AMC: A PoS Blockchain Consensus Protocol for Scalable Nodes

Mengnan Wang[1], Yahui Jiang[1], Jianhua Huang[1], and Ruicong Tang[2]

(Corresponding author: Jianhua Huang)

School of information science and engineering, East China University of Science and Technology[1]

130, Meilong Road, Xuhui District, Shanghai 200237, China

Email: jhhuang@ecust.edu.cn

Hong Kong DAEX Blockchain Limited[2]

Hong Kong 999077, China

(Received May 20, 2021; Revised and Accepted June 9, 2022; First Online July 3, 2022)

Abstract

In the public blockchain environment, most of the existing algorithms implement sharding through PoW. The difficulty-based sharding calculation brings energy consumption problems and seriously affects the randomness and performance of sharding. This paper proposed AMC (Agreement based on Multi-party Computation), a scalable PoS blockchain consensus protocol. AMC uses secure multi-party computing (MPC) to achieve sharding instead of consuming computing power. The MPC algorithm π_{random} is used to generate a recognized random number, which ensures the randomness of sharding. π_{random} includes verifying aggregate values, which can reduce the verification complexity to O(1) when all the verified parties are honest. In addition, an improved Tendermint algorithm is used to reach consensus in shards, and the follow-the-satoshi algorithm is used to make the leader election more random and secure. AMC introduced a deposit mechanism to deal with Sybil Attacks and established a forked accountability system to constrain the nodes' malicious behavior, thus protecting the system's security.

Keywords: Blockchain; Consensus Protocol; Secure Multiparty Computation; Sharding

1 Introduction

The blockchain is essentially a distributed database, but what is different from traditional distributed databases is that there are many nodes in the public blockchain and their identities are unknown [1]. In this case, the traditional consensus algorithms cannot be applied to the blockchain. In addition, the throughput, scalability and security of the consensus protocols have also become the bottleneck of the development of the public blockchain. The consensus algorithm of Bitcoin [21] is Proof of Work (PoW) which is simple and has good scalability. As the number of nodes increases, the PoW-based system will become more secure. However, the problems of PoW are low throughput and high energy consumption. From a long-term perspective, with the upgrade of technology, the constant concentration of computing power will inevitably bring security risks. PoW was originally proposed to resist Sybil attacks. Proof of Stake (PoS) [12] can completely replace PoW to achieve this function without generating any energy consumption. In addition, the classic combination of PoS and Practical Byzantine Fault Tolerance (PBFT) has the advantage of high throughput and high security. However, due to PBFT's own requirements for node identities and restrictions on the number of nodes, this combination is not scalable. Fortunately, the application of sharding technology breaks the deadlock. The sharding is to group nodes into different shards which can generate blocks in parallel. The more nodes and the more shards, the greater the throughput. Sharding undoubtedly has significant effects in improving the throughput and scalability of the blockchain. Currently, Ethereum, ELASTICO [18], and Zilliqa chain [27] are all doing sharding attempts. The latter two use PoW for sharding. Although this method is simple and convenient, there are two problems. First, if the difficulty of sharding calculation is low, this will allow nodes to perform multiple calculations in order to enter the shard that is beneficial to them, and may cause malicious nodes to gather in the same shard [23]. Second, if the difficulty of sharding is high, it will not only bring about the problem of energy consumption, but also affect the efficiency of consensus. In addition to the mainstream PoW method, OmniLedger [13] generates an unbiased random number through the RandHound [24] scheme, but requires VRF [9] to select a leader. Although the VRF algorithm can guarantee the randomness of leader selection, it cannot guarantee that the leader's behavior must be correct.

In this paper, we propose AMC (Agreement based on

Multi-party Computation), a scalable consensus protocol which achieves scalability and high throughput through sharding. Unlike the PoW-based sharding method, AMC implements random sharding of nodes based on the recognized random number generated by MPC and the deposit paid by nodes, which completely removes the dependence on computing power, reduces energy consumption and improves the efficiency and security of sharding. The stakeholders in the AMC shards use an improved Tendermint protocol to achieve strong consistency. At the same time, a deposit mechanism is introduced to ensure that each node entering the verification set has to pay the capital cost to deal with Sybil attacks, and a forked accountability system is established according to the margin to constrain the node behavior to protect the system security.

2 Related Work

Consensus protocols are the key to achieve the decentralization and data consistency of blockchains, and have a great impact on the performance, scalability and security of blockchains. Therefore, the research on the consensus protocols has been receiving wide attention. The earliest proposed blockchain consensus protocol is Nakamoto's Bitcoin algorithm, but for security, Bitcoin mining time is an average of 10 minutes, a transaction needs to wait for at least 6 confirmation blocks to be successful in the chain. The throughput is quite low. In 2015, Loi Luu *et al.* proposed a computationally extensible blockchain Byzantine Consensus Protocol (SCP) [17], using the idea of sharding to make the throughput and network computing power approximately linear. The core design idea of SCP is to use PoW to randomly divide the network into shards which generate blocks in parallel. In 2016, Bitcoin-NG [7] divides time into epochs and introduces key blocks for leader election and microblocks that contain the ledger entries. The above-mentioned improved algorithms all improve the throughput to a certain extent, but the defects of the PoW mechanism cannot be avoided. The cost of mining in PoW is very high, but the system's mining rewards are regularly reduced. When miners are unprofitable, they no longer maintain the consistency of the blockchain. This is the typical tragedy of the commons [19]. In this context, nodes with strong computing power are motivated to implement 51% attacks.

In order to avoid the problems of PoW, PPCoin [12] introduced an energy-efficient consensus protocol PoS (Proof of Stake). PoS is based on coin age rather than computing power to provide most of the network security, but malicious nodes would abuse the coin age to implement the double-spending attack. The introduction of PoS alleviated the problem of centralization of computing power, but brought about the problem of centralization of rights. Apart from doing evil actively, nodes with high stake may accept bribes from malicious nodes to create forks. Proof of Activity (PoA) proposed by Iddo Bentov [3] in 2014 greatly increased the cost of bribery attacks. PoA contains a core sub-algorithm called follow-the-satoshi. The sub-algorithm selects N stakeholders based on the empty block generated by the miners' computing power. The probability of a node being selected is proportional to the stake it holds. The former N-1 stakeholders are responsible for signing the empty block, and the last stakeholder is responsible for packaging transactions into the empty block and broadcasting the block. In PoA, attackers need to bribe both the miner and N stakeholders to generate the blocks they want, so the attack cost is very high. However, natural forks are still inevitable in PoA.

In order to avoid the natural fork, Jae Kwon [14] proposed the Tendermint algorithm in which the leader election is based on the deterministic round-robin algorithm and the consensus PBFT. Since Tendermint completely removes the requirement of computing power, the selected stakeholders can vote on multiple blocks at the same height without cost to cause the blockchain to fork, which causesthe "nothing at stake" problem. In order to solve this problem, Tendermint introduced a deposit mechanism. Nodes that abuse voting rights will be penalized for slashing deposit. Later, Vitalik Buterin *et al.* [4] proposed the Casper algorithm which uses the deposit mechanism in Tendermint. The original idea of this algorithm is to be used as the overlay on the existing PoW chain, and it sets checkpoints to resist long-range attacks. Due to the use of voting mechanisms, both Tendermint and Casper have node scalability issues. In 2016, Iddo Bentov *et al.* [2] improved PoA and proposed Chains of Activity (CoA). CoA completely removes the computation part, and uses the information of the first k blocks as the input of follow-the-satoshi to obtain the creators of the next k blocks, it solves the natural fork problem in PoA, but the protocol has a problem of nodes offline. In 2017, Aggelos Kiayias *et al.* [11] proposed Ouroboros which is the first provably secure PoS blockchain consensus protocol. Ouroboros presents a formal PoS protocol implementation model that uses the MPC-based coin-tossing protocol to generate a real random number as the input of follow-the-satoshi to achieve a fair election for a leader. Moreover, the incentive mechanism of Ouroboros enables nodes to behave honestly to reach a near-Nash equilibrium [15], even if the attacking parties collude, the honest parties can still benefit, effectively reducing block detention and selfish mining. However, the protocol generates leaders of multiple consecutive blocks at once, and subsequent leaders are vulnerable to attacks. In 2018, Bernardo David *et al.* [6] improved it and proposed Ouroboros Praos, introducing a verifiable random function (VRF) [20] to implement the secret election of block creators.

3 AMC Consensus Agreement

AMC is designed to scale transaction throughput of the network and provide the network security. The main approach to achieve the goal is sharding, i.e., dividing the mining network into small consensus groups called shards, which can process different transactions in parallel. AMC uses secure multi-party computing (MPC) to implement sharding instead of consuming computing power, thereby improving the efficiency and security of sharding.

3.1 Settings and Assumptions of the System

There are two roles in AMC: users and stakeholders. Users use the blockchain infrastructure to complete the transfers or contract calls, while stakeholders run the AMC protocol to participate in the block consensus and are responsible for the security of the blockchain. To become a stakeholder, a participant is required to pay more than a certain amount of deposit to join the validator set $VaSet$. The set includes each stakeholder's public key address, deposit amount and IP address. The deposit is used as a proof of stake to prevent Sybil attacks and to punish those who misbehave. If a block is successfully generated, the honest stakeholders will be rewarded in proportion to their deposit.

In the aspect of the attack model, the rational model is closest to reality. Rational participants will choose whether to attack the system according to their own benefits. AMC rewards the honest participants, and the malicious participants will be punished to guarantee that the honest behavior of participants can achieve near-Nash equilibrium. The protocol can tolerate attackers with less than 1/3 weight of stake. For the network model, assuming that the network communication channel is partially synchronized, there is an unknown upper bound on the information transmission in the network, and the immediate delay is arbitrary but limited. At the same time, it is assumed that the honest nodes can access the internal clock, which does not need to be exactly the same as the global clock, but floats up and down within a certain range of the global clock.

3.2 AMC Description

AMC divides time into epochs, and each epoch is further divided into multiple time slots, where a slot is a small period of time to generate blocks. A two-tuple (epochId, slotId) containing an epoch ID and a slot ID is used to mark a block.

AMC has a hierarchical blockchain architecture to scale the throughput of the network, as shown in Figure 1.

The network is logically divided into three types of shards: multiple block creation shards, an integration shard and a MPC shard. The block creation shards are the consensus groups which process transactions independently to generate microblocks. The integration shard is responsible for checking and merging the microblocks submitted by the block creation shards, and then broadcasting the integrated blocks to the entire blockchain network. The MPC shard is responsible for generating a random

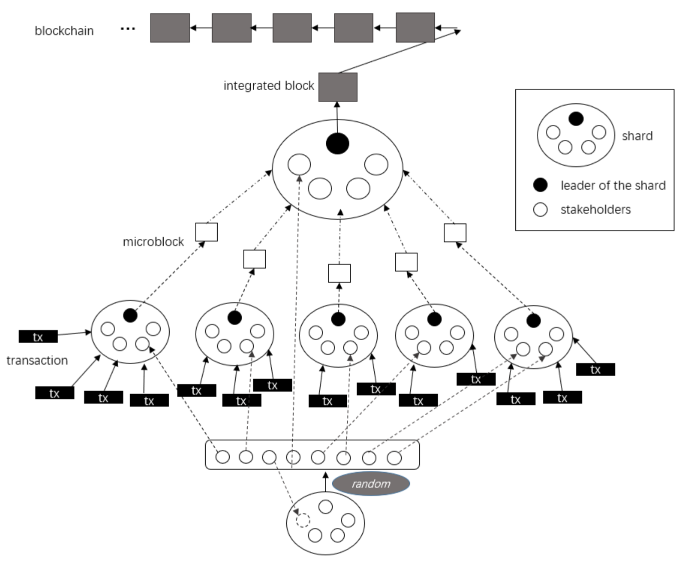

Figure 1: Hierarchical blockchain architecture

number needed for the next epoch sharding. The multiparty coin-tossing protocol uses the random number to assign participants to different shards. Each shard runs an intra-partition consensus protocol to independently process a group of transactions, pack the transactions into microblocks, and send them to the integration shard for generating final blocks.

The whole process is shown in Figure 2. At the beginning of each epoch, a random number generated by the multiparty coin-tossing protocol is used to assign the stakeholders in the network to the shards to which they belong. To ensure protocol initiation, the initial $VaSet$ and $random$ will be hard-coded into the genesis block.

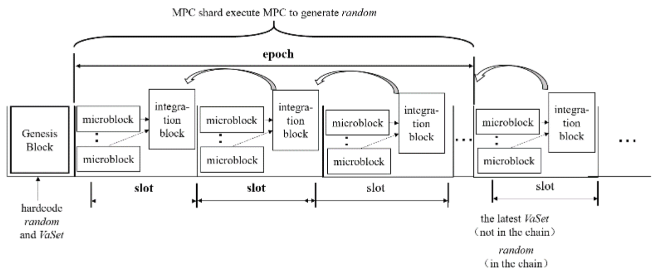
MPC shard execute MPC to generate random
epoch
Genesis Block
microblock
integration block
slot
...
hardcode random and VaSet
the latest VaSet (not in the chain)
random (in the chain)

Figure 2: Operation process of AMC

According to the random number random generated in the previous epoch, each stakeholder $P_i \in VaSet$, calculates $q = LSB_s(hash(random||address_{p_i}))$, where $address_{p_i}$ is the account address of P_i, and $LSB(\cdot)$ represents the rightmost s bits. The value of q is used as the basis for dividing nodes into different shards. When q=0, the node is divided into the MPC shard. When q=1, the node is divided into the integration shard. When q=2, the node is divided into the zeroth creation shard. When

q=3, the node is divided into the first creation block and so on. Anyone can verify the shard to which P_i belongs through *VaSet* and q.

After being assigned to a shard, each stakeholder hashes each address in $VaSet$ with *random* to create a membership stake table for all the shards. At the same time, each stakeholder needs to establish connections with other members in the member table of its own shard to ensure that there is a connected graph in the network.

Then, the shards of three kinds will perform their duties. The execution time of MPC shard is one epoch. block creation shards' work and integration shard's work are completed one after another in the same slot, and are executed repeatedly in slot cycles until new shards are re-established in the next epoch.

3.3 Multiparty Coin-Tossing Protocol with Guaranteed Output Delivery

Sharding randomness is critical to the security of the AMC protocol. The key is to choose a publicly verifiable unbiased random number. This paper proposes a multiparty coin-tossing protocol π_{random} with guaranteed output delivery to generate a recognized random number. π_{random} and the random number generation scheme SCRAPE [5] are both based on Public Verifiable Secret Sharing Scheme (PVSS). The difference is that π_{random} adds verification of the sum of the commitments, which is to verify a set of aggregated values. If the verification is successful, the secret reconstruction can be performed directly. In this case, the complexity of the polynomial verification is $O(1)$. The complexity is $O(N)$ only when the verification fails, while the complexity of polynomial verification in SCRAPE is always $O(N)$.

3.3.1 Protocol Description

π_{random} is based on Heidarvand's PVSS scheme [10], which ensures that no participant can dominate the generation of random numbers, and the protocol guarantees the output when most participants are honest, that is, whether malicious participants abort the protocol prematurely or deviate from the execution of the protocol, the honest participants can get the correct output. Throughout protocol execution, any external node can detect the correctness of the protocol performers' behavior. The description of π_{random} is shown as follows.

Assume that $\Lambda := (q, G, G_1, e)$ represents a bilinear mapping group, and g and h are two independently selected generators of G. The interactors in the protocol are h participants $P_1, \cdots, P_h$, and V represents any external verifier who can access public information. The protocol phases are as follows:

Setting: For all $1 \leq i \leq h$, participant P_i generates a private key sk_i, calculates the public key $pk_i = h^{sk_i}$ and publishes it.

Distribution: For all $1 \leq i \leq h$, participant P_i selects s_i as the distributor, calculates the secret $S_i = h^{s_i}$ and randomly selects $t-1$ parameters $c_1, \cdots, c_{t-1}$, and generates a polynomial $p_i(x) = \sum_{j=0}^{t-1} c_{ij}x^j, c_{i0} = s_i$. For all $1 \leq j \leq h$, P_i calculates secret share $s_{ij} = p_i(j)$, and uses P_j's public key to encrypt s_{ij} to get $\hat{s}_{ij} = {pk_j}^{s_{ij}}$, and calculates the corresponding commitments $v_{ij} = g^{s_{ij}}$ and $v_{i0} = g^{s_i}$. P_i announces $(\hat{s}_{i1}, \cdots, \hat{s}_{ih}, v_{i0}, v_{i1}, \cdots, v_{ih})$.

Disclosure: When all participants have completed the secret distribution, note that the set A contains at least t members who want to reconstruct the secret. For all $1 \leq i \leq h$, the participant $P_j \in A$ publishes the decrypted share $\tilde{s}_{ij} = \hat{s}_{ij}^{{sk_j}^{-1}} = h^{s_{ij}}$ to the blockchain.

Verification: For all $1 \leq j \leq h$, the sum of the shares owned by P_j is $\sum_{i=1}^{h} \tilde{s}_{ij}$, and its corresponding commitment is $v_j = \prod_{i=1}^{h} v_{ij}, v_0 = \prod_{i=1}^{h} v_{i0}$. For all $1 \leq i \leq h$, $1 \leq j \leq h$, V first checks if $e(\hat{s}_{ij}, g) = e(pk_j, v_{ij})$ and $e(pk_j, \tilde{s}_{ij}) = e(\hat{s}_{ij}, h)$ are true, then V randomly selects δ and selects t verified participants to form set B. If set B does not exist, the protocol is terminated; otherwise, V performs VSPS [8] verification on the commitment v_m of the sum of the shares, which means $v_\delta{}' = \prod_{P_m \in B} (v_m)^{\Delta_m}$ is calculated, where $\Delta_m = \sum_{l=0}^{t-1} \lambda_{lm}\delta^l$ and λ_{lm} is a constant that can be calculated directly from B. V verifies whether v'_δ and v_δ are equal. If they are equal, enter the reconstruction phase, otherwise enter the re-verification phase.

Reconstruction: For all $1 \leq i \leq h$, $P_j \in A$ recovers the value h^{s_i} by the formula based on interpolation $h^{s_i} = \prod_{j \in B} (\tilde{s}_{ij})^{\lambda_j}$, $\lambda_j = \prod_{k \neq j} \frac{k}{k-j}$. The resulting random number is $random = \prod_{1 \leq i \leq h} h^{s_i}$, and the protocol is terminated.

Re-verification: For all $1 \leq i \leq h$, V performs VSPS verification on the commitment v_{ij} published by P_i. The method is the same as that in the verification phase. If P_i's commitment does not pass VSPS verification, then P_i is a malicious participant. Note that the set B contains the verified commitments of group t. If B exists, enter the reconstruction phase; otherwise, the protocol is terminated.

In π_{random}, any external node can verify the correctness of the distribution, commitment, and disclosure of the stakeholder's secret shares. If it is found that a stakeholder maliciously sends wrong message, a transaction can be reported and the stakeholder will face the penalty of deposit destruction.

3.3.2 Protocol Security Analysis

π_{random} meets the following two security features:

1) Correctness: If the members in B are honest, all the information published by the members can be verified in the disclosure phase, and in the reconstruction phase, all the members can recover the secret to get the final random number.

2) Verifiability: For the secret s_i shared by participant P_i in the π_{random} verification phase, if the encrypted share $\hat{s}_{ij}$ and the share commitment v_{ij} belonging to the participant P_j pass the verification, there is a high possibility that P_j has correctly announced the information about the share. During the reconstruction phase of π_{random}, if the decrypted value of the encrypted share passes verification, it means that the recovered s_i is indeed a secret shared by P_i.

The key sub-protocol of π_{random} is a threshold secret sharing scheme [25]. If there are at least t honest participants, the secret value can be calculated by interpolation, so it can be intuitively seen that π_{random} meets the correctness. The verifiability of π_{random} is analyzed below.

Theorem 1. *In the π_{random} commitment phase, for all $1 \le j \le h$, if the $(v_{ij}, \hat{s}_{ij})$ published by the secret distributor P_i does not correspond, that is, $\log_g(v_{ij}) = \log_{pk_j}(\hat{s}_{ij})$ is not satisfied, any verifier can verify it.*

Proof. After P_i distributes S_i, any external verifier verifies whether $e(\hat{s}_{ij}, g) = e(pk_{ij}, v_{ij})$ is true. For all $1 \le j \le h$, if $(v_{ij}, \hat{s}_{ij})$ announced by P_i satisfies $\log_g(v_{ij}) = \log_{pk_j}(\hat{s}_{ij})$, and $e(\hat{s}_{ij}, g) = e(pk_j{}^{s_{ij}}, g) = e(pk_j, g)^{s_{ij}}$ and $e(pk_j, v_{ij}) = e(pk_j, g^{s_{ij}}) = e(pk_j, g)^{s_{ij}}$, then $e(\hat{s}_{ij}, g) = e(pk_j, v_{ij})$ holds. If the information published by P_i is $\log_g(v_{ij}) = a \ne \log_{pk_j}(\hat{s}_{ij}) = b$, any verifier can easily calculate $e(\hat{s}_{ij}, g) = e(pk_j, g)^a \ne e(pk_j, v_{ij}) = e(pk_j, g)^b$. □

Theorem 2. *In the π_{random} disclosure phase, if a member P_j in A publishes the wrong decryption share $\tilde{s}_{ij}$, any verifier can detect it.*

Proof. If $\tilde{s}_{ij} = h^a \ne h^{s_{ij}}$, a verifier can easily detect $e(pk_j, \tilde{s}_{ij}) = e(pk_j, h^a) \ne e(pk_j, h^{s_{ij}}) = e(\hat{s}_{ij}, h)$. □

Theorem 3. *If the $(v_{ij}, \hat{s}_{ij})$ published by the secret distributor P_i is not related to the correct share, that is, there is $\log_g(v_{ij}) = \log_{pk_j}(\hat{s}_{ij}) = s'_{ij}$ but s'_{ij} is not the correct share of the secret s_i, then any verifier can detect it.*

Proof. In the disclosure phase of π_{random}, the verifier V selects a qualified B and enters a fast verification phase. For convenience of explanation, it is assumed that the members in B are $P_1, P_2, \cdots, P_t$. For all $1 \le j \le t$, $1 \le i \le h$, $\sum_{i=1}^{h} s_{ij}$ is the point on $\sum_{i=1}^{h} p_i(x)$. Since the commitment s_{ij} is calculated as $v_{ij} = g^{s_{ij}}$ and has additive homomorphism, the commitment of $\sum_{i=1}^{h} s_{ij}$ is $v_j = \prod_{i=1}^{h} v_{ij}$ and commitment value of the secret sum is $v_0 = \prod_{i=1}^{h} v_{i0}$. Note that $\sum_{i=1}^{h} p_i(x) = f(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_{t-1} x^{t-1}$, there is:

$$\begin{pmatrix} 1 & 0 & \cdots & 0 \\ 1 & 1 & \cdots & 1 \\ \vdots & \vdots & \vdots & \vdots \\ 1 & t-1 & \cdots & (t-1)^{t-1} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{t-1} \end{pmatrix} = \begin{pmatrix} f(0) \\ f(1) \\ \vdots \\ f(t-1) \end{pmatrix}$$

Note that:

$$M \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{t-1} \end{pmatrix} = \begin{pmatrix} f(0) \\ f(1) \\ \vdots \\ f(t-1) \end{pmatrix}$$

Then:

$$\begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{t-1} \end{pmatrix} = M^{-1} \begin{pmatrix} f(0) \\ f(1) \\ \vdots \\ f(t-1) \end{pmatrix}$$

Let λ_{ij} be the value of the i-th row and j-th column of M^{-1}, then $a_i = \sum_{j=0}^{t-1} \lambda_{ij} f(j)$.

The verifier V randomly selects $\delta \in [1, t]$, and calculates:

$$\begin{aligned} v'_\delta &= g^{f(\delta)} = g^{\sum_{i=0}^{t-1} a_i \delta^i} = g^{\sum_{i=0}^{t-1} \sum_{j=0}^{t-1} \lambda_{ij} f(j) \delta^i} \\ &= \prod_{j=0}^{t-1} (g^{f(j)})^{\Delta_j} \\ &= \prod_{j=0}^{t-1} (v_j)^{\Delta_j} \end{aligned}$$

Among them $\Delta_j = \sum_{i=0}^{t-1} \lambda_{ij} \delta^i$.

If v'_δ is equal to the known v_δ, the members in B have the correct data. The above process is called VSPS (Verifiable Secret and Polynomial Sharing) property verification, which can verify whether the shares are on the same polynomial. The above is the verification process of aggregated values. Only the complexity of $O(1)$ can verify the correctness of the t member data. However, if the verification fails, the specific error data cannot be determined. At this time, we need to go to the re-verification phase. v_{ij} published by each participant P_i performs VSPS verification. At this time, the participant sending the wrong information can be detected, but the verification complexity is $O(N)$. In the end, any verifier can detect whether $\log_g(v_{ij})$ is the correct share of secret s_i according to VSPS verification.

Moreover, because the existing $e(\hat{s}_{ij}, g) = e(pk_j, v_{ij})$ and $e(pk_j, \tilde{s}_{ij}) = e(\hat{s}_{ij}, h)$ are established, it can be concluded that the published information is correct and the same random number can be obtained. □

3.4 Consensus in Shards

3.4.1 Leader Election

Members in the block creation shards and the integration shard use an improved Tendermint algorithm for consensus. When a leader is selected at the beginning of each slot, Tendermint uses a determined round-robin election algorithm [22], while AMC uses the follow-the-satoshi algorithm. Satoshi is the smallest unit of cryptocurrency. Follow-the-satoshi was first proposed in Proof of Activity. The goal of the algorithm is to make the probability of each stakeholder being selected as the leader proportional to his stake. By contrast, the result of the round-robin election algorithm is that the frequency of a stakeholder being selected as the leader is proportional to his stake. So anyone can know the certain order of leaders before the stake table is updated. In follow-the-satoshi, the leader of each round is uncertain. It can only be said that the greater the stake, the greater the possibility of being selected. In this case, the security is higher, because if the order of the leaders is known in advance, the attacker can make targeted attacks on future leaders.

The AMC implementation of follow-the-satoshi is as follows: The stakeholders arrange the member stake table of the shard in the order of public key addresses, and take the rightmost bits of the hash value of the integrated block generated by the previous slot, and determine the leader of the slot according to the stake interval in which it is located. For example, the stake table arranged in the order of public keys is $(P_1, 4), (P_2, 2), (P_3, 2), (P_4, 2)$, and the divided stake interval is shown in Figure 3. The total stake is 10 satoshi, which need to be represented by 4 bits, then take the rightmost 4 bits of the hash value of the previous integrated block. Assuming the value is 13, calculate 13 mod 10 = 3, which belongs to the P_1 interval, then P_1 is the leader of the slot.

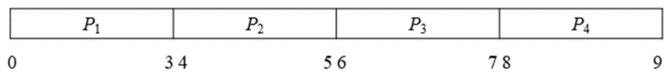

Figure 3: Stake range

3.4.2 Consensus Process in Block Creation Shards

In order to prevent a transaction from being repeatedly processed in different block creation shards, the sender's address is used to allocate the transaction, and the block creation shard to which the transaction belongs will be determined according to the value of $z = LSB_s\left(hash\left(address_{p_i}\right)\right) \bmod count_{block\ shard}$.

The leader in a block creation shard will package transactions to propose a microblock. Other stakeholders in the shard participate in the consensus by voting on the microblock. There are two types of voting specified in vote, namely prevote and precommit. The voting object is identified by the block hash. The *choice* field is 0 for no. This option is valid only during the prevote phase. 1 means yes. *Nil* means timeout and no message is received. Since it is assumed that the operation is in a semi-synchronous network environment and the internal clock of the nodes is not exactly the same as the global clock, there are two possible timeouts: the first is that the sender is a Byzantine node which will delay the message transmission indefinitely; the second is since the upper limit of network delay is unknown and the time limit set by the receiver is too short. AMC stipulates that when more than 2/3 of the voting weight is *nil*, the sender is judged to be a Byzantine node and will be punished. Otherwise, the sender is deemed to have sent the message, and the receiver will appropriately extend its timeout period.

A microblock can be submitted to the upper-level integration shard only if it obtains more than 2/3 of the votes in both the prevote and precommit phases. In the consensus process, 2/3 of the votes do not refer to the proportion of the number of stakeholders, but refer to the proportion of the stake of the stakeholders. The more deposits the stakeholders submit, the greater the voting weight.

The entire consensus process is divided into the following 4 phases:

Proposal phase. When a user sends a transaction to the blockchain network, the stakeholders first verify the validity of the transaction, such as checking whether the transaction belongs to the shard, whether the transaction has been tampered with, whether the signature is correct, whether the balance is sufficient, etc. After the verification is passed, the transaction is put into the transaction buffer pool. The leader packs transactions from his own transaction buffer pool and proposes an initial proposal microblock with the structure shown in Figure 4(a), where $blockhash = hash(epochnumber \,||\, slotnumber||partitionnumber|| \; timestamp \;||\; Transactionfees \;||\; TransactionMerkelroot)$.

Prevote phase. The stakeholder P_i verifies the microblock after receiving it within the prescribed time limit. If the verification is passed, the *choice* field is set to 1, otherwise the *choice* field is set to 0. If the stakeholder does not receive the microblock proposed by the leader after timeout, the *choice* field is set to *nil* and the timeout period is extended. After that, P_i will broadcast (P_i, *vote*) in the shard.

Precommit phase. The operation in this phase is divided into 3 cases according to the voting results in the prevote phase. (a) The stakeholder P_i receives more than 2/3 of the *yes* votes within the time limit, and sets the *choice* field to 1. (b) If P_i receives over 2/3 of the negative votes or *nil* votes within the time limit, the *choice* field is set to *nil*. At this time, it can be determined that the leader is a malicious participant, and P_i can report the formed evidence (microblock, leader's public key, more than 2/3 of negative votes or *nil* votes) to the entire network in the form of a transaction. The reported transaction will generally be processed in the next slot. (c) After timeout, if P_i receives less than 2/3 of the three types of votes, set the *choice* field to *nil* and extend

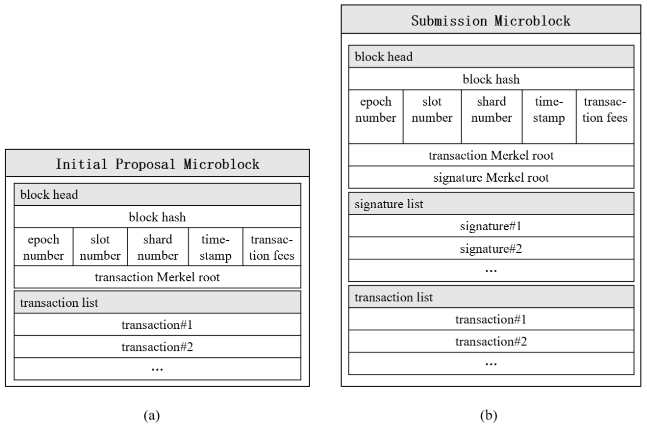

Figure 4: Microblock structure

the timeout period, and then P_i broadcasts (P_i, *vote*) in the shard.

Commit phase. If the leader receives more than 2/3 of the precommit approval votes within the time limit, the signatures of the supporters are added to the microblock, the signature Merkel root is added to the block header, and the hash value of the microblock remains unchanged. Finally, a submitted microblock is formed as shown in Figure 4(b), and the leader submits this microblock to the upper-level integration shard. If the leader receives less than 2/3 of the precommit approval votes after the timeout, *nil* is submitted to the integration shard.

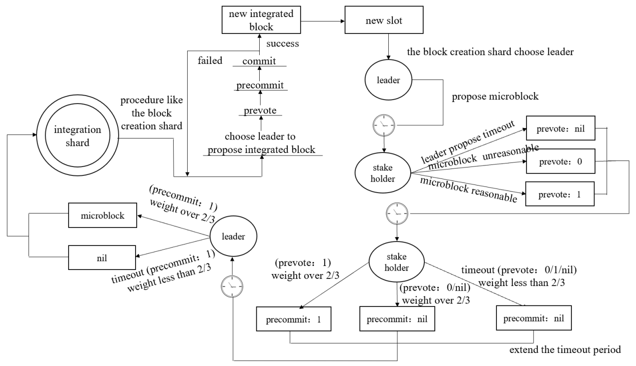
new integrated block
new slot
success
failed
commit
precommit
prevote
choose leader to propose integrated block
procedure like the block creation shard
integration shard
the block creation shard choose leader
leader
propose microblock
leader propose timeout
microblock unreasonable
microblock reasonable
stake holder
prevote: nil
prevote: 0
prevote: 1
(precommit: 1) weight over 2/3
microblock
nil
leader
timeout (precommit: 1) weight less than 2/3
(prevote: 1) weight over 2/3
stake holder
(prevote: 0/nil) weight over 2/3
timeout (prevote: 0/1/nil) weight less than 2/3
precommit: 1
precommit: nil
precommit: nil
extend the timeout period

Figure 5: Consensus process

The entire consensus process is shown in Figure 5. If any malicious behavior in the consensus is found, the stakeholders can report it to the entire network in the form of transactions. For example, the leader proposes multiple microblocks at the same time, the stakeholders vote on multiple microblocks, the same stakeholders vote in contradiction at the same phase, and overtime behavior occurs. If these reported transactions are verified later, the fixed value is taken from the deposit of the malicious node and distributed to the reporting node, and the remaining deposit is destroyed. The purpose of not distributing all the deposit to the reporting node is to prevent nodes from colluding to frame high-stakes nodes.

3.4.3 Consensus in Integration Shard

The consensus process in the integration shard is roughly the same as that in block creation shards. After the leader is selected, the leader verifies the microblocks submitted by the block creation shards. After the verification is passed, the microblocks are merged into the integrated block shown in Figure 6. Similar to Ethereum, AMC is based on the account/balance model. The blockchain using the AMC algorithm can be regarded as a transaction-based "state machine". When the leader processes all transactions from microblocks, it calculates the final states of all accounts. The storage structure of these states uses the Merkle Patricia Trie (MPT) [26] in Ethereum. The leader puts the MPT tree root of the account states into the block header. After the leader proposes the initial integration block, the consensus proceeds to the prevote, precommit, and commit phases in sequence. The final block structure is shown in Figure 7. The leader broadcasts the integrated block to the entire blockchain network, and then the next slot starts.

As shown in Figure 5, unlike the block shards, the integration shard can perform multiple rounds of consensus while the block shards can only perform one round of consensus. This is because multiple block creation shards operate in parallel and limiting the number of consensus rounds to 1 allows each block shard to submit consensus results within approximately the same time. In the integration shard, if the leader does not receive more than 2/3 of the precommit votes in the commit phase, in order to ensure the activity, the leader will be re-elected for consensus until the correct integration block is obtained. The input of the follow-the-satoshi algorithm is the rightmost bits of the previous slot block hash and the round number of this round of consensus.

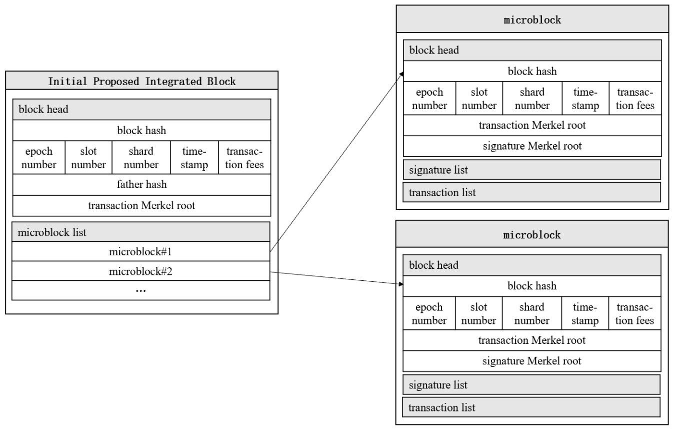

Figure 6: Initial proposed integrated block structure

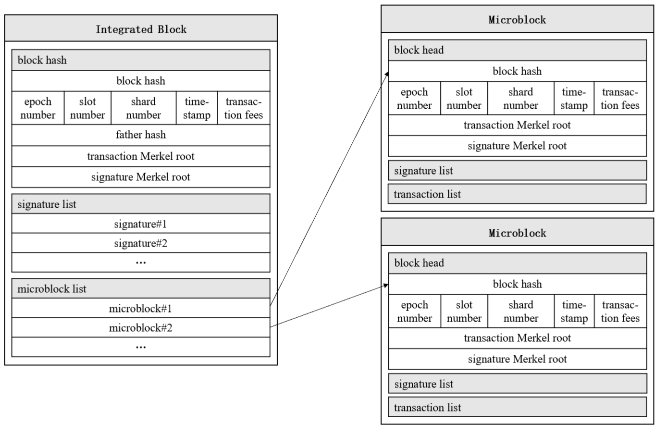

Figure 7: Integrated block structure

3.5 Reduction Strategies and Deposit Redemption Strategies

When a stakeholder needs to go offline temporarily, it should send an offline request transaction, and the stakeholder will be automatically ignored during the next sharding or the next vote. If stakeholders go offline without notifying the system, it may cause stake of the honest parties to be less than 2/3. In order to avoid this situation, such stakeholders will be punished with deposit reduction.

When a node wants to withdraw from the stake set $VaSet$, it needs to send an exit application transaction first. After the transaction is written into the blockchain, the node is no longer in the list of stakeholders, but its deposit will be refunded after a few months. This time is called the "thaw period". The nodes of the entire network need to synchronize the latest blockchain regularly and the time interval is less than the "thaw period", which is to prevent the nodes that exit the stake set from doing evil. Nodes that have been offline for a long time or light clients cannot immediately know the changes in the stake set. If the exiting node forges a block to deceive, with the deposit delay return measure, the node can be punished by destroying the deposit.

4 Security Analysis

4.1 Guaranteed Output of the Multi-party Coin-Tossing Protocol

The idea of generating random numbers by the multi-party coin-tossing protocol is that multiple participants each generate a pseudo-random number, and calculate these pseudo-random numbers through multiparty interaction to obtain a true random number recognized by everyone. Figure 8(a) shows the simplest two-party coin-tossing scheme. The process is as follows:

1) A generates a random string s_A, uses the commitment scheme to commit s_A to obtain $com_{k_A}(s_A)$, and then sends $com_{k_A}(s_A)$ to B.

2) After confirming that A has announced the commitment, B saves it, and then generates a random string s_B and sends it to A.

3) After receiving s_B, A reveals own commitment and sends the revealed value to B.

4) B checks whether it matches the previous commitment according to the plaintext of the random string. If it matches, XOR s_A and s_B to obtain a true random string recognized by both parties.

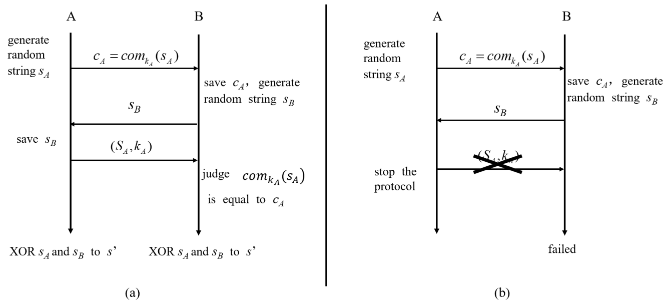
A
B
generate random string s_A
$c_A = com_{k_A}(s_A)$
save c_A, generate random string s_B
s_B
save s_B
(S_A, k_A)
judge $com_{k_A}(s_A)$ is equal to c_A
XOR s_A and s_B to s'
XOR s_A and s_B to s'
(a)
stop the protocol
failed
(b)

Figure 8: Coin-tossing Scheme without guaranteed output delivery

The two-party protocol can be easily extended to multiple parties. It can be seen that A and B need to interact at least 3 times in the above process. During the process, if a participant stops the protocol halfway, as shown in Figure 8(b), if A terminates the protocol after B sends the random string s_B, and does not reveal the commitment, B as an honest party will not get the final output.

In AMC, in order to ensure the smooth sharding, the adopted coin-tossing protocol can guarantee output under the assumption that most nodes are honest. π_{random}'s idea of generating a true random number is roughly the same as other coin-tossing protocols. Each participant generates a random string, publishes the random string after making a commitment to it, and finally multiplies all the random strings to obtain an unbiased random number. π_{random} can guarantee output because it uses a public verifiable threshold secret sharing scheme as its sub-protocol. In π_{random}, the secret to be shared by all distributors is h^s, which uses the Shamir secret sharing scheme to divide the secret parameter s into $(s_1, \cdots, s_h)$, commits these shares and publishes them after encrypting them with the public keys of the corresponding participants. After this step, all the task of the participants as distributors is completed. As long as more than t participants are honest, the fragmented cipher text can be correctly decrypted. The honest participants can obtain enough correct pieces of plaintext, and then recover the secret. Even if a few malicious participants terminate the protocol early, the output of the honest parties will not be affected. To ensure security, t is generally $h/2$.

4.2 Sharding Randomness

Many sharding-based blockchains use the PoW algorithm to implement sharding. The main idea is to calculate a random number *nonce* that meets certain conditions for a participant, and determine which shard the participant belongs to according to the rightmost l bits of *nonce*. Taking classic ELASTICO as an example, the condition for calculating random numbers is $O = H(epochRandom||IP||PK||nonce) \leq D$, where D is the difficulty value, IP and PK are the participants' IP addresses and public keys' address, *epochRandom* is a random number determined by the final committee in the previous epoch. This solution is simple and clear, and realizes anti-Sybil attack and node sharding at the same time.However, if fast sharding is required, the computational difficulty must inevitably be reduced. Nodes with stronger computing power can calculate multiple values, and then choose the most beneficial shard based on these values, which reduces the randomness of sharding. If the computational difficulty is increased in order to ensure security, the meaningless random number calculation will waste a lot of resources and reduce the consensus efficiency to a certain extent.

In response to the above problems, the MPC shard in AMC is designed specifically to generate unbiased random numbers. The adopted coin-tossing protocol can guarantee output when most participants are honest. With a reliable source of random numbers, participants directly use the random numbers and their public key addresses as the input of the hash function, and the result obtained determines the shard to which they belong. Since the computing power requirement is removed, malicious nodes can generate multiple false identities to enter any shard. Therefore, AMC uses PoS to prevent Sybil attacks and stipulates that only participants who have paid enough deposits can join the consensus. In this way, AMC guarantees the randomness of sharding.

4.3 Double-spending Attack Prevention

Double-spending attacks must be achieved through forks [16], but there is no fork in AMC. In a block creation shard or the integration shard, assuming that a malicious leader proposes two blocks A and B to be sent to different stakeholders. The most extreme case is that for the honest parties with 2/3 stake weights, A and B each get a third of the vote, Since the malicious party's stake is less than 1/3, even if the malicious party votes twice, the votes of A and B cannot exceed 2/3 at the same time. Therefore, there will be no forks. In AMC, the leader who proposes multiple blocks or the stakeholders who conduct multiple votes will be reported by other nodes, so any fork behavior will be punished. In some PoW-based blockchains such as Bitcoin, because forks are inevitable, there is no way to identify whether the fork is an attack or a natural occurrence, and it is impossible to hold accountable. However, in AMC, a forked accountability system can be established by node reporting to better protect the security of the blockchain.

4.4 Long-range Attack Prevention

A long-range attack is also known as a long-distance double-spending attack, which means that the attacker starts to create a fork from a block far away in history. This attack is mainly aimed at long-term offline or newly joined nodes. When these nodes want to synchronize the blockchain, there is no guarantee that the blocks they receive are from the main chain or the forked chain created by malicious nodes because there is no trusted source. The PoW mechanism follows the longest chain principle. As long as the forked chain becomes the longest chain, it will become the main chain, which also means that the history of the blockchain is completely rewritten. There is no problem of history rewriting in AMC. Because AMC achieves strong consistency, each block on the chain is finalized. Even if a node is offline for a long time or a newly added node mistakenly enters the fork chain, other nodes will not leave the main chain, and over time, all nodes will return to the main chain.

5 Experimental Evaluation

In this section, we conduct experiments to evaluate the performance of AMC in terms of throughput, delay, and scalability. There are two purposes of the experiments. The first is to test whether the actual performance of AMC is consistent with the theory. The change of throughput and delay with the number of nodes is observed. The second is to compare the performance of AMC with other consensus protocols, including throughput experiment and sharding delay experiment.

5.1 Experimental Environment and Parameter Settings

The experiments use the Docker virtualization technology to build a multi-node environment of the blockchain. Docker's operating platform is installed in a DELL R320 server with Xeon E5 CPU,64G memory, and Ubuntu 14.04 OS. In this paper, 20 nodes are initially selected for each experiment, and the experiment is divided into 10 groups. Each group adds 20 nodes in turn until it reaches 200 nodes. We fix the number of nodes in each shard to 4, so the number of shards also increases from 5 to 50 accordingly. All experimental data are the average of 20 experimental results under the same parameters.

5.2 Analysis Test of AMC Performance

The performance of AMC is mainly evaluated from the aspects of throughput and latency. In order to ensure the consensus efficiency, the block size needs to be limited. In the throughput experiment, we intend to illustrate the trend of throughput with the number of nodes. In order to facilitate the simulation, the size of a microblock is set to 2kB. Since the size of a contract transaction itself is not fixed, and the specific number of transactions in each block does not affect the change trend of the experimental results, it is simply assumed that there are 3 transactions in each microblock.

Figure 9 shows that the throughput of AMC increases linearly with the number of nodes. Each slot in AMC consists of a two-phase consensus. First, the block shards generate microblocks in parallel and submit them to the integration shard in the upper layer. The leader in the integration shard validate all transactions of these microblocks and forms a final block. The more block creation shards in the lower layer, the more transactions the leader has to process in batches. This brings about two problems. One is that the leader takes more time to process transactions; the other is that the integrated block will be larger. After entering the voting process, the communication delay caused by broadcasting the integrated block will greatly affect the consensus speed. As a result, the throughput in AMC cannot ideally increase almost linearly with the number of shards.

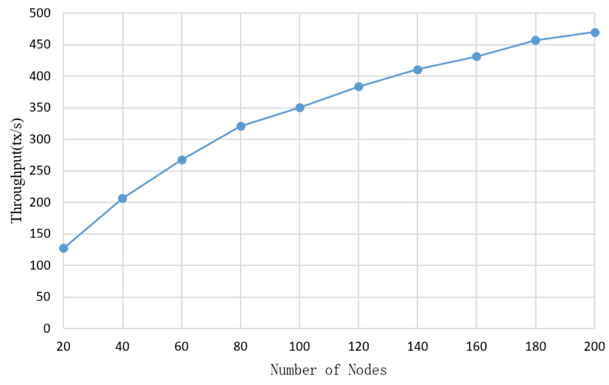

Figure 9: AMC throughput experiment

Delay τ is defined as the time interval from transaction submission to transaction on the blockchain. Since time duration for leaders in block creation shards to receive and package transactions are very short and can be ignored, the consensus delay in the experiment starts after the leaders propose microblocks. When the consensus of all the block creation shards is completed and the microblocks are submitted to the integration shard, the leader of the integration shard will verify the signatures in all the microblocks and merge all the correct microblocks, and then conduct the two-phase voting consensus.

The corresponding relationship between the AMC consensus delay and the number of shards is shown in Figure 10. Since microblocks are generated in parallel in the block creation shards, theoretically the consensus delay at this phase is not much related to the number of shards. However, the experimental results show that with the increase of the number of shards, the consensus delay of the

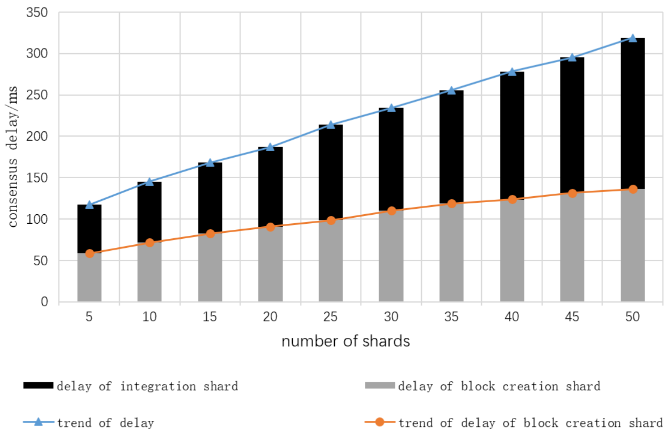

Figure 10: AMC consensus delay experiment

block creation shards increases slightly. This is because the delay calculation has to wait for all the block shards to reach consensus before the first phase is considered to be over. The consensus delay varies slightly from one shard to another. The more shards, the greater the consensus delay. However, the delay increase at this phase is very limited because AMC applies a timeout mechanism. Individual shards with very slow consensus will not affect the consensus progress of the entire system. In the case of the same number of shards, the consensus delay in the integration shard is greater than the consensus delay in the block creation shards. This is because the consensus block in the integration shard is the sum of all microblocks in the block creation shards. When the number of shards increases, the consensus delay of the integration shard also increases, because the leader needs to verify more transactions and signatures, and the block for consensus is also larger.

5.3 Sharding Delay and Throughput Comparison Experiment

Sharding delay refers to the time it takes to randomly partition all participants into different consensus groups. The experiment compares the sharding delay between the proposed sharding method and the PoW-based sharding method adopted by ELASTICO and Zilliqa.

The random number used in each epoch in AMC is generated in the previous epoch. Since the MPC shard runs in parallel with other shards, the process of generating random numbers does not affect the sharding delay. AMC only partitions among stake nodes. At the beginning of a new epoch, after all stake nodes obtain the random number, each stake node first directly hashes its own public key and the random number to determine its own shard and determine other members in the shard, and then the node requests to establish connections with other members in the shard. The sharding process ends when the members in the shard form a connected graph. The sharding delay of the MPC shard is the sum of the hash calculation time and the node connection time. PoW-based sharding can be performed in any node. First, a node needs to calculate a random number that meets the difficulty conditions, and determine its own shard according to the lowest bits of the random number, and then all nodes in the same shard are connected. Compared with the time required to calculate a random number, the time for a node to establish a connection is negligible, so this section considers the PoW sharding delay to be the time of calculating a random number.

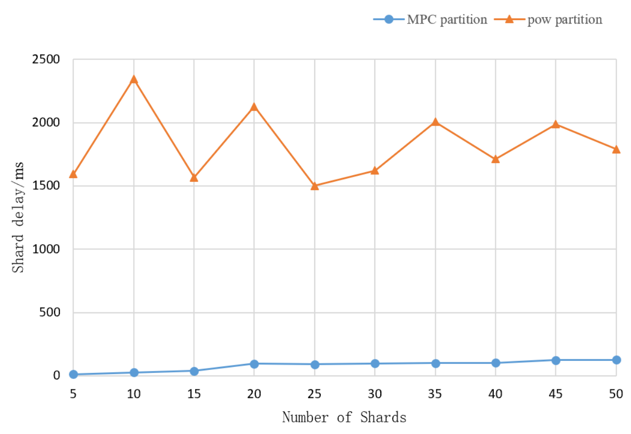

Figure 11: Comparison of sharding delay between MPC and PoW

The PoW and MPC sharding delays under different numbers of shards are shown in Figure 11. With the increase of the number of shards, the sharding delay based on PoW fluctuates around 1800ms, and the sharding delay of MPC is relatively stable. This is because the former is originally a probabilistic calculation. Under the conditions of the same level of difficulty and the same computing power, for different problems, the calculation time of PoW is not fixed. For the latter, the sharding delay is mainly spent on communication. In the case of a relatively stable network, the time for establishing connections between nodes in a shard is relatively fixed, and the shards do not affect each other. The number of shards will not have a large impact on the delay. In the figure, when there are more shards, the delay increases slightly, which is due to the difference between the sharding delays between different shards. We take the longest delay as the sharding delay.

As shown in Figure 11, when the number of shards is the same, the time required for PoW-based sharding is much greater than that in AMC. The sharding delay based on PoW fluctuates within the range of 1.5-2.3s. The value is not very large. That is because the calculation difficulty of the experimental settings is relatively low. In practice, the difficulty of sharding cannot be set too low, otherwise the randomness of sharding will be reduced. The sharding method in AMC can guarantee the randomness of sharding without any additional conditions, because the generation of random numbers is verifiable, and a stakeholder cannot dominate the shard to which it belongs. If a stakeholder wants to enter multiple shards, it must change its identity and repay the deposit. If a malicious node wants to form a scale in a certain shard, it needs to invest a lot of capital costs. Therefore, the sharding method in AMC is more efficient and more secure than the PoW-based sharding.

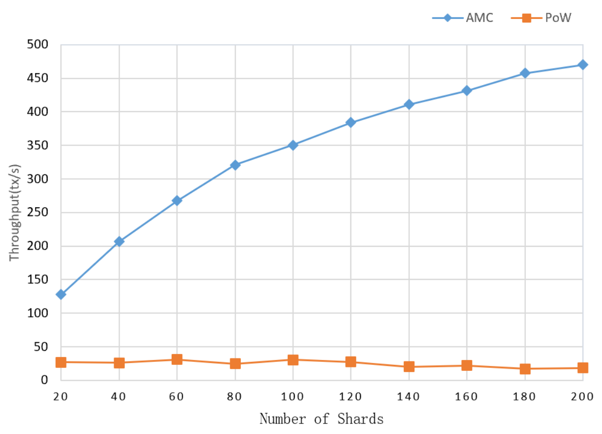

Figure 12: Throughput comparison between AMC and PoW

The throughput experiment compares the throughput of AMC and PoW. PoW is chosen as the comparison object because it is the most classic consensus algorithm in public blockchains. It has good scalability and can accommodate as many nodes as possible. The throughput comparison result is shown in Figure 12. In the case of the same number of nodes, the throughput of AMC is much greater than that of PoW. This is because in order to select block creators and prevent Sybil attacks, PoW must spend a lot of time to calculate random numbers, while AMC uses deposits to prevent Sybil attacks. The limited nodes in shards achieve consensus based on voting, and the consensus efficiency is higher. With the increase of the number of nodes, the number of AMC shards also increases. Since each shard processes transactions in parallel, the more shards there are, the more transactions processed at the same time. Compared with AMC, the throughput of PoW has remained stable. So AMC presents better scalability.

6 Conclusion

This paper proposed the scalable PoS blockchain consensus protocol AMC for resolving the problems of poor sharding randomness and low consensus efficiency in the existing blockchains based on sharding. AMC uses the MPC protocol π_{random} to generate a recognized random number. Based on the random number, the nodes that submit the deposit are randomly assigned into different shards. The entire sharding process completely removes the requirement for computing power, reduces energy consumption and improves security. $\pi_{random}$is a multi-party coin-tossing protocol with guaranteed output delivery. It uses polynomials to share secrets. It includes a fast verification step. When the behavior of nodes is honest, the complexity of the polynomial verification is O (1).

The shards in AMC include the MPC shard that executes π_{random}, the block creation shards that generate microblocks in parallel, and the integration shard that integrates all microblocks. AMC introduces a security deposit mechanism. Once the malicious behavior of members in any shard is detected, they will face the penalty of deducting the deposit. The improved Tendermint algorithm is used in the block creation shards and integration shard to achieve consensus. In addition, the leader election mechanism in the Tendermint consensus algorithm is improved. The follow-the-satoshi algorithm is used to enhance the randomness of the leader election and avoid the danger of the leader being attacked in advance. Experimental results showed that the throughput of AMC increases with the increase of the number of nodes, and the sharding time in AMC is much less than that based on PoW.

Acknowledgments

This research was supported in part by the National Science Foundation of China (grant number 61472139), and National High-tech R&D Program of China (grant number SS2015AA020107).

References

[1] T. Alam, "A survey on the use of blockchain for the internet of things," *International Journal of Electronics and Information Engineering*, vol. 13, no. 3, pp. 119-130, 2021.

[2] I. Bentov, A. Gabizon, A. Mizrahi, "Cryptocurrencies without proof of work," in *International Conference on Financial Cryptography and Data Security*, pp. 142-157, 2016.

[3] I. Bentov, C. Lee, A. Mizrahi, *et al.*, "Proof of activity: Extending bitcoin's proof of work via proof of stake Extended Abstract," *ACM Sigmetrics Performance Evaluation Review*, vol. 42, no. 3, pp. 34-37, 2014.

[4] V. Buterin, V. Griffith, "Casper the friendly finality gadget," Oct. 25, 2017. (`https://arxiv.org/abs/1710.09437`)

[5] I. Cascudo, B. David, "SCRAPE: Scalable randomness attested by public entities," in *International Conference on Applied Cryptography and Network Security*, pp. 537-556, 2017.

[6] B. David, P. Gaži, A. Kiayias, *et al.*, "Ouroboros Praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 66-98, 2018.

[7] I. Eyal, A. E. Gencer, E. G. Sirer, R. van Renesse, "Bitcoin-NG: A scalable blockchain protocol," in *Proceedings of the 13th USENIX Conference on Networked Systems Design and Implementation*, pp. 45-59, 2016.

[8] R. Gennaro, M. O. Rabin, T. Rabin, "Simplified VSS and fast-track multiparty computations with applications to threshold cryptography," in *Proceedings of the 17th Annual ACM Symposium on Principles of Distributed Computing*, pp. 101-111, 1998.

[9] Y. Gilad, R. Hemo, S. Micali, G. Vlachos and N. Zeldovich, "Algorand: Scaling byzantine agreements for cryptocurrencies," *Proceedings of 26th Symposium Operating System*, pp. 51-68, 2017.

[10] S. Heidarvand, J. L. Villar, "Public Verifiability from pairings in secret sharing schemes," in *International Workshop on Selected Areas in Cryptography*, pp. 294-308, 2008.

[11] A. Kiayias, A. Russell, B. David, *et al.*, "Ouroboros: A provably secure proof-of-stake blockchain protocol," in *Annual International Cryptology Conference*, pp. 357-388, 2017.

[12] S. King, S. Nadal, "PPCoin: Peer-to-peer cryptocurrency with proof-of-stake," *Self-published paper*, Aug. 19, 2012. (`https://peercoin.net/whitepapers/peercoin-paper.pdf`)

[13] E. Kokoris-Kogias, P. Jovanovic, L. Gasser, N. Gailly, E. Syta, B. Ford, "OmniLedger: A secure scale-out decentralized ledger via sharding," *Proceedings of IEEE Symposium Security Privacy*, pp. 583-598, May 2018.

[14] J. Kwon, "Tendermint: Consensus without mining,", 2016. (`https://tendermint.com/static/docs/tendermint.pdf`)

[15] J. Li, G. Kendall, R. John, "Computing nash equilibria and evolutionarily stable states of evolutionary games," *IEEE Transactions on Evolutionary Computation*, vol. 20, no. 3, pp. 460-469, 2016.

[16] I. C. Lin, T. C. Liao, "A survey of blockchain security issues and challenges," *International Journal of Network Security*, vol. 19, no. 5, pp. 653-659, 2017.

[17] L. Luu, V. Narayanan, K. Baweja, *et al.*, "SCP: A computationally-scalable Byzantine consensus protocol for blockchains," Cryptology ePrint Archive, 2015. (`http://pdfs.semanticscholar.org/2596/03d8d1c2a6d439eb8fa5038659a94aac08e1.pdf`)

[18] L. Luu, V. Narayanan, C. Zheng, *et al.*, "A secure sharding protocol for open blockchains," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pp. 17-30, 2016.

[19] D. T. Manning, J. E. Taylor, J. E. Wilen, "General equilibrium tragedy of the commons," *Environmental & Resource Economics*, vol. 69, no. 4, pp. 1-27, 2018.

[20] S. Micali, M. Rabin, S. Vadhan, "Verifiable random functions," in *40th Annual Symposium on Foundations of Computer Science*, pp. 120-130, 1999.

[21] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," *Decentralized Business Review*, 21260, 2008.

[22] Z. Nylira, "Proposer selection procedure in tendermint," Aug. 27, 2018. (`https://github.com/tendermint/tendermint/blob/master/docs/spec/reactors/consensus/proposer-selection`)

[23] E. U. Opara, O. J. Dieli, "Enterprise cyber security challenges to medium and large firms: An analysis," *International Journal of Electronics and Information Engineering*, vol. 13, no. 2, pp. 77-85, 2021.

[24] E. Syta, P. Jovanovic, E. K. Kogias, N. Gailly, L. Gasser, I. Khoffi, *et al.*, "Scalable bias-resistant distributed randomness," *Proceedings of IEEE Symposium Security Privacy*, pp. 444-460, May 2017.

[25] C. C. Yang, T. Y. Chang, M. S. Hwang, "A (t, n) multi-secret sharing scheme," *Applied Mathematics & Computation*, vol. 151, no. 2, pp. 483-490, 2004.

[26] C. Yue, Z. Xie, M. Zhang, "Analysis of indexing structures for immutable data," in *2020 ACM SIGMOD International Conference on Management of Data*, pp. 14-19, 2020.

[27] The ZILLIQA Team, *The ZILLIQA Technical Whitepaper*, Aug. 10, 2017. (`https://docs.zilliqa.com/whitepaper.pdf`)

Biography

WANG Mengnan born in 1996. She is a master's student of East China University of Science and Technology. Her research interests include blockchain technology and information security.

JIANG Yahui born in 1994. She received the M.S. degree from East China University of Science and Technology, Shanghai, China, in 2019. Her research interests include blockchain technology and information security.

HUANG Jianhua born in 1963, Ph.D., associate professor. His main research interests include computer networks and information security and blockchain.

TANG Ruicong born in 1990, M.S. His main research interests include blockchain and financial information technology.