of Max and Min can be selected according to the alliance situation, business requirements, and expected block creation time. Figures 1–3 only list a few different cases.

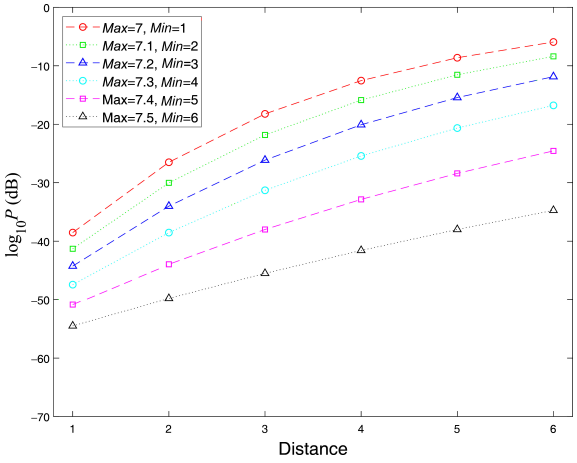

Fig. 1 Weight of historical blocks when $n = 6$.

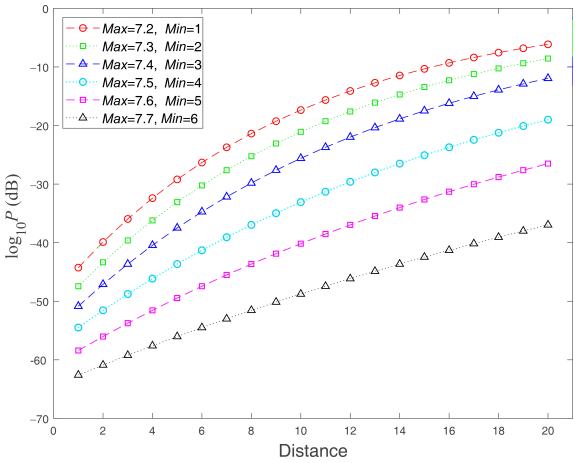

Fig. 2 Weight of historical blocks when $n = 20$.

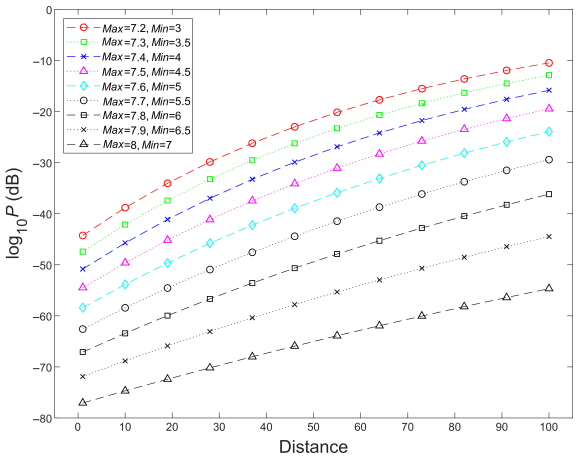

Fig. 3 Weight of historical blocks when $n = 100$.

3.3 Weight impact analysis

For node k that is the easiest to create the new block m in the entire system, the distances of the historical blocks that are most likely to affect the creation of block m are n and $2n$. So according to Eqs. (3) and (8),

$$C_{(k,i)} = \begin{cases} 1, & i = m-n \text{ or } i = m-2n; \\ 0, & \text{else} \end{cases} \tag{11}$$

$$W_{(k,m)} = 256 - 2^{T_n} - 2^{T_{2n}} \tag{12}$$

Since $2^{T_{2n}} \ll 2^{T_n}$, $W_{(k,m)} \approx 256 - 2^{T_n}$.

In this way, the relationship between the computing power and the creation time of a new block can be estimated.

We assume that the consortium blockchain expects to create a block every 1 s. Then we can use a scheme similar to that of Bitcoin to estimate the demand for computing power.

$$Difficulty = \frac{Bdiff}{Target} \tag{13}$$

$$Hashrate = \frac{Difficulty}{Time} \tag{14}$$

$$Bdiff = 2^{256} \tag{15}$$

$$Target = DT_{(k,m)} \approx 2^{W_{(k,m)}} \tag{16}$$

$$W_{(k,m)} = 256 - 2^{T_n} \tag{17}$$

$$Hashrate = \frac{\frac{Bdiff}{Target}}{Time} \approx \frac{\frac{2^{256}}{2^{256-2^{T_n}}}}{1} = 2^{2^{T_n}} \tag{18}$$

T_n should be slightly greater than $(Max + Min)/2$. Table 1 lists the impact of different T_n on the hashrates under different initial conditions.

At present, the performance of the graphics processing unit (GPU) far exceeds that of the central processing unit (CPU) in the computing hash. The general GPU computing power is approximately 20–30 MH (10^6 hashes/s), which is in the same order of magnitude as when T_n is 4.5. If T_n reaches 5, then the investment cost will be higher.

Table 1 Hashrates at different T_n.

T_n	Hashrate (hashes/s)
2	16
2.5	50
3	256
3.5	2545
4	65 536
4.5	6.5×10^6
5	4.3×10^9
5.5	3.2×10^{13}
6	1.85×10^{19}

Therefore, by setting the appropriate Max and Min according to the business demand and overall computing power, it can fully meet our design requirements of moderate competition without wasting much computing power.

If node g in the alliance wants to create blocks faster than node k, then node g needs more computing power because it is obviously harder to create blocks than node k. The cost of doing so is very high while the ultimate benefit may be minimal.

When the number of the nodes in the system is small, the difficulty of creating blocks for each node is particularly significant due to the PoR consensus mechanism. With proper Max and Min, one can only rely on luck to create a block preemptively. With a larger number of nodes, the difficulty difference is less, and there would be a concurrence of several nodes that can easily create a new block. At this point, the value of Min can be fine-tuned upward so that the expected creating time of these nodes is slightly over 1 s, and there will always be a lucky node that will win out. Over a long period of time, the system still creates a new block per second on average.

In addition, with the development of computer hardware, the GPU computing power will gradually increase. In this case, only slight adjustments to Max and Min can directly eliminate the computing power gain brought about by hardware advancements, and maintain the block creation time at approximately 1 s. After all, the current single GPU computing power is only at the MH level. By slightly increasing the values of Max and Min, the computing power demand can be greatly increased to the GH (10^9 hashes/s) or even TH (10^{12} hashes/s) level, so the impact of hardware development can be ignored in a short period.

3.4 New problems

This study modifies the calculation method of the difficulty value, such that a node with a lower difficulty requirement can have many choices of nonce that can meet the difficulty value requirements when creating a block. We present a few new issues to address in the following sections.

3.4.1 Available storage space

For a node with a small difficulty value, it is easy to create a new block in tens or even milliseconds. The new block contains only a few transactions, but its block header has a fixed length. With the same length of time and number of transactions, the larger the number of

blocks, the more block header data that need to be stored, thus wasting the available storage space.

In response to this problem, we can set a time threshold and transaction volume threshold. If the number of transactions exceeds the threshold, then the block can be created immediately. Otherwise, if the number of transactions is not enough, then the block is created after the time threshold is reached to ensure that the few transactions can still be recorded in the block in time without waiting too long. In fact, long-running blockchain systems will always encounter storage space problems. In the early stage of a blockchain operation, with not so many transactions, there will be a problem that the block header accounts for an excessively large proportion.

In our design, the block header has been reduced as much as possible, and the magic number and difficulty value that can be computed from historical blocks have been removed to minimize the waste of storage space.

3.4.2 Handling of forks

Based on the inequality of the difficulty value, the block creator can create multiple different blocks that all meet the requirements and scatter them to different nodes to achieve malicious forks. In the Bitcoin system, different nodes create different blocks and spread them throughout the network all the time. The blockchain will split briefly. Sometimes, there are multiple forks, and even forks would split. However, each node will create blocks after the longest fork. If a fork occurs earlier, then the probability for it to be accepted by the nodes is higher. In other words, the greater the computing power a fork obtained, the greater the probability that the next block will be generated after it, which makes it longer. Thus, nodes will eventually reach a consensus on the longest fork. Our system still follows the principle of the longest chain; that is, the earlier generated fork can attract more nodes, so it is more likely to win. However, this is not necessarily the case because the difficulty of creating blocks varies from node to node, and the next block would be created by nodes with low difficulty values. While the time it takes for a fork to appear on the timeline affects its survival probability, the fork has a higher probability of surviving in the competition if it is first received by nodes with low difficulty values. That is, the influence of the reception time on the survival probability is sometimes greater than the appearance time. Introducing the uncertainty makes the blockchain run more healthily. In general, the PoR consensus mechanism can achieve ultimate consistency and thus still have the ability to resist forks.

3.4.3 Data tampering

Any node can also tamper with blocks published by others by modifying the nonce field. Although this does not directly benefit the node, it is a potential threat to the entire system and should be avoided.

In this study, the signature of the block creator is added to the related block header. The integrity and non-repudiation of data are guaranteed by cryptography, which ensures that the block created by the creator cannot be tampered with by other nodes at will. The inspiration for this design comes from Fabric[14], which is not available in Bitcoin systems.

Blocks are correlated by the hash of the previous block header. The Merkle tree root is used to achieve binding between the block header and block body. The information of the creator is stored in the transaction. The difficulty value is calculated by the historical blocks. The integrity of the block is guaranteed by the asymmetric encryption algorithm. The above series of relationships forms a closed loop. As long as the hash is irreversible and the RSA algorithm cannot be cracked, the block information can be guaranteed to be valid, tamper-proof, and undeniable.

3.4.4 Waste of computing resources

According to the formula of the difficulty value, it is quite difficult for a node that has already created a block to create a new block in the following period of time. As is basically impossible, it is actually a waste of computing power to keep trying. Therefore, our system allows a node to take a rest after creating a block. During this period, the node will only collect transactions and verify the blocks published by other nodes, rather than trying to create a block. This is how PoR comes. When the node is resting, other nodes are actually creating blocks, so in a sense, each node still serves the entire alliance in turn. The rest time can be comprehensively considered and determined according to the number of nodes, business requirements, network conditions, and expected block creation speed. However, generally, it should not be too long. The node should be restored to work before it becomes one of the easiest nodes to create blocks in the system.

3.5 Structure of the block

We design the block structure of the consortium blockchain as shown in Fig. 4.

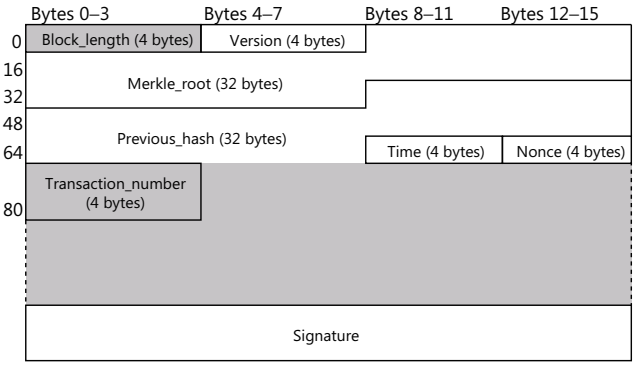

Fig. 4 Block structure.

The length of signature is determined by key length. The block is divided into four parts: total length of the block, block header, block body, and signature. The block header contains version (4 bytes), previous block header hash (32 bytes), Merkle root (32 bytes), timestamp (4 bytes), and counter (4 bytes). The block body contains the transactions that have been verified by the node. The first four bytes of the body are used to indicate the number of transactions and then the specific transaction content. The information of the block creator is recorded in the first transaction, followed by other transactions. A transaction includes the timestamp, initiator, signature, and smart contracts. The signature of the block creator is added at the end of the block header, whose length is equal to that of the RSA private key[2].

4 Strong Smart Contracts

Smart contracts were initially proposed by Nick Szabo in 1995, which he defined as follows: "A smart contract is a computerized transaction protocol that executes the terms of a contract."[25]

Currently, programming languages for writing smart contracts usually include Solidity, Serpent, and Lisp like languages. A variety of smart contracts are deployed in Ethereum. For a blockchain, a smart contract is the data stored in it. For a client, a smart contract is a piece of code that can be run. In general, the client regularly checks whether the smart contract can be successfully executed, and the time interval is determined according to actual needs.

We hope to realize the self-renewal of the blockchain, i.e., update some parameters of the chain, such as Max and Min in the difficulty value algorithm, IP, and port of the core nodes. In this way, a smart contract stored in the consortium blockchain can in turn affect the blockchain itself. To distinguish it from the general smart contracts that cannot affect the blockchain itself, but only intra-chain transactions, we call these smart contracts "strong smart contracts". This paper gives the strong smart contracts more powerful capabilities and stronger restrictions as well. The goal of strong smart contracts is to be able to update the chain based on the chain itself, and even gradually update all parts like the ship of Theseus.

We designed three kinds of strong smart contracts in the consortium blockchain. The first type is to remove the node that does not create a block after $2n$ consecutive blocks. The second is to increase or decrease the number of participating nodes. The third is to update the Max and Min parameters. Among them, the first type of strong smart contracts does not need to be released. In fact, the entire consortium blockchain can be considered a smart contract. Every time a block is created, it will check whether any node is punished for being slack. The second and third types of smart contracts require the agreement of all nodes in the alliance before they can be executed. In fact, the entire alliance can only modify parameters and increase or decrease participating nodes by convening a joint meeting and reaching an agreement while the smart contracts are just the implementation of the decision.

When a node creates a new block, it stores the information that needs to be modified and its own signature into the blockchain, indicating a vote. Other nodes also store their signatures in the consortium chain to indicate an agreement. After the agreement of all nodes, the strong smart contract should be executed immediately. We cannot use the scheme of a periodic inspection because it is difficult to find the appropriate time interval to ensure the immediate implementation of the contract. The consumption of resources is too large if the interval is too short, which is not necessary.

We specify that the first type of strong smart contracts has the highest priority, which means that the system punishes the node that does not contribute at any time. Industry alliance will not add or remove participating nodes or update parameters multiple times within a day. Thus, for the second and third types of smart contracts, only one contract is valid at the same time. Each initiated contract has a validity period. If sufficient consent votes cannot be obtained within the validity period, then the strong smart contract will be abandoned.

We indicated many restrictions on strong smart contracts because such smart contracts can affect the entire consortium chain, which is completely different from general smart contracts. Thus, it is necessary

to provide more restrictions to ensure a smooth operation of the blockchain and reduce the impact of malicious behaviors. In this way, we can use strong smart contracts to control the update of consensus mechanism parameters, and to achieve self-renewal of the blockchain. The operation diagram of strong smart contracts in shown in Fig. 5.

5 Conclusion

We used Alibaba Cloud and Tencent Cloud to build the consortium blockchain. All data were encrypted to prevent malicious eavesdropping and transferred over the HTTP protocol. The blockchain is operating well. Although there are forks, a consensus can be quickly reached. Strong smart contracts can be used to change the parameters in the consortium chain, including Max and Min in the difficulty value algorithm and the number, IP, port, public key, and other contents of participating nodes.

We propose the PoR consensus mechanism and strong smart contracts that adapt to the scenario of consortium blockchain, so that each participating node takes turns to rest and create blocks. The system can automatically punish the slack nodes, update the parameters, and increase or decrease the number of nodes, which achieves the self-renewal of the consortium blockchain.

The basic version of the alliance chain is currently in trial operation in Shanghai Blockchain Technology Association for the storage and traceability of activities for Popular Science.

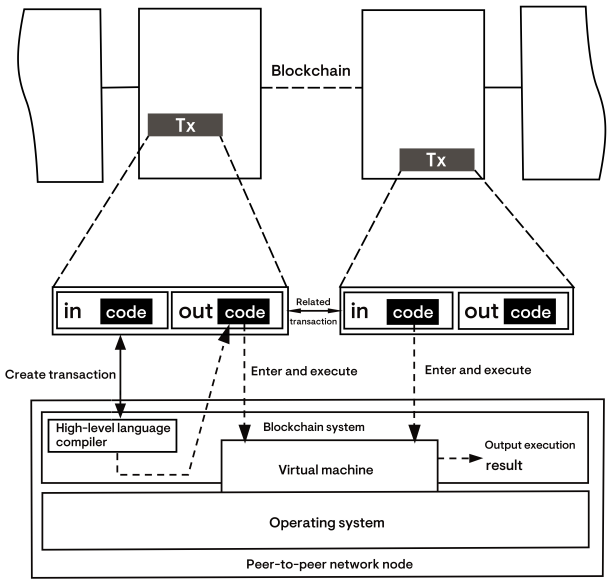

Fig. 5 Operation diagram of strong smart contracts.

Acknowledgment

This work was supported by the National Key R&D Program of China (No. 2018YFB2101100).

References

[1] NBO Standards, Data encryption standard, *Federal Information Processing Standards Publications*, pp. 632–646, 1977.

[2] R. L. Rivest, A. Shamir, and L. M. Adleman, A method for obtaining digital signatures and publickey cryptosystems, *Communications of the ACM*, vol. 21, no. 2, pp. 120–126, 1978.

[3] V. S. Miller, Use of elliptic curves in cryptography, in *Advances in Cryptology-CRYPTO 1985, Lecture Notes in Computer Science*, H. C. Williams, ed. Berlin, Germany: Springer-Verlag, 1986, pp. 417–426.

[4] N. Koblitz, Elliptic curve cryptosystems, *Mathematics of Computation*, vol. 48, no. 177, pp. 203–209, 1987.

[5] T. Elgamal, A public key cryptosystem and a signature scheme based on discrete logarithms, *IEEE Transactions on Information Theory*, vol. 31, no. 4, pp. 469–472, 1985.

[6] W. W. Peterson and D. T. Brown, Cyclic codes for error detection, *Proceedings of the IRE*, vol. 49, no. 1, pp. 228–235, 1961.

[7] R. L. Rivest, The md4 message digest algorithm, in *Proceedings of the 10^{th} Annual International Cryptology Conference on Advances in Cryptology, ser. CRYPTO'90*, A. Menezes and S. A. Vanstone, eds. Berlin, Germany: Springer-Verlag, 1990, pp. 303–311.

[8] H. Dobbertin, A. Bosselaers, and B. Preneel, Ripemd-160: A strengthened version of ripemd, in *Fast Software Encryption*, D. Gollmann, ed. Berlin, Germany: Springer-Verlag, 1996, pp. 71–82.

[9] Q. Dang, Secure hash standard, Federal information processing standards (NIST FIPS), https://doi.org/10.6028/NIST. FIPS.180-4, 1995.

[10] C. Cachin and M. Vukolić Blockchain consensus protocols in the wild, arXiv preprint arXiv: 1707.01873, 2017.

[11] L. Lamport, The part-time parliament, *ACM Trans. Comput. Syst.*, vol. 16, no. 2, pp. 133–169, 1998.

[12] M. Castro and B. Liskov, Practical byzantine fault tolerance，in *Proc. the 3^{rd} Symposium on Operating Systems Design and Implementation(OSDI)*, New Orleans, LA, USA, 1999, pp. 173–186.

[13] I. Abraham, G. Gueta, and D. Malkhi, Hot-stuff the linear, optimal-resilience, one-message BFT devil, arXiv preprint arXiv: 1803.05069, 2018.

[14] E. Androulaki, A. Barger, V. Bortnikov, C. Cachin, K. Christidis, A. D. Caro, D. Enyeart, C. Ferris, G. Laventman, Y. Manevich, et al., Hyperledger fabric: A distributed operating system for permissioned blockchains, in *Proc. the 13^{th} EuroSys Conference*, Porto, Portugal, 2018, pp. 1–15.

[15] T. Hanke, M. Movahedi, and D. Williams, Dfinity technology overview series, consensus system, arXiv preprint arXiv: 1805.04548, 2018.

[16] T. -H. H. Chan, R. Pass, and E. Shi, Pala: A simple partially

synchronous blockchain, *IACR Cryptol. ePrint Arch.*, no. 981, pp. 1–21, 2018.

[17] R. Pass and E. Shi, Rethinking large-scale consensus, in *Proc. IEEE 30th Computer Security Foundations Symposium (CSF)*, Santa Barbara, CA, USA, 2017, pp. 115–129.

[18] S. Nakamoto, Bitcoin: A peer-to-peer electronic cash system, https://bitcoin.org/bitcoin.pdf, 2008.

[19] V. Buterin, A next-generation smart contract and decentralized application platform, *Ethereum White Paper*, vol. 3, no. 37, pp. 1–36, 2014.

[20] E. Kokoris-Kogias, P. Jovanovic, N. Gailly, I. Khoffi, L. Gasser, and B. Ford, Enhancing bitcoin security and performance with strong consistency via collective signing, in *Proc. 25th USENIX Security Symposium* (*USENIX security 16*), Austin, TX, USA, 2016, pp. 279–296.

[21] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich, Algorand: Scaling byzantine agreements for cryptocurrencies, in *Proc. the 26th Symposium on Operating Systems Principles*, Shanghai, China, 2017, pp. 51–68.

[22] I. Abraham, D. Malkhi, K. Nayak, L. Ren, and A. Spiegelman, Solida: A blockchain protocol based on reconfigurable Byzantine consensus, arXiv preprint arXiv: 1612.02916, 2016.

[23] L. Luu, V. Narayanan, C. D. Zheng, K. Baweja, S. Gilbert, and P. Saxena, A secure sharding protocol for open blockchains, in *Proc. the 2016 ACM SIGSAC Conference on Computer and Communications Security*, Vienna, Austria, 2016, pp. 17–30.

[24] M. Zamani, M. Movahedi, and M. Raykova, Rapidchain: Scaling blockchain via full sharding, in *Proc. the 2018 ACM SIGSAC Conference on Computer and Communications Security*, Toronto, Canada, 2018, pp. 931–948.

[25] N. Szabo, Smart contracts, http://szabo.best.vwh.net/smart.contracts.html, 1995.

Wenyu Shen received the bachelor and master degrees from Fudan University, Shanghai, China in 2017 and 2021, respectively. His research interests include blockchain technology and digital signal processing. He is passionate about information science and technology.

Yunshan Fu received the bachelor degree in communication engineering from Fudan University, Shanghai, China in 2021. His research interests include software development and blockchain.

Li Ling received the master degree from Fudan University, Shanghai, China. He is a professor of School of Information Science and Technology, Fudan University, Shanghai, China. His main research interests include network communication and security, blockchain technology, big data analysis, and cloud computing. He has published *Deconstruction of Blockchain*, *Network Protocols and Network Security*, etc.

Xuebing Huang received the bachelor degree from Xiamen University, Xiamen, China in 2019. She is pursuing the master degree in Fudan University in Shanghai. Her research interests include blockchain technology, privacy protection, and digital image processing.

Yongwei Hou is pursuing the bachelor degree in Fudan University, Shanghai, China. His research interests include blockchain technology, smart contracts, and deep learning for image processing.