Computers, Materials & Continua
DOI: 10.32604/cmc.2022.026437
Article

Tech Science Press

PoEC: A Cross-Blockchain Consensus Mechanism for Governing Blockchain by Blockchain

Jieren Cheng[1,3], Yuan Zhang[2,3,*], Yuming Yuan[4], Hui Li[4], Xiangyan Tang[1,3], Victor S. Sheng[5] and Guangjing Hu[1,3]

[1]School of Computer Science and Technology, Hainan University, Haikou, 570228, China
[2]School of Cyberspace Security, Hainan University, Haikou, 570228, China
[3]Hainan Blockchain Technology Engineering Research Center, Haikou, 570228, China
[4]Hainan Huochain Tech Company Limited, Haikou, 570100, China
[5]Department of Computer Science, Texas Tech University, Lubbock, 79409, United States of America
*Corresponding Author: Yuan Zhang. Email: 13697665791@163.com
Received: 26 December 2021; Accepted: 02 April 2022

Abstract: The research on the governing blockchain by blockchain supervision system is an important development trend of blockchain technology. In this system there is a supervisory blockchain managing and governing the supervised blockchain based on blockchain technology, results in a uniquely cross-blockchain demand to consensus mechanism for solving the trust problem between supervisory blockchain and supervised blockchain. To solve this problem, this paper proposes a cross-blockchain consensus mechanism based on smart contract and a set of smart contracts endorse the cross-blockchain consensus. New consensus mechanism called Proof-of-Endorse-Contracts (PoEC) consensus, which firstly transfers the consensus reached in supervisory blockchain to supervised blockchain by supervisory nodes, then packages the supervisory block in supervisory blockchain and transmits it to the smart contract deployed in the supervised blockchain, finally miners in supervised blockchain will execute and package the new block according to the status of the smart contract. The core part of the consensus mechanism is Endorse Contracts which designed and implemented by us and verified the effectiveness through experiments. PoEC consensus mechanism and Endorse Contracts support the supervised blockchain to join the governing blockchain by blockchain system without changing the original consensus mechanism, which has the advantages of low cost, high scalability and being able to cross-blockchain. This paper proves that our method can provide a feasible cross-blockchain governance scheme for the field of blockchain governance.

Keywords: Proof-of-endorse-contracts; PoEC; cross-blockchain consensus mechanism; governing blockchain by blockchain

1 Introduction

1.1 Background

Blockchain is a technology that is maintained by multiple parties jointly and uses cryptography to ensure the security of transmission and access, which can achieve data consistency, immutability and non-repudiation [1], realizing the transformation from information Internet to value Internet [2]. Blockchain is becoming another emerging technology that has a significant impact after big data, cloud computing, artificial intelligence, virtual reality, etc. [3]. Its application has extended from the initial digital currency to finance, Internet of Things, intelligent manufacturing and other fields, attracted extensive attention from the industry and governments of various countries [4].

Blockchain supervision technology is the key to ensure the healthy and sustainable development of blockchain technology and industry [5]. However, the anonymity of blockchain network, the immutability of blockchain transactions and the distributed authority of blockchain nodes make it difficult to govern the abnormal and illegal behaviors of blockchain [6]. For example, the governance measures taken after The DAO project suffered a reentrant attack directly led to the hard forks of Ethereum [7]. This shows that the traditional governance method oriented a centralized architecture cannot govern the public blockchain well, and a powerful governance method that can transform the consensus of the governing body into the governing object is urgently needed, that is, the consensus of the public blockchain network. It is simply imagined that this process must require the deep involvement of blockchain technology. Therefore, Academician Chen Chun of The Chinese Academy of Engineering pointed out that the research on the supervision system of governing blockchain by blockchain is an important development trend of blockchain supervision technology at The Blockchain Technology Conference 2019CCF [8].

1.2 Motivations, Problems and Challenges

The governing blockchain by blockchain regulatory system refers to transforming the blockchain governance scheme into the code-based rule paradigm in the blockchain network with the help of consensus mechanism, smart contract, incentive mechanism and other key blockchain technologies [9], to coordinate the legitimate interest demands of non-specific subjects. It supports supervisory nodes included in supervised blockchain to form supervisory blockchain, to review and govern smart contracts and node transactions in accordance with established rules, and to cancel and impose fines on non-compliant contracts or transactions [10], thus effectively reviewing, monitoring and governing the blockchain community, which is shown in Fig. 1.

In general, the supervisory blockchain is a permissioned consortium chain or private chain composed of regulatory nodes, with access mechanism and long-term fixed access. The network model is synchronous network or partially synchronous network. The supervised blockchain is usually a permissionless public chain, and in a few cases, it is a permissioned consortium chain. Nodes typically have no access mechanism and free to join or leave. The network model is asynchronous network or partially synchronous network [11]. Tab. 1 summarizes the main differences between the supervisory and supervised chains in eight aspects.

The heterogeneous characteristics of the supervising blockchain and the supervised blockchain in terms of access, node connection, network model and other aspects bring great challenges to the traditional consensus mechanism designed for homogeneous blockchain. In particular, the hierarchical cross-blockchain regulatory model puts forward a unique demand for cross-blockchain consensus mechanism, which makes the existing consensus mechanism unable to be directly applied to the

governing blockchain by blockchain framework. Therefore, it is urgent to study the scalable cross-blockchain consensus mechanism oriented to the governing blockchain by blockchain framework to solve the applicability problem of the closed single-chain consensus mechanism.

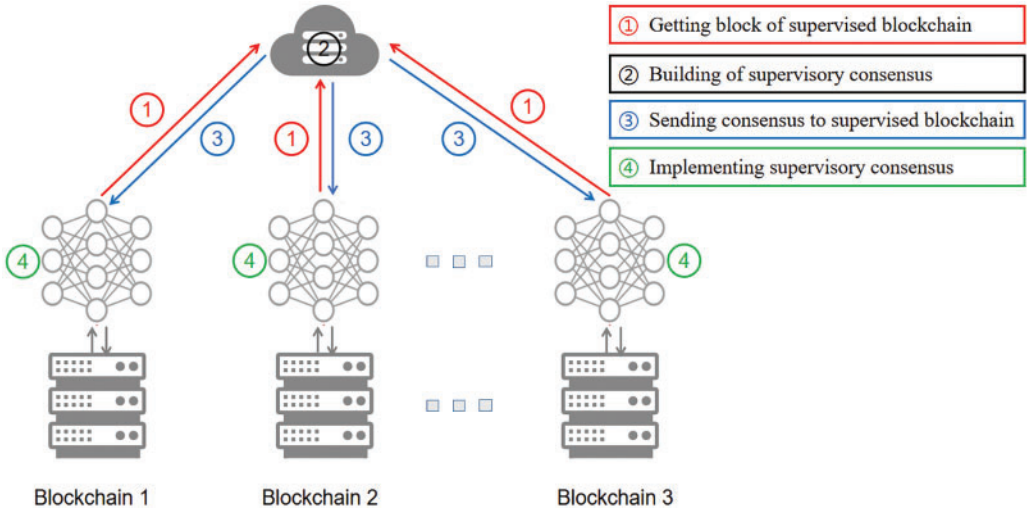

Figure 1: Architecture of governing blockchain by blockchain system

Table 1: A comparison of supervisory and supervised blockchain

Indicators	Supervisory blockchain	Supervised blockchain
Governance	Consortium	Public/Consortium
Node participation	Permissioned	Permissionless
Node access mechanism	Exist	Usually do not exist
Node connectivity	Long-term fixed access.	Free join or leave
Network size	Smaller	Larger
Network connectivity	High	Low
Network synchrony	Partially synchronous/synchronous	Asynchronous/partially synchronous
Transaction capacity (tps)	Higher	Lower
Application examples		Cryptocurrency, smart contract, electronic evidence, DApp

1.3 The Contributions

This paper proposes a new cross-blockchain consensus mechanism based on intelligent contract to meet the requirements of both supervision and governance in chain-governing application scenarios. The new consensus mechanism is known as the Proof-of-Endorse-Contract (PoEC), which supports the supervisory blockchain to transfer the consensus reached to the smart contract in the supervised blockchain, and then the miners in the supervised blockchain will package and verify the new block according to the status of the smart contract. PoEC protocol supports the supervised blockchain to

join governing blockchain by blockchain system without changing the original consensus mechanism, which belongs to a kind of cross-blockchain consensus mechanism with scalability.

1.4 The Structure of Paper

The rest of this article is structured as follows. In Section 2 we briefly introduce the work related to consensus mechanism. Section 3 outlines the design and architecture of Proof-of-Endorse-Contract (PoEC) consensus mechanism. Next in Section 4 we introduce the implementation details of Endorse Smart Contacts. Section 5 presents the designs and settings of experiments, as well as discussion of the experiment results. Finally, we summarize our contributions and discuss future research directions in Section 6.

2 Related Work

Since our paper mainly discusses the subject of consensus agreement, we have made a brief literature review of the mainstream consensus mechanism, analyzed the main ideas and characteristics, and discussed its applicability in the multi-stage, heterogeneous, cross-blockchain governing blockchain by blockchain application scenario, to lay a foundation for the discussion of our PoEC consensus protocol in Chapter 3. Consensus mechanism can usually be divided into classical distributed consensus mechanism and blockchain consensus mechanism, among which the classical distributed consensus mechanism is also known as Byzantine consensus mechanism [12], and blockchain consensus mechanism can usually be divided into Proof-of-X (PoX) consensus mechanism, authorization consensus mechanism and mixed consensus mechanism.

2.1 Classic Distributed Consensus Mechanisms

The classical distributed consensus mechanism is the consensus mechanism used in the traditional distributed network, which realizes the distributed consensus through the state machine replication between network nodes. In [13] proposed the Byzantine General problem and studied how non-fault nodes reach agreement on specific data in the case of possible failure nodes or malicious attacks, which became the basis for the research on consensus mechanism. In [14] proposed a Paxos algorithm to solve the Problem of Byzantine generals. This algorithm can tolerate the collapse of a certain number of nodes in the network, to reach an agreement on a specific value in the distributed system. In [15] proposed the Practical Byzantine Fault Tolerance (PBFT). As a solution to the Byzantine generals' problem, PBFT could achieve the final consensus among honest nodes while the number of enemies was no more than 1/3 of the total number of nodes. In [16] proposed a new common algorithm: Mixed Byzantine Fault Tolerance (MBFT). Functionally, MBFT partitions the nodes participating in the consensus process and improves scalability and efficiency without sacrificing security. MBFT also introduces a random node selection mechanism and a credit mechanism to improve security and fault tolerance. In [17] proposed a dynamic reputation practical Byzantine fault tolerance algorithm. The dynamic reputation practical Byzantine fault tolerant algorithm adopts the consensus election method based on credit. The monitoring node divides the remaining nodes into two types of nodes according to their reputation values: consensus nodes and auxiliary nodes, which participate in different stages of the block generation process respectively, and dynamically update the consensus nodes with low reputation scores.

2.2 PoX Consensus Mechanism

PoX consensus mechanism is usually a blockchain consensus mechanism oriented towards public chain. Its core idea is to determine the probability and expectation of the nodes to obtain the accounting right based on the proportion of certain key resources owned by the nodes, to improve the security of the public chain network. In [18] realized the design of bitcoin system based on the traditional Proof-of-Work (PoW), and the blockchain was proposed for the first time as its underlying technology. In [19] proposed Proof-of-Stake, and introduced the concept of age of currency for the first time. The core idea is that the more coins a node has and the longer it has been holding coins, the more likely it will be chosen as a blocker. In [20] proposed Permacoin based on Proof-of-Capacity (PoC), which requires participants to be able to store part of a large file. In [21] proposed a novel lightweight Proof-of-Block&Trade (PoBT) algorithm for the blockchain of the Internet of Things and its integrated framework, which can verify transactions and blocks with reduced computing time. In [22] proposed a novel consensus mechanism called Proof-of-Negotiation (PoN). PoN introduced a trust mechanism to realize the random selection of honest miners and conducted a round of block creation through a negotiation mechanism.

2.3 Authorization Consensus Mechanism

The main idea of authorization consensus mechanism is to complete the generation and maintenance of blocks through distributed consistency algorithm after nodes have been authenticated. In [23] proposed the basic framework for Hyperledger Fabric. Hyperledger Is a series of open source blockchain projects initiated by the Linux Foundation, which aims to provide an enterprise-class open source distributed ledger framework and source code. Hyperledger Fabric is a community-based project that provides a supporting framework for blockchain applications. In [24] proposed the DFINITY consensus mechanism. DFINITY protocol operates in periods and divides all participating nodes into different groups. A random committee is responsible for transaction processing and consensus operation in each period, and at the end of each period, a random number function is used to determine the group serving as the committee in the next period. The PaLa consensus mechanism proposed by [25] realizes the rapid consensus in the authorization network. PaLa uses the method of parallel pipeline to improve the efficiency of block processing and adopts the sub-committee sliding window reconfiguration to ensure the sustainability of transaction processing during the reconfiguration.

2.4 Hybrid Consensus Mechanism

The main idea of hybrid consensus mechanism is to select some nodes as the consensus committee through PoX consensus mechanism and run Byzantine consensus mechanism inside the Committee to complete the generation of blocks. In [26] first combined the classical distributed consistency algorithm PBFT with blockchain and proposed the PeerCensus consensus algorithm. Bitcoin is used as the underlying chain to select a certain number of nodes and complete the generation of the final block through the Chain Agreement (CA) algorithm after their identity authentication. In [27] proposed the Hybrid Consensus mechanism, which realized state machine replication in an unauthorized environment by using workload proof. Hybrid Consensus for the first time uses formal security model and modular design to model the Hybrid consensus mechanism, and proves that it can meet the safety characteristics such as consistency and activity. In [28] proposed ELASTICO, a fragmentation consensus mechanism, which divides nodes participating in consensus into multiple groups, outputs a block from each group and then obtains the total block. In [29] proposed the RapidChain consensus mechanism, which realized computing sharding, communication sharding and storage sharding. Its

main modules include startup, consensus and reconfiguration. In [30] proposed a Proof-of-QoS (PoQ) based on Quality-of-Service (QoS). In this validation protocol, the whole network is divided into several small regions, each region specifies a node according to its QoS, and then runs deterministic Byzantine fault tolerant consensus among all the specified nodes.

Although the above-mentioned consensus mechanisms on the indices such as security and efficiency have excellent performance, but the consensus mechanism is still facing single-chain or homogeneous blockchain, cannot be directly applied to multilevel heterogeneous and cross-blockchain application scenario of governing blockchain by blockchain. It still needs a kind of safe, efficient and scalable cross-blockchain mechanism for governing blockchain by blockchain framework.

3 The Proof-of-Endorse-Contract Consensus Mechanism

The Proof-of-Endorse-Contract (PoEC) consensus mechanism is a cross-blockchain consensus mechanism designed for the scenario of chain regulation.

3.1 Design Requirements for PoEC

There are two main functional requirements in the governing blockchain by blockchain model: supporting the chain of supervision to obtain the data of the supervised blockchain; Support the supervised blockchain to implement the reward and punishment measures given by the supervised blockchain. Among them, the reward and punishment measures are mainly divided into incentive measures and punishment measures. Incentive measures are mainly public chain nodes that participate in the consensus mechanism. How to set the incentive measures is determined by the incentive mechanism, and the implementation of incentive measures is relatively simple. After the supervised blockchain reaches consensus on the incentive measures, at least m of the N supervision nodes can initiate the transfer transaction based on the multi-signature algorithm. Penalties can be specified in three ways: fines, trade bans and frozen balances (while rollbacks are too limited to be considered). From the perspective of consensus mechanism, the essence of realizing penalty is to force the supervised blockchain to accept certain transactions, while the essence of realizing prohibited transaction and frozen balance is to force the supervised blockchain not to accept certain transactions. At the same time, the consensus mechanism has the non-functional requirements to ensure its own consistency, security, scalablility, activity and finality.

3.2 Architecture of PoEC Consensus Process

PoEC architecture is to insert supervisory nodes into the supervised blockchain, which is composed of supervisory nodes and other types of nodes, to realize the supervision and review of the supervised blockchain, and implement the corresponding reward and punishment measures, as shown in Fig. 2. In the figure, the leftmost block corresponds to supervisory blockchain network, which is generally an open public chain network where any node can issue intelligent contracts or transactions. The right-most block corresponds to a supervisory zone chain network, which is generally a licensed alliance chain network or a private chain network composed of representative nodes, various regulatory agencies, notaries or other stakeholders from the supervised blockchain. The supervisory node is deployed in both the supervisory blockchain network and the supervised blockchain network, and acts as the gateway node to realize cross-blockchain communication between the supervised blockchain and the supervised blockchain. Based on this architecture, the first aspect of functional requirements can be easily realized – supporting the supervisory chain to obtain the data of the supervised blockchain: N supervisory nodes are inserted into the supervised blockchain,

and each supervisory node maintains the data of the whole public chain locally. Corresponding to the real-time monitoring function, when a new block is confirmed by the supervisory blockchain, supervisory nodes will broadcast it to the supervisory blockchain network. Corresponding to the function of active review, nodes in supervisory blockchain can issue data requests to supervisory nodes. Furthermore, due to the balance between scalability and efficiency, the amounts of supervisory nodes can be dynamically changed.

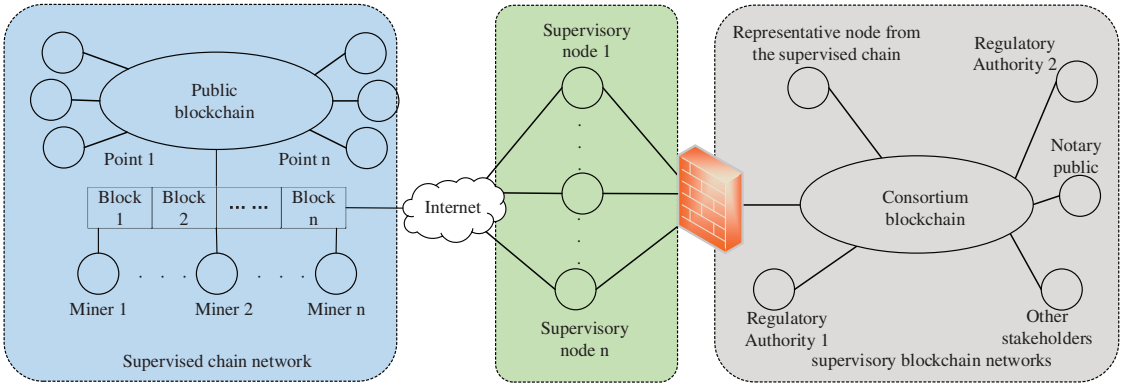

Figure 2: Architecture of PoEC consensus process

3.3 PoEC Consensus Process Description

As noted above, penalties can be specified in three categories: fines, trade bans, and frozen balances. From the perspective of consensus mechanism, the essence of realizing penalty is to force the supervised blockchain to accept certain transactions, while the essence of realizing prohibited transaction and frozen balance is to force the supervised blockchain not to accept certain transactions. The specific process of implementing the three punishment measures is as Fig. 3.

Process 1. Synchronize lastest block. Each supervisory node synchronizes the lastest block of supervised blockchain into supervisory blockchain via the message $(blockheight, block, node_{id})$, where $node_{id}$ is the number of sender node. Both *blockheight* and *block* will be accepted if the amounts of senders have reached trust threshold value $n = \left[\frac{m+1}{2}\right]$, where m is total amounts of supervisory nodes and [ ] is the operator rounding down.

Process 2. Supervisory consensus reached. After the lastest block of supervised blockchain is synchronized, supervisory blockchain reaches supervisory consensus (*S_blockheight*, *S_block*) based on whether Hotstuff, PBFT or other Byzantine algorithms, where *S_blockheight* is the height of supervisory block and *S_block* is the content of supervisory block.

Process 3. Send consensus to contracts. Each supervisory node synchronizes above consensus to supervised blockchain by sending consensus message $(S\_blockheight, S\_block, node_{id})$ to Endorse Contracts in supervised blockchain. Endorse Contracts record the message and the consensus will be accepted once the amounts of senders have reached trust threshold value n.

Process 4. Execute supervisory consensus. miners in supervised blockchain query the height and content of lastest supervisory consensus (*S_blockheight*, *S_block*) before executing and packaging

normal transactions, the supervisory transactions in *S_block* will be executed and packaged preferentially and immediately when *S_blockheight* has been newly added. Algorithm 1 describes PoEC consensus.

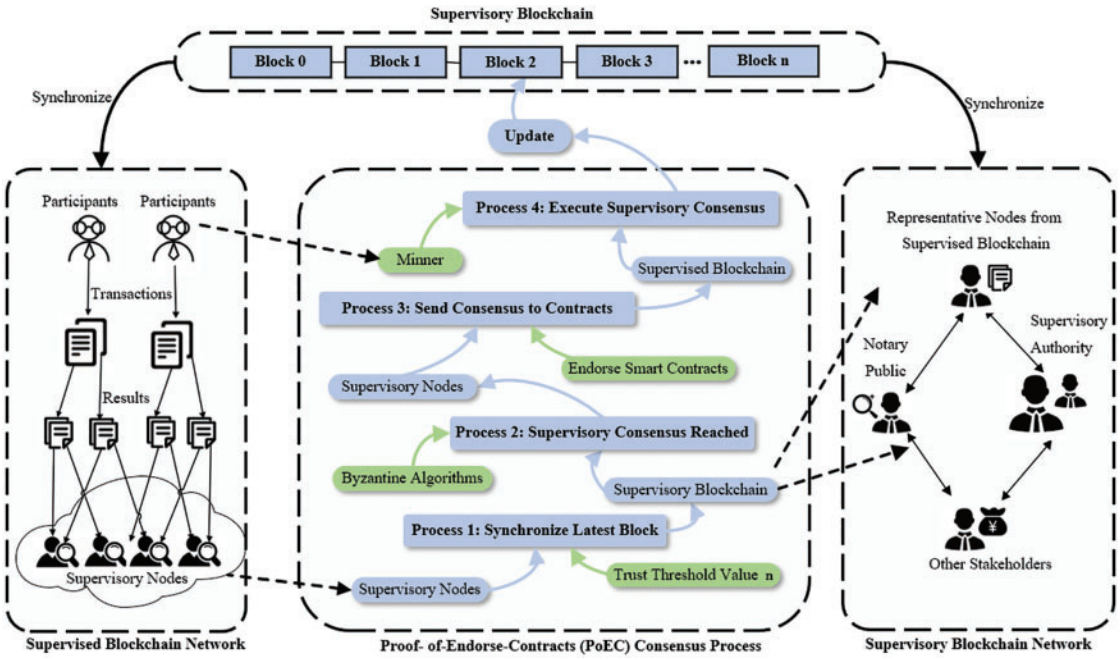

Figure 3: Proof-of-Endorse-Contracts consensus mechanism

Algorithm 1 Proof-of-Endorse-Contracts consensus

InPut: lastest block height of supervised blockchain *blockheight*, lastest block content of supervised blockchain *block*, number of sender node $node_{id}$

for each supervisory node **do** //Process 1
  send $(blockheight, block, node_{id})$ to supervisory blockchain
  $m \leftarrow$ total amouts of supervisory nodes
  $n \leftarrow$ the amounts of sender nodes
  if $2n - 1 < m$ **then**
    supervisory blockchain accept $(blockheight, block)$
    break
  end if
end for
while supervisory consensus not reached **do** //Process 2
  sleep()
end while
$S\_blockheight \leftarrow$ height of supervisory block
$S\_block \leftarrow$ content of supervisory block
for each supervisory node **do** //Process 3
  send $(S\_blockheight, S\_block, node_{id})$ to Endorse Contracts
  $m \leftarrow$ total amouts of supervisory nodes
  $n \leftarrow$ the amounts of sender nodes
  if $2n - 1 < m$ **then**
    supervised blockchain accept $(blockheight, block)$
    break
  end if
end for
if *S_blockheight* has been newly added **then** //Process 4
  minners in supervised blockchain execute *S_block*
end if

The above process is based on the following two assumptions:

Assumption 1. After the K block is confirmed by the public chain and before the K + 1 block is generated, the supervisory node and the supervised blockchain can complete the process of cross-blockchain, consensus and cross-blockchain.

Assumption 2. Based on the preset incentive mechanism, miners participating in this consensus process will receive corresponding remuneration, which may be gas fee or remuneration other than gas fee. This will ensure that the status of the communication contract in the regulatory block can be updated in the supervised blockchain.

There may be a doubt that if enforcing supervisory transactions will lead to hard branch because some miners may don't accept the supervisory transactions enforced without sender's private key. But the problem should be resolved primarily by relevant functional departments of governing blockchain by blockchain system, PoEC consensus mechanism is a hard fork protocol in fact.

4 Implementation Details of Endorse Smart Contacts

Endorse Smart Contacts are written in Solidity language, compiled and tested using Remix IDE, and local simulation is performed using JavaScript VM. Remix is convenient for users to write and execute smart contract code and provides a debugging and testing environment for solidity code. This section discusses implementation details.

There are three smart contracts are deployed on the blockchain, which are the node management contract *PointManage*, the address management contract *AddressManage*, and the block record contract *BlockRecord*. The entity relationship diagram of Endorse Smart Contacts are shown in Fig. 4.

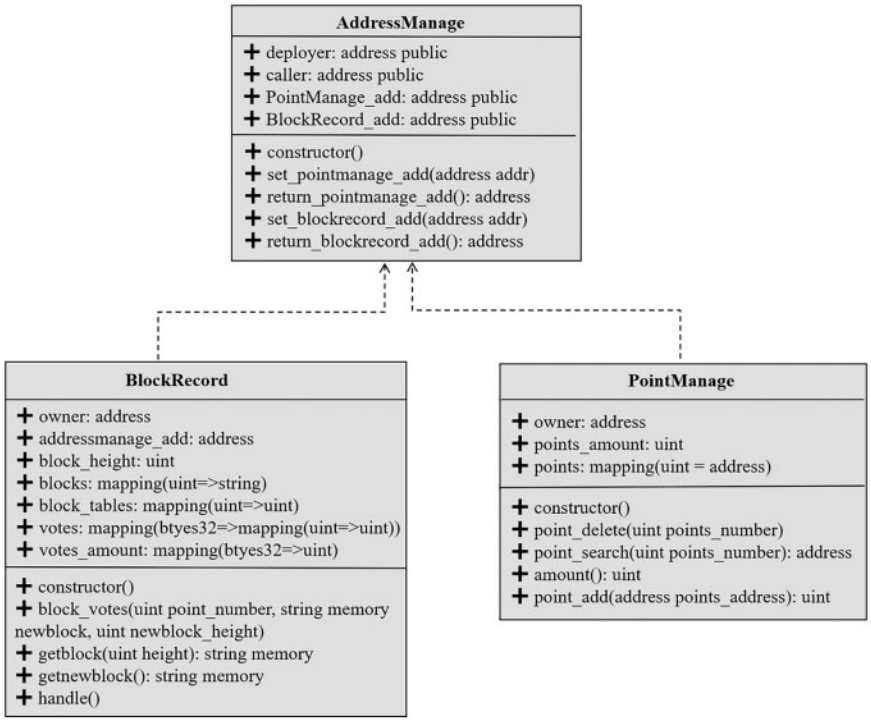

Figure 4: The contract entity relationship diagram of endorse smart contacts

4.1 PointManage Contract Details

The *PointManage* contract is responsible for the management of supervisory nodes list. It includes variables such as the contract owner address *owner*, node address mapping *points*, and the total number of nodes *points_amount*. It provides functions including node addition function *point_add*, node deletion function *point_delete*, node query function *point_search*, and node total number function *amount*. The Implementation details are shown in Tab. 2.

Table 2: Variables and functions of *PointManage* contract

Name	Types	Permission	Describe
points_amount	Integer variable	Public	Total amounts of supervisory nodes
points	Mapping (integer variable to address variable)	Public	Number and address mapping of supervisory nodes
point_add( )	Input: address variable Return: integer variable	Only owner	Add a new supervisory node
point_search( )	Input: integer variable Return: address variable	Public	Search a supervisory node
point_delete( )	Input: integer variable	Only owner	Delete a supervisory node

4.2 AddressManage Contract Details

The *AddressManage* contract is responsible for the management of the contract address. It mainly contains variables such as the node management contract address *PointManage_add* and the block recording contract address *BlockRecord_add*. The functions provided by it include the node management contract address setting function *set_pointmanage_add* and query function *return_pointmanage_add*, block recording contract address setting function *set_blockrecord_add* and query function *return_blockrecord_add*. The implementation details are shown in Tab. 3.

Table 3: Variables and functions of *AddressManage* contract

Name	Types	Permission	Describe
PointManage_add	Address variable	Public	Address of *PointManage*
BlockRecord_add	Address variable	Public	Address of *BlockRecord*
set_pointmanage_add( )	Input: address variable	Only owner	Set *PointManage_add*
return_pointmanage_add( )	Return: address variable	Public	Return *PointManage_add*
set_blockrecord_add( )	Input: address variable	Only owner	Set *BlockRecord_add*
set_blockrecord_add( )	Return: address variable	Public	Return *BlockRecord_add*

Algorithm 2 algorithm of *block_votes*()

Input *point_number*, *newblock*, *newblock_height*

```
function BLOCK_VOTES(point_number, newblock, newblock_height)
    point_address ←PointManage.point_search(point_number) //query address of superviosry node
    if msg.sender == point_address && newblock_height == block_height then
        newblock_hash ← sha256(newblock) // compute block hash
        if votes[newblock_hash][point_number] == 0 then //if vote is valid
            votes[newblock_hash][point_number] ← 1
            votes_amount[newblock_hash]++
            m ←PointManage.points_amount()
            if 2 * votes_amount[newblock_hash] >= m + 1 && block_tables[block_height] == 0 then
  //if amounts of votes reach trust threshold value now and this block has not been executed
                block_tables[block_height] ← 1
                blocks[block_height] ← newblock
                block_height++
            end if
        end if
    end if
end function
```

4.3 BlockRecord Contract Details

The Implementation details of *BlockRecord* contract are shown in Tab. 4 and the algorithm of *block_votes( )* is shown in Algorithm 2.

Table 4: Variables and functions of *BlockRecord* contract

Name	Types	Permission	Describe
addressmanage_add	Address variable	Public	Address of *AddressManage*
block_height	Integer variable	Public	Height of superviosry block
blocks	Mapping (integer variable to string variable)	Public	Content of superviosry block
block_tables	Mapping (integer variable to integer variable)	Public	If block has been execute
votes	Mapping (integer variable to (mapping of integer variable to integer variable))	Public	Votes record
votes_amount	Mapping (integer variable to integer variable)	Public	Amount of votes record
block_votes( )	Input: integer variable, integer variable, string variable	Only superviosry nodes	Block votes function

(Continued)

Table 4: Continued

Name	Types	Permission	Describe
getblock( )	Input: integer variable Return: string variable	Public	Get a accepted supervisory block
getnewblock( )	Return: string variable	Public	Get lastest accepted supervisory block

5 Experiments

5.1 Experiment Design and Settings

In this section, we evaluate whether PoEC consensus mechanism and Endorse Smart Contracts can meet the requirements of low cost, high scalability and being able to cross-blockchain.

For evaluating PoEC consensus mechanism, we simulated the blockchain network on a Windows 10 system with an I5 Intel Core CPU and 8 GB of RAM, which connected to governing blockchain by blockchain system based on the PoEC consensus mechanism proposed in Section 3.

In details of block settings, we set the average time for creating a block $B_{interval}$ to 12.42 s, the average block propagation delay B_{delay} to 6 s and the total simulation time T_{sim} to 2000 s. So the amounts of blocks we simulated in each period are around 160 while the amounts of main blocks are around 120. In details of transactions settings, we got the real data of transactions in Ethereum and fitting the *GasLimit*, *GasUsed* and *GasPrice* of simulated transaction by *GaussianMixture* and *RandomForestRegressor* functions. For evaluating Endorse Smart Contracts, we firstly deployed Endorse Smart Contracts in Ethereum Ropsten (3) network by Remix IDE. Afterwards we respectively test the process of managing supervisory points and recording supervisory block. Finally we recorded the gas used of core functions by MetaMask extension of Google Chrome.

5.2 Experiment Results of PoEC Consensus Mechanism

The transactions in our simulation has two kinds, one of which is normal transaction generated in Ethereum and the other is supervisory transaction propagated from supervisory blockchain. The initial amounts of normal transactions is constantly 2000, and we gradually changed the ratio of supervisory transactions to normal transactions in [0, 0.5] for steps in 0.05.

Defining the transaction delay T_{delay} as the average transaction delay that from being generated to being packaged of whole chain, we ran the simulation in 10 times and recorded an average value of T_{delay}. Trends at T_{delay} could reflect the scalability of PoEC consensus mechanism under the supervisory transactions increasing. Fig. 5 plots the delay of supervisory transactions ST_{delay}, normal transactions NT_{delay} and total transactions TT_{delay}, while Fig. 6 plots the ratio that part of transactions delay to others. We can verify the scalability of PoEC consensus mechanism from two aspects as follows.

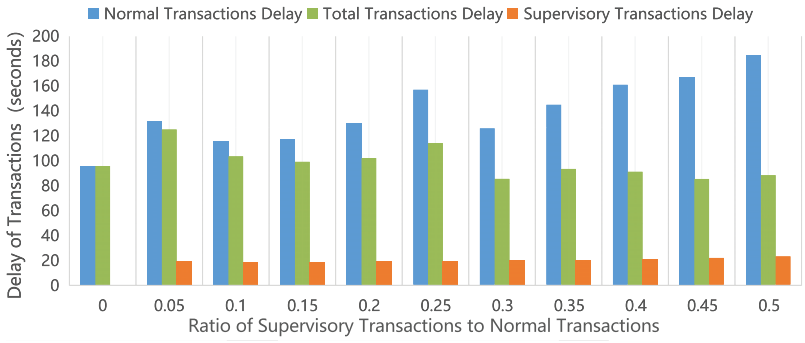

Figure 5: Transactions delay of three kinds in second, the blue one is NT_{delay}, the green one is TT_{delay} and the orange one is ST_{delay}, with the ratio of supervisory transactions to normal transaction changed in [0, 0.5] for steps in 0.05

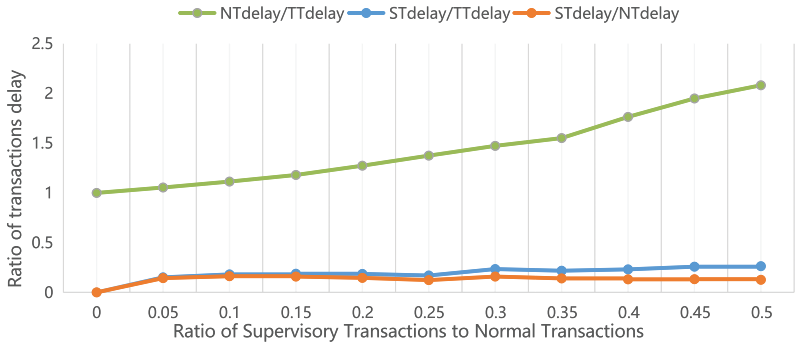

Figure 6: Ratio of transactions delay, while NT_{delay} means normal transaction delay, ST_{delay} means supervisory transactions delay and TT_{delay} means total transactions delay, with the ratio of supervisory transactions to normal transactions changed in [0, 0.5] for steps in 0.05

Firstly, we can find in Fig. 5 that the value of ST_{delay} is always under 20 s while the value of NT_{delay} has a steady tendency from 80 s to 130 s, showing great scalability of supervisory transactions with the ratio of supervisory transactions to normal transactions increasing, because supervisory transactions is preferred packaged in PoEC consensus mechanism **Process 4**.

Secondly, we can find in Fig. 6 that the relative value of ST_{delay} to total blockchain transactions, ST_{delay}/TT_{delay}, sustains smaller than 30 percent with the ratio of supervisory transactions increasing, reflecting the scalability of our consensus as same as Fig. 5. Otherwise, there may be a doubt that if PoEC could bring performance reduction to supervised blockchain. In fact, it depends on the amounts of supervisory transactions, in other words, the requirements of supervision.

5.3 Experiment Results of Endorse Smart Contracts

This part describes the cross-blockchain testing of the key functions of Endorse Smart Contacts, with the corresponding output log attached. For our test scenario, we deployed three smart contracts: *AdressManage*, *BlockRecord* and *PointManage*. Their address are "0xd9145CCE52D386f254917e481eB44e9943F39138", "0xd8b934580fcE35a11B58C6D73aDeE468a2833fa8" and "0xf8e81D47203A594245E36C48e151709F0C19fBe8" respectively, which deployed by a same node, in other words, they

belong a same owner whose address is "0x5b38da6a701c568545dcfcb03fcb875f56beddc4". Most of the features we tested had one or more state requirements that we would not be able to use without meeting the requirements.

Testing 1. Points management. We assume that the supervised blockchain has deployed four supervisory nodes, which address are "0x5B38Da6a701c568545dCfcB03FcB875f56beddC4", "0xAb8483F64d9C6d1EcF9b849Ae677dD3315835cb2","0x4B20993Bc481177ec7E8f571ceCaE8A9e22C02db"and"0x78731D3Ca6b7E34aC0F824c42a7cC18A495cabaB" respectively, while point numbers are 0, 1, 2 and 3 respectively. We use owner account of the contract to increase the fifth point, which address is "0x617f2e2fd72fd9d5503197092ac168c91465e7f2" and point number is 4, while the return value is the point number in contract, such as Fig. 7. Then we inquired the point address with the point number 4, which was the same as the address entered before, as shown in Fig. 8. This test uses the *point_add* function and the *point_search* function in the *PointManage* contract.

```
[vm] from: 0x5B3...eddC4 to: pointmanage.point_add(address) 0xd8b...33fa8 value: 0 wei data: 0xeb7...5e7f2 logs: 0 hash: 0x312...eed04

status                  true Transaction mined and execution succeed
transaction hash        0x312ccb6b23f707d8546ff1bb50380829fea9187cad8048bbdbe202d6fcbeed04
from                    0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
to                      pointmanage.point_add(address) 0xd8b934580fcE35a11B58C6D73aDeE468a2833fa8
gas                     3000000 gas
transaction cost        52288 gas
execution cost          29608 gas
hash                    0x312ccb6b23f707d8546ff1bb50380829fea9187cad8048bbdbe202d6fcbeed04
input                   0xeb7...5e7f2
decoded input           { "address points_address": "0x617F2E2fD72FD9D5503197092aC168c91465E7f2" }
decoded output          { "0": "uint256: 4" }
logs                    []
value                   0 wei
```

Figure 7: Transaction details of adding point by *point_add*

```
[vm] from: 0x5B3...eddC4 to: pointmanage.point_search(uint256) 0xd8b...33fa8 value: 0 wei data: 0x971...00004 logs: 0 hash: 0x19c...f28ab

status                  true Transaction mined and execution succeed
transaction hash        0x19ce5f4c044061b6c070a0d89c2ea3d84633a4ce81e24d4d510b64e58e4f28ab
from                    0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
to                      pointmanage.point_search(uint256) 0xd8b934580fcE35a11B58C6D73aDeE468a2833fa8
gas                     3000000 gas
transaction cost        23541 gas
execution cost          2077 gas
hash                    0x19ce5f4c044061b6c070a0d89c2ea3d84633a4ce81e24d4d510b64e58e4f28ab
input                   0x971...00004
decoded input           { "uint256 points_number": { "type": "BigNumber", "hex": "0x04" } }
decoded output          { "0": "address: 0x617F2E2fD72FD9D5503197092aC168c91465E7f2" }
logs                    []
value                   0 wei
```

Figure 8: Transaction details of searching point by *point_search*

Testing 2. Block record. We continue the testing based on the previous content, and now five nodes have been deployed in the supervised blockchain, i.e., $point\_amount = 5$. At the same time, we assume that three regulatory blocks have been recorded in the contract, with heights of 0, 1 and 2 respectively. For testing purposes, we assign the contents of blocks to the string "This block has a height of 0", "This

block has a height of 1" and "This block has a height of 2" in UTF-8 encoding format respectively. Now we add a block with height of 3, which content is string "This block has a height of 3". This test uses the *block_votes* function and the *getnewblock* function of the *BlockRecord* contract. As a rule of majority, the new block is recorded in the smart contract if and only if the number of nodes sending the new block to the smart contract reaches 3. At the beginning of the test, until the third node votes, we use the *getnewblock* function to query the current most recent block, and the return value is always a block of height 2, as shown in Fig. 9. Then we use node No. 4 to vote for the intelligent contract. This node is the third voting node, and the transaction details are shown in Fig. 10. Then we use the *getNewBlock* function again to query the current most recent block, and it is easy to see that it returns a block with a height of 3, as shown in Fig. 11.

[vm] from: 0x5B3...eddC4 to: blockrecord.getnewblock() 0xd8b...33fa8 value: 0 wei data: 0x735...28db1 logs: 0 hash: 0xc14...e695f

status	true Transaction mined and execution succeed
transaction hash	0xc14fc38ef386d1ecc8113aa90f52e849717ce1658cf9c49877066da7fbee695f
from	0x5B38Da6a701c568545dCfcB03FcB875f56beddC4
to	blockrecord.getnewblock() 0xd8b934580fcE35a11B58C6D73aDeE468a2833fa8
gas	3000000 gas
transaction cost	26194 gas
execution cost	4922 gas
hash	0xc14fc38ef386d1ecc8113aa90f52e849717ce1658cf9c49877066da7fbee695f
input	0x735...28db1
decoded input	{}
decoded output	{ "0": "string: This block has a height of 2" }
logs	[]
value	0 wei

Figure 9: Transaction details of getting newest block by *getnewblock*

[vm] from: 0x617...5E7f2 to: blockrecord.block_votes(uint256,string,uint256) 0xd8b...33fa8 value: 0 wei data: 0xc9e...00000 logs: 0 hash: 0x659...4992e

status	true Transaction mined and execution succeed
transaction hash	0x659f9072e270f427181c557224b07f27b38e38d215ab1aba0e9d3b156694992e
from	0x617F2E2fD72FD9D5503197092aC168c91465E7f2
to	blockrecord.block_votes(uint256,string,uint256) 0xd8b934580fcE35a11B58C6D73aDeE468a2833fa8
gas	3000000 gas
transaction cost	118634 gas
execution cost	94674 gas
hash	0x659f9072e270f427181c557224b07f27b38e38d215ab1aba0e9d3b156694992e
input	0xc9e...00000
decoded input	{ "uint256 point_number": { "type": "BigNumber", "hex": "0x04" }, "string newblock": "This block has a height of 3", "uint256 newblock_height": { "type": "BigNumber", "hex": "0x03" } }
decoded output	{}
logs	[]
value	0 wei

Figure 10: Transaction details of voting block by *block_votes*

Testing 3. Contract Cost. We deployed above contracts in Ropsten (3) network by Remix IDE and MetaMask and run some important functions, with recording the gas used overall process. The gas used and ether cost of each contract and function is shown in Tab. 5, which has an advantage of low cost comparing with other solutions based on smart contract.

Figure 11: Transaction details of getting newest block by *getnewblock*

Table 5: Gas used and Ether cost of each contract and function

Contract/Function name	Operation	Argument	Gas used	Ether cost
PointManage	deploy		382540	0.00038254
AddressManage	deploy		377716	0.000377716
BlockRecord	deploy		716747	0.000716747
points_add	call	*(address)*	52288	0.000052288
points_delete	call	*(uint)*	26190	0.00002619
points_search	call	*(uint)*	23541	0.000023541
set_pointmanage_add	call	*(address)*	46000	0.000046
return_pointmanage_add	call	*( )*	23420	0.00002342
set_blockrecord_add	call	*(address)*	46067	0.000046067
return_blockrecord_add	call	*( )*	23442	0.000023442
block_votes	call	*(uint, string memory, uint)*	118634	0.000118634
getblock	call	*(uint)*	26331	0.000026331
getnewblock	call	*( )*	26194	0.000026194

6 Summary and Future Work

Aiming at the application scenarios of supervision and governance, this paper proposes a cross-blockchain consensus mechanism based on intelligent contract for chain governance. New consensus mechanism called Proof-of-Endorse-Contract (PoEC) consensus, which supports the supervisory blockchain to transfer the consensus reached to the smart contract in the supervised blockchain, and then the miners in the supervised blockchain will package and verify the new block according to the status of the smart contract. We introduce multiple digital signatures into the contract to ensure the security of cross-blockchain communication. PoEC protocol supports the regulated chain

to join the chain regulation system without changing the original consensus mechanism, which has the advantages of low cost.

The experiments prove that our method can provide a feasible cross-blockchain governance scheme for the field of blockchain governance. To the best of our knowledge, our approach is the first to provide practical consensus solutions for governing blockchain by blockchain system and the first consensus mechanism to introduce smart contracts into consensus processes. This provides a basis for further applying machine learning and making the supervisors and supervised blockchain work easily, which are the subject of our future work.

Funding Statement: This work was supported by National Natural Science Foundation of China (Grant No. 62162022 and 62162024), Key Projects in Hainan Province (Grant ZDYF2021GXJS003 and Grant ZDYF2020040), the Major science and technology project of Hainan Province (Grant No. ZDKJ2020012).

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] J. Li, J. Cheng, N. X. Xiong, L. G. Zhan and Y. Zhang, "A distributed privacy preservation approach for big data in public health emergencies using smart contract and SGX," *Computers Materials & Continua*, vol. 65, no. 1, pp. 723–741, 2020.

[2] V. Mavroudis, K. Wüst, A. Dhar, K. Kostiainen and S. Capkun, "Snappy: Fast on-chain payments with practical collaterals," in *Proc. NDSS, San Diego*, CA, USA, 2020.

[3] K. R. Ozyilmaz and A. Yurdakul, "Designing a blockchain-based IoT with Ethereum, Swarm, and LoRa: The software solution to create high availability with minimal security risks," *IEEE Consumer Electronics Magazine*, vol. 8, no. 2, pp. 28–34, 2019.

[4] J. Cheng, J. Li, N. X. Xiong, M. Z. Chen, H. Guo *et al.,* "Lightweight mobile clients privacy protection using trusted execution environments for blockchain," *Computers Materials & Continua*, vol. 65, no. 3, pp. 2247–2262, 2020.

[5] A. Ghosh, S. Gupta, A. Dua and N. Kumar, "Security of cryptocurrencies in blockchain technology: State-of-art, challenges and future prospects," *Journal of Network and Computer Applications*, vol. 163, no. 1, pp. 102635.1–102635.35, 2020.

[6] M. Kouhizadeh, S. Saberi and J. Sarkis, "Blockchain technology and the sustainable supply chain: Theoretically exploring adoption barriers," *International Journal of Production Economics*, vol. 231, no. 1, pp. 1–21, 2021.

[7] S. Saurabh and K. Dey, "Blockchain technology adoption, architecture, and sustainable agri-food supply chains," *Journal of Cleaner Production*, vol. 281, no. 1, pp. 1–13, 2021.

[8] P. J. Taylor, T. Dargahi, A. Dehghantanha, R. M. Parzi and K. K. R. Choo, "A systematic literature review of blockchain cyber security," *Digital Communications and Networks*, vol. 6, no. 2, pp. 147–156, 2020.

[9] C. Chen, "Key technology in alliance chain and challenges in monitoring blockchain," *Scientific Chinese*, vol. 1, no. 22, pp. 35–37, 2019.

[10] X. H. Hong, Y. Wang and F. Y. Liao, "Review on the technology research of blockchain security supervision," *Bulletin of National Natural Science Foundation of China*, vol. 34, no. 1, pp. 18–24, 2020.

[11] Y. Xiao, N. Zhang, W. J. Lou and Y. T. Hou, "A survey of distributed consensus protocols for blockchain networks," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 2, pp. 1432–1465, 2020.

[12] Y. Qu and N. Xiong, "RFH: A resilient, fault-tolerant and high-efficient replication algorithm for distributed cloud storage," in *Proc. ICPP*, Pittsburgh, PA, USA, pp. 520–529, 2012.

[13] L. Lamport, R. E. Shostak and M. C. Pease, "The byzantine generals problem," *ACM Transactions on Programming Languages and Systems (TOPLAS)*, vol. 4, no. 3, pp. 382–401, 1982.
[14] L. Lamport. "The part-time parliament," *ACM Transactions on Computer Systems (TOCS)*, vol. 16, no. 2, pp. 133–169, 1998.
[15] M. Castro and B. Liskov, "Practical byzantine fault tolerance," in *Proc. OSDI*, New Orleans, LA, USA, pp. 173–186, 1999.
[16] M. X. Du, Q. J. Chen and X. F. Ma, "MBFT: A new consensus algorithm for consortium blockchain," *IEEE Access*, vol. 1, no. 8, pp. 87665–87675, 2020.
[17] W. J. Cai, W. Jiang, K. Xie, Y. Zhu, Y. L. Liu *et al.,* "Dynamic reputation-based consensus mechanism: Real-time transactions for energy blockchain," *International Journal of Distributed Sensor Networks*, vol. 16, no. 3, pp. 1–13, 2020.
[18] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008. [Online]. Available: https://bitcoin.org/bitcoin.pdf.
[19] S. King and S. M. Nadal, "PPcoin: Peer-to-peer crypto-currency with proof-of-stake," 2012. [Online]. Available: https://www.semanticscholar.org/paper/PPCoin%3A-Peer-to-Peer-Crypto-Currency-with-KingNadal/0db38d32069f3341d34c35085dc009a85ba13c13.
[20] A. Miller, A. Juels, E. Shi, B. Parno and J. Katz *et al.,* "Permacoin: Repurposing bitcoin work for data preservation," in *Proc. SP*, The Fairmont, San Jose, CA, pp. 475–490, 2014.
[21] S. Biswas, K. Sharif, F. Li, S. Maharjan, S. P. Mohanty *et al.,* "PoBT: A lightweight consensus algorithm for scalable IoT business blockchain," *IEEE Internet of Things Journal*, vol. 7, no. 3, pp. 2343–2355, 2020.
[22] J. Y. Feng, X. Y. Zhao, K. X. Chen, F. Zhao and G. H. Zhang, "Towards random-honest miners selection and multi-blocks creation: Proof-of-negotiation consensus mechanism in blockchain networks," *Future Generation Computer Systems*, vol. 105, no. 1, pp. 248–258, 2020.
[23] C. Cachin, "Architecture of the hyperledger blockchain fabric," 2016. [Online]. Available: https://pdfs.semanticscholar.org/f852/c5f3fe649f8a17ded391df0796677a59927f.pdf.
[24] T. Hanke, M. Movahedi and D. Williams, "DFINITY technology overview series, consensus system,"2018. [Online]. Available: https://arxiv.org/pdf/1805.04548.pdf.
[25] T. H. Chan, R. Pass and E. Shi, "Pa La: A simple partially synchronous blockchain," *IACR Cryptology ePrint Archive*, vol. 1, no. 981, pp. 1–21, 2018.
[26] C. Decker, J. Seidel and R. Wattenhofer, "Bitcoin meets strong consistency," in *Proc. ICDCN*, New York, NY, United States, pp. 13:1–13:10, 2016.
[27] R. Pass and E. Shi, "Hybrid consensus: Efficient consensus in the permissionless model," in *Proc. DISC*, Vienna, Austria, pp. 39:1–39:16, 2017.
[28] L. Luu, V. Narayanan, C. D. Zheng, K. Baweja and S. Gilbert, "A secure sharding protocol for open blockchains," in *Proc. CCS*, New York, NY, United States, pp. 17–30, 2016.
[29] M. Zamani, M. Movahedi and M. Raykova, "RapidChain: Scaling blockchain via full sharding," in *Proc. CCS*, Toronto, ON, Canada, pp. 931–948, 2018.
[30] B. Yu, J. Liu, S. Nepal, J. S. Yu and P. Rimba, "Proof-of-QoS: QoS based blockchain consensus protocol," *Computers & Security*, vol. 87, no. 1, pp. 101580.1–101580.13, 2019.