Multichain-MWPoW: A $p/2$ Adversary Power Resistant Blockchain Sharding Approach to a Decentralised Autonomous Organisation Architecture

Yibin Xu, Yangyu Huang, Jianhua Shao and George Theodorakopoulos

Cardiff University, United Kingdom

Abstract

Blockchain Sharding (BS) is a blockchain improvement approach. It increases the overall transaction throughput, reduces the resource required, and increases the reward expectation for nodes by splitting the blockchain into several parallel-running committees (shards). Recently, several flexible sharding methods that can tolerate up to $n/2$ Byzantine nodes ($n/2$ security level) have been proposed. However, the nodes in these methods may be frequently reassigned from shard to shard to maintain the security when others leave the system.

Theoretically, nodes in non-sharding blockchains have different weight (power or stake) for creating a consensus, so that the adversary needs to control half of the overall weight of the system to make a piece of faulty information accepted into the blockchain ($p/2$ security level). However, all the nodes in the BS approaches carry the same weight, and it is only under the assumption that the honest participants are creating as many nodes as they can, that the $n/2$ security level BS approach reaches the $p/2$ security level.

In this paper, we present Multichain MWPoW, a $p/2$ security level BS architecture that does not require honest participants to create multiple nodes and allows for fewer node reassignments. It combines the Multiple Winners Proof of Work consensus protocol (MWPoW) and a flexible $n/2$ blockchain sharding approach. Our experiments suggest that Multichain MWPoW largely outperforms existing BSs in terms of security, transaction throughput and flexibility.

Keywords: Blockchain, Distributed ledger, Blockchain Security, Blockchain sharding, Blockchain Performance

1. Introduction

Different kinds of blockchain, e.g., Nakamoto Blockchain [1], Ethereum[2], are proposed in the past ten years of cryptocurrency. They hold promise for more sophisticated usage, such as powering Decentralised-Autonomous-Organisations (DAO) or Decentralised-Autonomous-Companies (DAC), where anonymous participants globally do tasks together without centralised control. Mechanisms are in place to secure the integrity of the work results as well as the incentives to the participants. However, blockchain suffers from both security and performance problems that significantly limits its scope of usage.

The fairness and decentralisation of blockchain-based systems are dependent on how participants reach public consensus, which is usually done by a strength competition known as *mining*. Participants sync and verify a block (a consensus candidate) published by the participant that presented the highest strength in every time window. Participants need to download all the updates from the network to verify blocks, and they approve a block by creating new blocks on top of it. Then, rewards are given to the block creator as incentive. However, this procedure overlooks the heterogeneous nature of devices globally, causing a vicious circle between the reward rate deprivation and the arms race for stronger computation ability as well as broader network bandwidth. This vicious circle may ultimately result in a centralised system when some participants are always winners of the competition, while more and more leave the system. Besides, the blockchain has a low throughput and can also be centralised if the throughput is increased by extending the block size or employing mining pools. Extending the block size may force the disadvantaged devices to leave the system as they are already exhausted from continually downloading and verifying updates that happen all over the network. A mining pool assembles together minimal computation powers and uses them collectively as one to create blocks; however, the mining pool participants do not know how their computation power is used; this brings concern over security.

Various approaches have been explored to solve the security and the performance problems of blockchain. These approaches can be commonly categorised into off-chain approaches [3, 4], lightweight block approaches [5, 6, 7], weighted models [8, 9, 10], directed acyclic graphs [11, 12], and blockchain sharding[13, 14, 15, 16].

1.1. Proof of Work, blockchain and cryptocurrency

Proof of Work (PoW) describes a system that is difficult to create but easy to verify. The most widely used Proof-of-Work scheme – Hashcash[17] – is based on SHA-256 and is later introduced as a part of Bitcoin (Nakamoto blockchain) as the computation strength competition method. There are different kinds of PoW alternatives proposed for blockchains [18, 19, 20].

A block in the blockchain embeds the information of a period; the blockchain periodically attaches new blocks. In the blockchain, *Difficulty* is a measure of how difficult it is to generate a PoW.

$$Difficulty = \frac{difficulty\_target}{current\_target} \tag{1}$$

where $difficulty\_target$ is a constant 256 bit number and $current\_target$ is a 256 bit number. When calculating the difficulty for a hash, one will use the hash as the $current\_target$, and then the difficulty can be derived. The blockchain network has a global block difficulty: valid blocks must have a hash below the current target, and the hash is adjusted by changing the value of Nonce (a field in the block). The global difficulty is adjusted to limit the rate at which the network can generate one new block in an approximately fixed time interval. The blockchain has a pre-set security threshold that the honest people must take more than 50% of calculation power so that the malicious people do not have enough power to create a longer fork branch of blocks when honest people are working on another. New participants can determine the correct records by staying with the longest chain (the mainchain) supposed this chain is longer than the second-longest chain for at least a given length.

When powering a cryptocurrency using the blockchain, the participants only need to check whether the sender of a transaction has spent the funding or not in the blocks of the blockchain before they accept this transaction. As a result, double-spending is prevented: no one should be able to send the same money to more than one receiver at the same time.

1.2. Off-chain approach

"If a tree falls in the forest and no one is around to hear it, does it make a sound?" The quote questions the relevance of unobserved events – if nobody hears the tree fall, whether it made a sound or not is of no consequence [3]. In a blockchain, if only two participants care about an everyday recurring transaction, it is not necessary for all other nodes in the blockchain network to know about that transaction [3]. Off-chain approaches are created under this philosophy. Nodes use Micropayment channels [21, 22] to establish a relationship between two parties to perpetually update balances, deferring what is broadcast to the blockchain in a single transaction, netting out the total balance between the two parties [21]. The off-chain approach empowers nodes to transfer funding privately through micropayment channels. It has mechanisms to secure the interest of the other side of the channel once a node makes a violation of its previous off-chain statements [4]. Off-chain approaches are quick and efficient. However, if we see it from the financial point of view: the network Bidirectional Payment Channels [23] (similar to a BGP system) is needed when multi-parties are involved. Substantially, users with big money become banks; the system is then tending to be financially centralised. The off-chain transactions are only broadcast when one party violated previous transactions. If we see the off-chain approach from the point that transaction broadcasting is seldom needed, users must perpetually monitor the blockchain and to refund their funding when violations made [3]. This design prevents personal devices like phones, desktops from using the off-chain blockchains directly because they might not monitor the blockchain all day long. Also, it is not clear how to use the off-chain approach in non-financial related matters.

1.3. Lightweight block approach

Because the transactions are broadcast to the network, it is relatively safe to assume the nodes have received the majority of the transactions before receiving a block. The transactions inside a lightweight block are replaced using tiny transaction hashes, and the relevant plain-text transactions are only shared when a node fails to decode a lightweight node. Graphene [5] is a blockchain protocol that makes a block contain over 2000 transactions but sized only $2.1 Kbytes$; similar approaches are Xtreme Thinblocks [6] and Compact Blocks [7]. The lightweight block approach significantly extends block throughput. However, disadvantaged nodes might not be able to hear the extensive information due to limited bandwidth or verify the information and calculate PoW on time to catch up with the mainchain. Thus, it may cause a more severe arms race among nodes.

1.4. Weighted models

Some criteria are used to weight nodes in the weighted models, the duty of a node is different by weights. The lightweight node system is an example of the weighted model; a lightweight node does not store any block and is the client of full nodes. Full nodes are referred to as nodes that sync all the transactions and blocks. The lightweight nodes use Simple Payment Verification (SPV) inquires to require relevant previous transactions from the full node to verify a new transaction. A lightweight node only takes up to $4.2 MBytes$ per year, regardless of the total size of blockchain [8], but it cannot verify the next blocks and can be misled by full nodes. Delegated Proof of Stake (DPoS) [9] is a model that people elect a fixed number of representatives and contribute their stakes to these representatives; these representatives then compete in the game of PoS [18]. DPoS has a more massive throughput because the representative nodes usually have a superpower regarding calculation ability, storage, and network bandwidth.

These models are now commonly used in many blockchain-powered IoT systems, where lightweight nodes are at the edge, or the nodes contribute their stakes to DPoS to function the system. These models are using authoritarian/superior nodes, and they are potential-centralised. The system security depends on these representatives; thus, weighted models are not the appropriate approach for DAO and DAC in their original ideas.

1.5. Directed acyclic graph

IOTA[24] is a Directed Acyclic Graph (DAG) implemented in graph blockchain eliminating the power centralisation problem caused by the extensive PoW competition. It does not like ordinary blockchain, having the mining mechanism, nor hold periodical competitions; all actions in IOTA are asynchronous. However, IOTA is more vulnerable to 34% power attacks [12], and it still requires a rather long pending time to use a transaction as the INPUT in the new transactions. Nodes are still required to download a high number of transactions to determine the reliability of the subsequent transactions.

1.6. Blockchain sharding

Blockchain sharding is a blockchain improvement approach that divides the transactions and participants into multiple zones (referred to as shards); it increases the blockchain throughput in the increase of participants and shards. Every transaction has a corresponding shard that deals with further transactions based on that transaction. Through enabling multiple shards to process the transactions in parallel, the throughput is zoomed without lifting the requirement for nodes. Blockchain sharding has also been used to reduce the storage requirement for non-sharding blockchains [25], which helps the blockchain to be implemented in IoT devices that are lacking storage space. Financial models [26] can be built into the blockchain sharding approach to link the digital labour and the market behaviour with the changes in pay and service prices.

However, the first BS consensus protocol Elastico [13] has four weaknesses. (1) After every iteration, all the shards needs to be rebuilt, and the identities of nodes need to be reset. (2) Because it demands much more time to fill up all the shards by solving enough PoWs, the latency grows linearly with the increase of network size. (3) The adversary may calculate PoW in advance so that the adversary can mislead the process of assigning nodes to shards. (4) As a small size of a Shard (around 100 members) is needed to restrict the running Practical Byzantine Fault Tolerance (PBFT) [27] in each Shard, this primarily increased the failure probability. The protocol is insecure in practices because the failure probability can be over 0.97 after six iterations [14]. Meanwhile, even though Elastico enables each participant only to verify a subset of transactions, they must sync every block from every Shard.

RSCoin [15] is a sharding-based protocol designed to scale the centrally-banked cryptocurrencies. It is an approach that transparentise banking systems nowadays by combining a distributed network with a centralised monetary supply; however, this blockchain protocol relies on a trusted source. The system is not Byzantine fault-tolerant because each Shard is executed on a two-phase commit protocol.

OmniLedger [14] solved the problem of Elastico. Nevertheless, it can only tolerate $n/4$ adversary nodes. RapidChain [16] increased the security to $n/3$ Byzantine node resistant. Every Shard in RapidChain uses a fixed size instead of a Binominal (or approximately, Poisson) random variable; the shard size is significant so that the throughput is still not significantly improved.

Xu and Huang [28] proposed a new approach that takes the blockchain sharding into $n/2$ security level. This approach classifies nodes into different classes and maintains the number of nodes of different classes in every Shard to be equal to each other. This approach not only increases the security level, but it also mostly shrank the shard size. However, it can be halted by the adversary with less than $n/2$ of nodes. When the system is halted, the blockchain stopped from generating new blocks, but the record in the blockchain is still correct. There is an extension to this approach[29]; it dynamically alters the number of classes as well as the size of shards to bound the adversary's probability of global halting and makes the system to recover from halting eventually. Because of the dynamic reclassification, it is harder to construct a halting attack as the attacker may not be able to place its nodes into the target classes accurately, and the system will eventually recover from the halting. However, the number of shards in this extension can be reduced drastic, which unstabilised the throughput. For both [28] and [29], every time new nodes are added or the nodes go offline, a global node membership adjustment will occur. These drawbacks increase the frequency and uncertainty in data synchronisation, system stabilisation, and the throughput globally.

In the remaining of this section, we show the security model of the existing blockchain sharding approaches in detail.

1.6.1. Blockchain sharding hypothesis

Taking the philosophy from the off-chain approach: it is not necessarily for everyone to hear every tree falling to maintain the fairness of the system. The fact that a tree falls and the time when a tree falls is correct when it is recognised by most people around the tree assuming these persons have not colluded. Collusion is hard to happen with a sufficient number of people being assigned randomly and completely distributed to subareas of the forest. Moreover, the accumulation of adversary power is prevented by relocating nodes from time to time. As long as the random and distributed assignment is secure and follows the principle of proportionality, taking control of a subarea requires a similar effort as taking control of the whole forest. Figure 1 shows an illustration of this hypothesis.

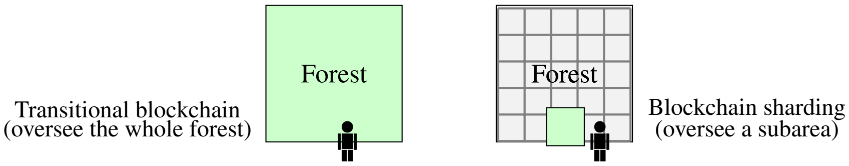


Figure 1: The philosophy of blockchain sharding

In particular, this proposal is secure when (1) only people assigned to a subarea of the forest are legal to record the information about this subarea. (2) any person cannot control or predict which subarea it is about to be assigned in. (3) the assignment follows a globally recognised rule, not by the arbitrary willing of some specific group of superior people. (4) people are periodically reassigned. (5) only qualified people can be assigned, the time to qualify a person is not shorter than the time that one can continuously stay in a subarea. Meaning it is of no benefit to quit a subarea and start over, in case the miner is assigned to a shard it does not want to stay.

When the blockchain security threshold is kept, and more than half of the total population (half of the overall calculation power) are honest people, they do not need to hear every falling by themselves. They would only need to check what is the recognised falling time of a tree to their interest from the subarea where this tree belongs. In this way, people do not need to have super hearing power when the forest is dense. Instead, they only need to focus on monitoring the subarea they are assigned to and split/merge the subareas when the subarea becomes dense or sparse.

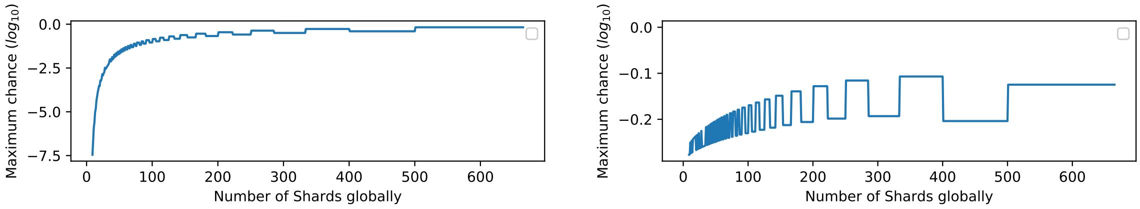
The challenges in blockchain sharding are as follows (1) how to distribute people to subareas in a decentralised and unpredictable way? (2) how can people determine if a record of a subarea is made by people assigned to that area? (3) without monitoring what happened in a subarea, how can outsiders know if the majority in that subarea support a record or not? (4) to make a "collusion" hard to happen, how large the population in a subarea should be, and how many subareas does the forest needs?

1.6.2. Failure probability

Assuming there exist methods to solve all or some of the challenges stated in section 1.6.1. We now calculate how many times of node assignments is required to guarantee a "collusion" to happen. The probability of obtaining no less than $X, (X > m/2)$ adversary nodes when randomly picking a shard sized m (m number of nodes inside the shard) can be calculated by the cumulative hypergeometric distribution function without replacement from a population of n nodes. Let X denote the random variable corresponding to the number of adversary nodes in the sampled shard. The failure probability for one committee is

$$\Pr[X > [m/2]] = \sum_{X=[m/2]}^{m} \frac{\binom{t}{X}\binom{n-t}{m-X}}{\binom{n}{m}} \tag{2}$$

where t is the number of adversary nodes in the system. Figure 2 shows the maximum probability to fail with $n = 2000, t = n/3$, $t = n/2$ and $m = n/s$ where s is the number of shards. As can be seen from the result, the system has a very high failure chance when the adversary taken $n/2$ of nodes. That is the main reason why the most blockchain sharding approaches are only withstanding up to $n/3$ of nodes being bad, and only a few shard can exist.



$t = n/3 = 666$ $t = n/2 = 1000$

Figure 2: The chance to fail when $n = 2000$ and $m = n/s$ where s is the number of shards;

1.7. n/2 security level blockchain sharding approach

In this section, we introduce an approach that can withstand $n/2$ of nodes being adversary in a system of n nodes.

Let there be m classes of nodes, s number of shards. Every shard must have one and only one node of each class, so every shard has m nodes from the m classes. A consensus of a statement is reached when at least a predefined T number of nodes agree on this statement, $T > m/2$. Separate the system into a working zone and a pending zone, where the nodes inside the working zone are placed into shards and can mine (verify the transactions and propose blocks) while the nodes in the pending zone is waiting to be assigned into the working zone. New nodes choose a class when they report to the system and are placed into the pending zone afterward. Nodes of every class queue in the pending zone in the order of the time they chose a class. Let $wq(i, j)$ represent the number j node of class i in the pending zone, and let $lwq(i)$ represent the number of nodes of class i in the pending zone. Every time when $\min(lwq(i)), i \in [1, m] \geq Q$ where Q is a predefined number, then the first Q nodes of every class in the pending zone are added to the working zone, while all the nodes in the working zone are reassigned to new shards.

Let the adversary control A_i nodes in class i in the working zone. Assume, without loss of generality, that the adversary puts all the controlled nodes into classes $i = 1, \ldots, T$ (as the adversary has no more than n/2 of nodes, which is not enough to fill up all the spots in T classes). Then, the probability for the adversary to secure a manipulated consensus inside a shard is

$$Pr[T] = \prod_{i=1}^{T} \frac{A_i}{s} \tag{3}$$

where T is the number of the nodes the adversary must take in a shard to manipulate the consensus. Table 1 shows a node assignment schedule table for ten shards run in parallel with a shard sized five (five people in different classes). In Table 1, A refers to the adversary node, H refers to the honest node. To derive the maximised $Pr[T]$, we want $\prod_{i=1}^{T} A_i$ to be maximised because s is the same. Let the adversary have t number of nodes inside the system, then $t = \sum_{i=1}^{m} A_i$. To let the value of $\prod_{i=1}^{T} A_i$ maximise, we consider

Table 1: Court Jury Schedure

Shard / Class	0	1	2	3	4
Class 1	A	A	A	A	A
Class 2	H	A	H	A	H
Class 3	A	H	A	H	A
Class 4	H	A	H	H	A
Class 5	H	H	H	H	A

$$A_i = \lceil (t/T) \rceil, \quad i \in [1, t \bmod T] \tag{4}$$

$$A_i = \lfloor (t/T) \rfloor, \quad i \in (t \bmod T, T] \tag{5}$$

This scenario is the maximised because given any positive integer Z,

$$Z \times Z > (Z-1) \times (Z+1) = Z \times Z - 1. \tag{6}$$

So that, the maximum chance for the adversary to success on an attack is to balance its nodes into T classes. Thus,

$$Pr[T]_{max} \approx (\frac{t}{T \times s})^T \tag{7}$$

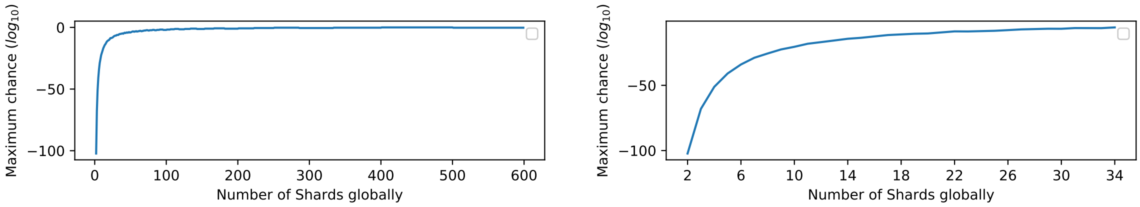
If $T = m$ (all the people in the jury should reach the same decision when making a consensus), then

$$Pr[T = m]_{max} \approx (\frac{t}{s \times m})^m \tag{8}$$

Let $t = \frac{s \times m}{2}$ (half of the overall population), then

$$Pr[T = m]_{max} \approx (\frac{1}{2})^m \tag{9}$$

Though the adversary cannot manipulate a consensus when it does not have T people inside a Shard, it can halt a sentence to be reached when it has $m - T + 1$ number of the nodes in a shard. Then this sentence cannot be made until the next court (the group of juries are re-selected). Thus, to make the system function more smoothly, we want $T \approx [m/2]$ while meeting the security threshold (e.g. 10^{-6} failure chance). Figure 3 shows the maximum failure chance with different s,$n = s \times m = 2000$, $T = 0.7 \times m$ and $t = 1000$ ($1/2$ fraction of the overall population).



$s \in [2, 600], t = 1000 = n/2$

$s \in [2, 34], t = 1000 = n/2$

Figure 3: The chance to fail with different s when $n = 2000$ and $m = n/s$ where s is the number of shards;

As can be seen from the result, when there are ten shards and $n/2$ people being evil, the failure chance is below 10^{-20}, which significantly outperforms the previous BS approach at below 10^{-6} (see Figure 2) when it has ten shards and only $n/3$ nodes being evil. If we maintain the 10^{-6} failure chances at this circumstance with $T = 0.7 \times m$ in $n/2$ approach, there can be 33 shards at the same time.

1.7.1. Global halting problem

The whole system can be halted when there are $s * (m - T + 1)$ adversary nodes, and all of these nodes are in the same $m - T + 1$ classes. In this scenario, it is guaranteed that the adversary has $m - T + 1$ of nodes inside every shard. Table 2 shows an example of a system halting, where $m = 5$ and $T = 4$, the adversary takes $m - T + 1$ number of nodes in every Shard. The halting problem cannot be eased by adding more nodes, as the shard which in charge of the membership issues is also stoped to function.

Table 2: A halted scenerio

Shard Class	0	1	2	3	4
Class 1	A	A	A	A	A
Class 2	A	A	A	A	A
Class 3	H	H	H	H	H
Class 4	H	H	H	H	H
Class 5	H	H	H	H	H

1.8. Flexible n/2 security level blockchain sharding approach

A flexible $n/2$ security level blockchain sharding approach [29] is proposed to solve the problem of global halting. In this approach, every node has a colour from the colour spectrum, and every Shard categories its nodes by grouping them to the closest *base colours*. If there are a m number of categories inside a Shard, then there are a m number of *base colours*, which togetherly represent the colour spectrum as a whole. Figure 4 shows the example for the *base colour*. The system may change the number of categorisations (combine/split the shards) to maintain the security threshold. When a global halting occurred, mechanisms are placed to increase the categorisation number globally (then the number of shard is decreased) to conquer the halting problem. Because the Byzantine does not have more resources than the honest people globally, in the worse case, the halting problem can be solved when the number of shards reduced to one. When the halting problem is solved, the system can then begin to split the shards again.

1.8.1. T adjustment

Assuming the adversary has a $t = \frac{n}{2} - 1$ number of nodes. The chance for the adversary to take control of a shard in a system which has n nodes, m *base colours* and s number of shards ($s = n/m$) is:

$$\Pr[T]_{Max} = \left(\frac{t}{T \times s}\right)^{T} = \left(\frac{t/T}{n/m}\right)^{T} \approx \left(\frac{m}{2 \times T}\right)^{T} \tag{10}$$

when the adversary takes $T(T > 0.5m)$ colour categorisations of nodes inside a shard, it will control this Shard. As can be seen from Figure 5, T can be adjusted with the change of m when maintaining a fixed threshold failure chance. When the m is over 800, T/m is very close to 0.5 (the adversary needs to take approximately $n/2$ of nodes).

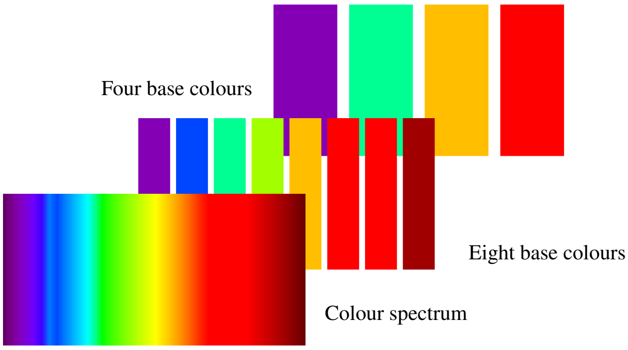


Figure 4: The colours spectrum and *base colours*

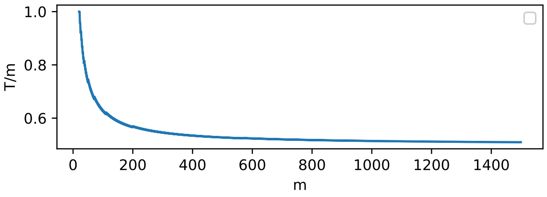


Figure 5: T/m for maintaining a 10^{-6} failure chance with different m

2. $p/2$ security level Blockchain sharding approach

The BS approaches use plurality voting instead of resource/strength competition (mining) to generate the consensus for every shard. That is to say, they trust a statement voted by most people inside a shard, instead of the one proposed by the most strengths/influences inside the shard. In theory, the adversary in classical blockchains need more than 50% of the overall strength to replace the mainchain with a new chain (referred as $p/2$ security level); in BS approaches, the adversary needs to take more than $n/3$ or $n/2$ of nodes to write a faulty information into the blockchain ($n/3$ or $n/2$ security level).

$n/3$ security level is always less secure than $p/2$ approach. $n/2$ security level is generally less secure than $p/2$ security level. $n/2$ and $p/2$ security level are of equal security only when all the participants in the $n/2$ security level BS approaches create as many nodes as they can (as every node only need to present a threshold strength to join in the system, the participants need to fully present all their strengths). However, the cost for maintaining the nodes may excel the benefit from putting nodes into the system, and this cost exists because the participant's nodes can be assigned to different shards, causing multiple workloads in syncing and processing data. An honest participant, especially those using a personal device, is likely to create one or a small number of nodes of the membership threshold power/strength. Nevertheless, in order to gain control of the system, the adversary may create as many nodes of threshold power as possible.

In this section, we propose a $p/2$ blockchain sharding approach that

1. provides less frequent data resynchronisation and membership adjustment, compared to [28, 29].
2. conquers the halting problem of [28] before a complete halting occurred.
3. causes less loss in transaction throughput when recovering from halting than [29].
4. lifts the $n/2$ Byzantine node resistant blockchain into a $p/2$ Adversary (Byzantine) power resistant level.

2.1. Model description and failure chance analysis

Let every node has different strength when voting a consensus. In PoW based systems, the strength represent the calculation power while in PoS based systems, the strength represent the amount of stock. In this paper, we refer the strength as the calculation power.

1. **Node classification**. Line up all the nodes into a list $L = \{L_0 ... L_n\}$ in the sequence of their strengths where n is the number of nodes in the system (exclude the pending nodes). Let CP_x represent the strength of L_x. Let there be a pre-defined Sg number of groups for nodes, every group $i \in [0, Sg)$ has a lower strength boundary $bl(i)$ in the CP list.

   $$bl(i) = CP_{\lfloor \frac{n}{Sg} \times i \rfloor}, i \in [0, Sg) \tag{11}$$

   Every shard must have at least one node from every group.
2. **Block evaluation**. We assume when the adversary proposed a block containing faulty information, the honest nodes would not vote for it. Let AP be the overall strength of the adversary nodes, the chance for a block of shard j being evil (proposed by the adversary) is

   $$\Pr(j) = \prod_{i=0}^{i<Sg} \frac{\binom{AP/tt}{NgS(i,j)}}{\binom{n/Sg}{NgS(i,j)}} \tag{12}$$

   where $Ngs(i, j)$ is the number of nodes in group i which are currently located in shard j and have voted for the block; $DG(i) = 1$ if at least one node from group i in Shard j voted for this block, otherwise $DG(i) = 0$. Let

   $$tt = \sum_{i=0}^{i<Sg} DG(i) \times bl(i) \tag{13}$$

   In order to make the system maintain a $p/2$ security level, we consider

   $$AP = \frac{\sum_{x=0}^{x<n} CP_x}{2} \tag{14}$$

   Formula 12 brings an overestimated result because we assume every node in any group i has the same strength ($bl(i)$). Figure 6 shows an illustration of equation 12.

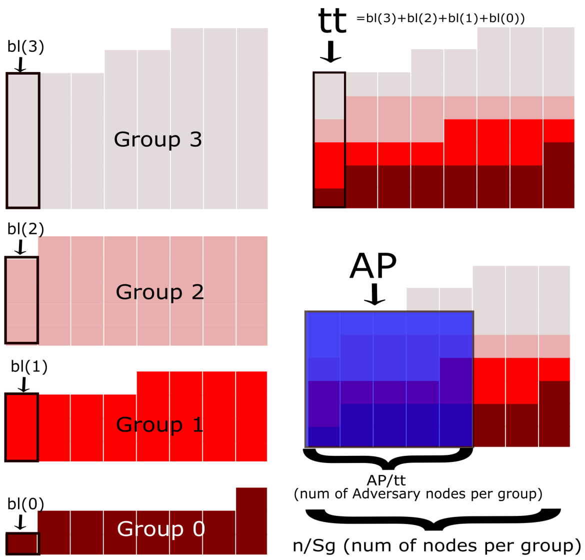

bl(3)
Group 3
bl(2)
Group 2
bl(1)
Group 1
bl(0)
Group 0
tt =bl(3)+bl(2)+bl(1)+bl(0))
AP
AP/tt
(num of Adversary nodes per group)
n/Sg (num of nodes per group)

Figure 6: The explanation of formula 12.Sg=4. Let the height of the bars represent the strengths of the nodes. If a block has supports from every group, then $tt = \sum_{i=0}^{3} bl(i)$. The adversary would create AP/tt number of nodes in every group —- the best strategy for the adversary is to place an equal number of nodes into different groups for the reason given in section 1.7.

We can finally accept this block safely if (1) more than half of the strength in the Shard j has voted for it (majority principle), and (2) the chance for the block to be evil is lower than the security threshold.

Nodes can still mine on blocks that are insecure, but transactions in them would only be finally accepted when the blocks or the branches stemmed from them reached the security threshold. Shard j will be merged with another Shard when:

1. $Max(Pr(j)) > Threshold$, where $Threshold$ is the security threshold (e,g. 10^{-6}). $Max(Pr(j)) = Pr(j)$, when for every $i \in [0, Sg), DG(i) = 1$. It is obvious that the shard j should be combined to others when all the nodes inside the shard j has voted for a block but the chance for this block to be evil (proposed by the adversary) is still larger than the security threshold.
2. When at least five continuous blocks in the mainchain of Shard j have not reached the security threshold. In this case, we say a **local halt** occurred.
3. When there is no node from a group currently located in Shard j.

To make a local halt difficult, we want every possible AP/tt to be smaller. Thus, when adding new nodes, the system should add those whose strength is close to the average strength of nodes in particular groups in priority. They should post penalty or delay adding nodes that would raise the AP/tt. Restrictions should also be placed to avoid an extremely unbalanced power distribution inside the system.

2.2. *Our approach*

In our approach, chains are like shards in the previous $n/2$ approach but can dynamically merge with/separate from each other to fit into the workload. Nodes are ranked according to their calculation power, and they are divided into groups, as introduced in section 2.1. Restrictions are placed to secure every chain having at least one node from every group, and there are mechanisms to restore the restrictions when they are broken.

We rule that,

1. Taking any $2/3$ fraction of nodes out from the system, the sum of their strengths must be equal or larger than $1/2$ fraction of overall strength in the system. Nodes will be kept in the pending status if adding them to the system would compromise this rule.
2. Every shard can assign nodes to other shards, there is no particular chains to deal with the membership issues (like the committee shard in [28, 29]).

3. Nodes only required to sync the blocks of the chain they are assigned into and the block headers of all the chains.
4. The nodes are relocated from chain to chain on a periodical basis (every Ti iterations of the *mining game* after the node participated, Ti is a pre-defined parameter). We do not adjust the nodes globally when adding new nodes.
5. When new miners are joining in the system, we ask them to present Ti times of strength that they intended to use per iteration of the mining game. In this way, it requires the same effort between remaining in a chain for Ti iteration of mining and qualifying a new miner.
6. A miner does not need to be re-quantified if it is reassigned to other chains after Ti runs of the competition.
7. In every round of the mining game, the chains exchange their local group boundaries of their nodes. They do so by recording that information into the block header, which is synced by everyone. By viewing all the block headers, $bl(i \in [0, Sg)$ of formula 11 can be derived.

We use an edited MWPoW [19] (will be discussed in session 3), a decentralised mining pool like blockchain protocol as the protocol that runs on every chain. The design of MWPoW that a miner needs to register calculation power before participate in the mining game can secure the power distribution in a multichain scenario. It also helps to divide the nodes into different groups automatically. By using the edited MWPoW, nodes of other chains can determine if a block of a chain is created by the population of that chain and if the majority supports that block.

Figure 7 and Table 3 shows a comparison among the IOTA, blockchain, Multichain MWPoW and other BS approaches. Full nodes in the IOTA store every transaction ever sent to the network. Full nodes in blockchain store every block in the mainchain. Nodes in other BS approaches keep the transactions in their areas as well as the node membership information from a particular Shard (the committee or the court office).

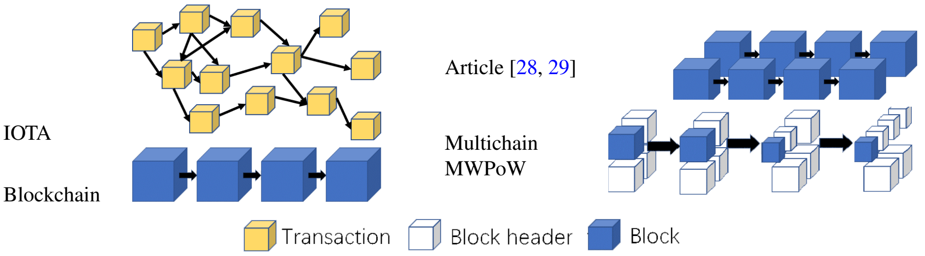


Figure 7: Local storage comparison

In Rapidchain [16], whenever a fixed number of nodes is added to a shard, there is the same number of old nodes being reassigned to other shards. Nodes need to sync data from the new shard once they are reassigned. In [28, 29], when nodes are added or dropped, the assignment needs to be adjusted to meet the categorisation requirement. Also, a local halt is solved by a global membership rearrangement. When global halting, the approach in [29] will also reassign nodes while cutting the colour categorisation. In our approach, the adjustment is more stable. We reassign nodes to other shards every Ti iteration of the mining game. Ti is a predefined global parameter. The Shard would be merged with another to conquer a local halting. There is no need to make a global membership adjustment to conquer a local halting.

To avoid the double-spending problem, in our approach, transactions are governed by different chains: further transactions can only be conducted in the governing chain of the INPUT transaction. If a user wants to transfer a transaction to another chain, they need to conduct a cross-chain operation. Multichain MWPoW does not have levels of committees, and it inherited the fast confirmation property of MWPoW [19]. Multichain MWPoW general outperforms other strength-based blockchains in terms of transactions per second.

We will describe Multichain MWPoW in detail in Section 4, answer the challenges in Section 5. Further analysis of the Multichain MWPoW structure in terms of the bandwidth demand is given in Section 6. An experiment is shown in Section 7. The paper is then concluded in Section 8.

2.3. *Our contribution*

In summary, Multichain MWPoW provides the following novelties:

Table 3: Approach comparison

Name	Transactions stored *	Byzantine fault tolerance level	Storage refreshing time#	Transaction pending time (minutes)$	Maximum num of shardsˆ
Nakamoto	Tx	$p/2$	None	30(3 block confirmations)	1
Article [28]	Tx/s	$n/2$, global halt may occur.	$T_{add} + T_{drop}$	10(Plurality voting) &	33
Article [29]	Tx/s	$n/2$, global halt recoverable.	$T_{add} + T_{drop}+$ restriction restore.	10(Plurality voting) &	33
IOTA	Tx	$p/2$	None	Unstable @	None
RapidChain	Tx/s	$n/3$	T_{add}/Avg_{add}	Plurality voting and unstable %	10
Multichain MWPoW	Tx/s	$p/2$, no global halt.	Every Ti iteration and restriction restore.	Around 12.5 (0.25 block interval)~	33

* Tx is the number of overall transactions, s is the number of shards.

# T_add/drop refers to every time there are nodes added/dropped.
Avg_add refers to the average number of nodes added per time.

$ The block interval is 10 minutes, the transactions must be pending before the block reached acceptance criteria.

@ IOTA requires an accumulation of later transactions pointed to a transaction in order to accept that transaction finally. Thus, the speed to finally accept a transaction is largely dependent on how active the network is.

% As it needs approval from all the Input committees, the time to accept a transaction is unstable.

& Plurality voting systems require nodes to vote within a pre-defined time window (block interval), the transaction is confirmed after one block interval if a consensus is voted.

~ Nodes in Multichain MWPoW need to submit four Shares (PoWs) per iteration. Thus, when a node reached the Acceptance Difficulty, a support rate can be derived similarly to Plurality voting. Sometimes the support rate of a block is not enough to make the block finally accepted, we need to wait until new blocks created on top of it. On average, the block can be accepted in 1/4 block interval after it has been announced (reached Acceptance Difficulty).

ˆ In a 2000 node system and the system is secure from global halting.
For Multichain MWPoW, there are 33% of power is adversary power. For others, the evil nodes are taken 33% of the overall node population.

1. **Increased Byzantine Resiliency**. Multichain MWPoW is the first BS approach that can withstand up to 50% of Adversary (Byzantine) power without assuming the honest nodes holding as many nodes as possible. There is less chance for a global halt attack to occur, as a global halting cannot be deliberately planned as like the scenario we discussed in section 1.7. If a global halt does happen in an accident, it will be resolved as like how we deal with a local halt.
2. **More Flexibility**. A chain (shard) is split and merged base on the data flow of it. Every chain can carry a different number of participants in Multichain MWPoW. However, the number of shards in RapidChain [16] and [28] are fixed. In [29], the number can be changed; however, nodes are still equally divided into shards. When they lose a node in a shard, they need to cancel that Shard, pushing nodes back to the pending section and reorganise one Shard from the pending section.
3. **Increased Transaction per Second**. Less time is spent on halting, and an attack is hard to be conducted. Multichain MWPoW allows less number of nodes per chain, and the chains are stabler. More shards can process transactions in parallel for the same security threshold compared to others.
4. **Faster Transaction Confirmation**. There is no level of election network in Multichain MWPoW. A transaction is confirmed when the governing chain has confirmed it. There is only one governing chain per transaction.

3. Multiple Winners Proof of Work protocol

Multiple Winners Proof of Work (MWPoW) protocol [19] is proposed to shorten the transaction pending time, improve the reward rate for individual miners, and ease the centralisation problem of blockchain.

3.1. MWPoW outline

3.1.1. Definition

- **Calculation Power Claim:** A miner's calculation power is defined as the hash difficulty one can achieve in a fixed time window. Calculation Power Claim is the hash difficulty that a miner intended to reach in every episode of the mining game.

$$CP = CP_0 + CP_1 + ... + CP_{N-1} \tag{15}$$

  where CP is the overall power claimed by registered participants, N is the number of registered participants in the network, CP_{N-1} is the Calculation Power Claim of registered participant NP.

- **New Join:** New Join is a data set, which records the Calculation Power Claim of a participant and a wallet address of this participant (the wallet address is used for receiving remuneration). There is a Nonce field in New Join, which is used for adjusting the hash of New Join. For a New Join to be valid, the hash of this New Join must meet at least the Calculation Power Claim indicated in this New Join.

- **Try Range:** Try range TR is a number interval of the Nonce in the block header.

$$TR_i = \left[\sum_{k=0}^{i-1} Tt_k, \sum_{k=0}^{i} Tt_k\right), Tt_{i\in N} = \frac{CP_i}{CP} * 2^{256} \tag{16}$$

  N is the number of registered participants in the network. Miner $i \in N$ mines on TR_i.

- **Acceptance Difficulty:**The first block which reached the Acceptance Difficulty in an episode of mining should be placed in the mainchain. Acceptance Difficulty is adjusted base on how much time consumed for the winner block to achieve the Acceptance Difficulty.

$$AD_x = \frac{BI * AD_{x-1}}{Timestamp_{x-1} - Timestamp_{x-2}} \tag{17}$$

  where AD$_x$ is the Acceptance Difficulty at block height X; BI is the predefined block interval, and $Timestamp_x$ is the time when block X is created.

- **Entrance Difficulty:** A block is broadcasted to the network when this block reached Entrance Difficulty. Entrance Difficulty of a new episode is adjusted base on how many blocks reached Entrance Difficulty in the previous round of the game.

$$ED_x = min(\frac{NE_{X-1}}{DN} * ED_{X-1}, \frac{AD_x}{2}) \tag{18}$$

  where ED$_x$ is the Entrance Difficulty at block height X; NE$_{x-1}$ is the number of blocks reached Entrance difficulty at block height $X - 1$; DN is the ideal number of NE, we set $DN = 1$.

- **Share:** Share is a container of Nonce when broadcasting. The Nonce inside a Share, which sent by a miner, must make the hash of the block fulfill at least 25% of this miner's Calculation Power Claim.

- **Countable Share:**If a miner has sent at least two Shares for a block, the difficulties of these Shares will be count towards the Support Rate of this block, and the miner will be able to receive remuneration for announcing this block if this block wins the game later.

- **Share Difficulty Cap:**The maximum sum of difficulties of Countable Shares sent by a miner X in a round of the game is CP_X (its calculation power claim). If it sent more, the sum is capped at CP_X.

- **Reward:**

$$R_{i \in N^R} = \frac{SD_i}{SD_{\{X\}}} * R_{\{X\}} \tag{19}$$

N^R is the miners who contributed Countable Shares for announcing block X; $R_{\{X\}}$ is the overall reward assigned from the system for the block in block height X; Shares of block X are embedded in block $X + 1$; $SD_{\{X\}}$ is the total difficulty of the Countable Shares embedded in block $X + 1$; SD_i is the difficulty of the Countable Shares miner i contributed. $R_{i \in N^R}$ is the amount of remuneration given to miner i as a Coinbase transaction in block $X + 1$.

- **Valid Block:**A miner determines a block as a valid one when the transactions, New Joins, and Shares in this block are correct; the Shares and New Joins must be more than 90% previously known to the miner.

- **Support Rate:** The Support rate of a block is defined as the ratio between the sum of the difficulties of the Countable Shares for the branches stem from this block and the sum of difficulties of all Countable Shares of all the branches in the blockchain since the block height of this block.

$$\mathrm{SR}_X = \frac{\sum_{i=X}^{XL} SD_{\{i\}}}{\sum_{i=0}^{k} \sum_{j=i}^{iL} SD_{\{j\}}} \tag{20}$$

SR_X is the support rate of block X; XL is the latest block on top of the blockchain branch stem from block X; k is the number of all the branches; iL refers to the latest block on top of the specific branch; $SD_{\{X\}}$ is the total difficulty of the Countable Shares for block X.

3.1.2. Game overview

Miners need to claim the calculation power they intended to put into the mining game before participating in the game. Each miner is given a unique TR based on the calculation power it claimed. When a miner created a block and found a Nonce that fulfilled Entrance difficulty in its TR, it will broadcast the block as well as the Share. Then other miners will attempt to find a Nonce in their TR to make this block fulfill the Acceptance difficulty if they acknowledge this block as a valid one. Ideally, miners should announce a block collectively by doing PoW in their TR in parallel. When a Share of a block is broadcasted, meanwhile, the Nonce in it made the block reached the Acceptance Difficulty, this block is announced. The first block reached Acceptance Difficulty is the winner block, miners who contributed Shares to this block will divide the remuneration of mining. During the announcement, miners should send Shares that do not fulfill the acceptance difficulty but fulfilled at least 25% of the power they claimed previously as the prove of contribution. A miner can only send up to four Shares to the network per round of the game. If more than one block is successfully announced in one round of the game, miners should mine on top of the one which first reached Acceptance Difficulty. Miners may have different views in terms of which block reached the Acceptance Difficulty first due to the network delay. Assuming this winner block is the block X, the blocks of the next block height (block $X + 1$) will embed Shares of block X. According to the Shares integrated, if a miner failed to find the Shares which together weighed more than 50% of the power it previous claimed, this miner will be expelled from the game. This expulsion means the miner's TR will be canceled since BH $X + 1$. The remuneration for the miners of block X is given in the block height BH $X + 1$ as Coinbase transactions. All the valid miners of block X divide the reward based on the difficulty of the Shares they sent. As every miner oversees different Try Range, it is easy to determine which miner should receive what amount of remuneration.

3.1.3. Game procedure

- **Register Power:** Create and submit a New Join to the system.

- **Get a Try Range:** Miners whose New Join are embedded into a block will be assigned with Try Ranges.

- **Mining:** Try to create a block and find a Nonce that fulfills Entrance Difficulty in the miner's TR. If a miner's block has reached the Entrance Difficulty and miners approved this block, miners will try to find a Nonce of Acceptance Difficulty in their Try Ranges.

- **Getting Reward:** If the miner submitted an adequate number of valid Shares for the winner block, the amount of reward would be given at the next block height.
- **Rearrange Try Range and Start Over:** After one round of the game, the invalid miners will be globally expelled. Miners who failed to send Shares which stand for at least 50% of the power they claimed will get their Try Ranges cancelled. New miners will be added as well as the Try Ranges for all the valid miners to be rearranged. After that, a new round of the game starts. Miners who submitted New Join before and were not expelled do not need to register power again to participate in the new round of the game

3.2. *Block simplification*

We use a block simplifies algorithm Graphene [5] to simplify the block as the block size is increased due to embedding New Joins and Shares. Graphene [5] is a block simplify method which combines the Bloom filter [30] and IBLT [31]. Graphene can encode dozens of thousands of transactions into several Kbytes. Graphene encoded blocks can be decoded using the previously received information. Graphene has detailed mechanisms to deal with the failure of decoding. The structure of MWPoW block is given in Figure 8. It is important to simplify MWPoW blocks so that the increase of the participant number will not largely affect the block size. A block of extended size slows down the block broadcasting and may, as a result, affect the fairness of the system (miners received a block faster start to mine the next block quicker). Though the block is simplified, nodes still need to sync all the New Joins and Shares in the system to decode the simplified blocks. However, the bandwidth requirement is not as large as it seems to be because the New Joins and Shares were sent not at the same time but during every iteration. A New Join sized 102 bytes while a Share sized $36 bytes$, according to [19], compared to Nakamoto blockchain, MWPoW only requires around additional $2Kbytes/s$ of bandwidth for a participant to sync all the New Joins and Shares in a bitcoin-like system with 8000 nodes in it and the block interval setting of 10 minutes.

3.3. *Distributed remuneration*

According to the Shares embedded in the block, the compensation for announcing the preceding block is given to the miners in the winner group directly in Coinbase transaction. Figure 9 shows an example of a remuneration distribution, where the sum of the difficulty of the shares sent by the Miner A and Miner B are 212 and 49 respectively, and the sum of the difficulties of all the valid shares of the block is 1000. The total reward amount of last block height is 100. Miner A and Miner B receive 21.2 coins and 4.9 coins, respectively.

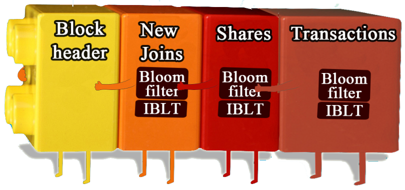


Figure 8: The structure of a MWPoW block

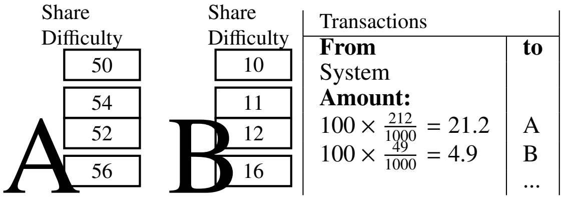


Figure 9: Reward assignment

3.4. *Fast block confirmation*

In the Nakamoto blockchain, nodes have no information about the Support Rate of a block. Miners hold the blocks until the difference of the accumulated difficulty between different fork branches is large enough for nodes to accept the most difficult one as the mainchain comfortably. In which case, the blocks in that branch are finally accepted. However, in MWPoW, by registering power, we know the overall calculation power in the game. By counting Shares, we can acquire how much calculation power has agreed on which branch of the blockchain and the Support Rate of a block can be easily calculated and compared. This procedure waives the need for later block confirmations.

It is predefined that if a miner has sent two Shares for a block, this miner will not be allowed to change branches in this round of the game. Otherwise, it will be expelled, and its contribution will not be count toward the Support Rate. It is also predefined that the miners should mine on the block, which, to their knowledge, first reached the Acceptance

Difficulty. A miner can shift to mine on another block when this miner has not yet sent two Shares for a specific block. A miner will likely do so if there is a block of more Support Rate.

A block is finally accepted when:

- This block is announced and is inside the highest branch of the mainchain;
- The support rate of this block is more significant than 50%;
- The amount of power that supported this block is more significant than the amount of power that backed the second largest block plus 25% of the registered power of the latest block height.

Figure 10 shows an example of the branch choosing where D stands for the difficulty, and SR stands for Support Rate. In (a), when the block A,B, C are announced, none of them get a more than 50% Support Rate; thus, we cannot determine which block is finally accepted. In (b), when there are succession blocks of block A, B, C, the Support Rate of block A, B, C are changed. Block C, and D are finally accepted because they have more than 50% of the Support Rate. Meanwhile, this Support Rate is larger than the Support Rate of either block of the same block height plus 25% of the registered power.

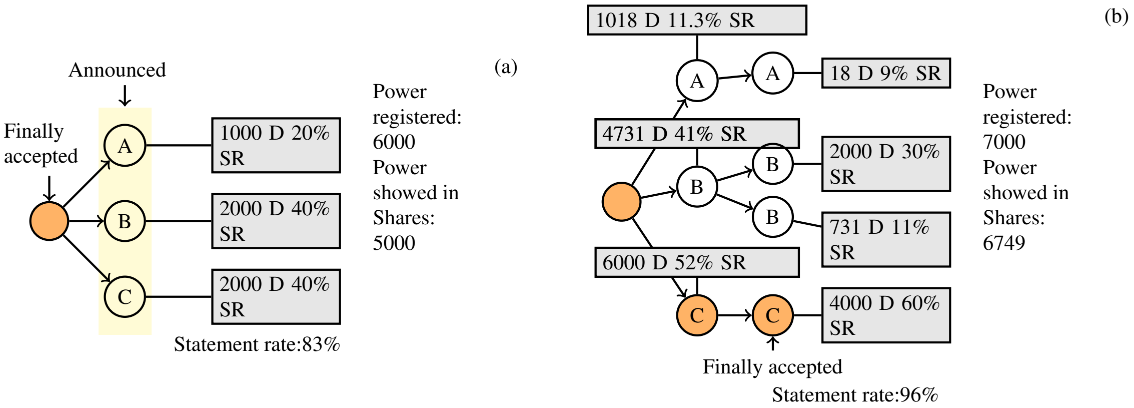


Figure 10: Finally accepting a block

4. Multichain MWPoW

MWPoW provides disadvantaged nodes the ability to profit from mining game and to judge blocks collectively. A finally accepted block can be quickly confirmed if half of the mining power has voted for this block. However, because there are New Joins and Shares, verifying a block is more resource-demanding. Especially when the block interval becomes minuteness, the bandwidth of individual nodes needs to be high to sync all the data on time. Multichain MWPoW provides a solution to these disadvantages and increases the scalability of the system by splitting the transactions into multiple parallel MWPoWs.

4.1. Multichain MWPoW outlines

4.1.1. Definitions

- **Chain ID:** Chain ID is formatted as $C + digits$. Chain ID is given and changed base on the history of the split/merge of chains since the start of the system. A chain can only be split into two at the same time. The two new chains use new names by adding a digit of “0” or “1” to the end of the ID of in binary format, respectively; for example, $C1$ split into $C2$ and $C3$. When two chains are merged, if they stemmed from the same branch, the ID after merging is the old name of that branch. For example, $C2$ and $C3$ were stemmed from $C1$, if $C2$ and $C3$ merged, the name will be $C1$ again. If two chains are merged into one and they did not stem from the same branch, the name for the merged chain is the smaller one of the two Chain IDs before merging. For example, if $C5$ and $C3$ are merged, the new ID would be $C3$.

- **Lifelength:** Lifelength refers to the time (continuous iterations of mining game) that a miner can play in a chain after assigned into this chain. There is a predefined Lifelength Ti; $Ti \ mod \ 4 = 0$.

- **New-Assign-Join:** New-Assign-Join is like New Join in MWPoW but with one additional field: "Identity_Key". To make a New-Assign-Join valid, the hash of this New-Assign-Join must fulfill T times of its Calculation Power Claim.

- **Chain Limits:** Every chain has an upper limit of K and a bottom limit $\frac{K}{2}$ of the number of transactions and New-Assign-Joins per block. When the pending transaction and New-Assign-Join number exceeded/broke the upper/bottom limits, a chain will split into two or merge with others.

- **Ordinary Block:** An Ordinary block (Ob) records the same information as the block in MWPoW records, except it does not record New Joins.

- **Power-assignment Block:** Apart from information of an ordinary block, Power-assignment block (Pab) additionally records New-Assign-Joins. Pab is used to assign the owners of the recorded New-Assign-Joins into different chains. Pab records up to K number of New-Assign-Joins while recording up to K number of transactions. The preceding block of a Pab is an Ob.

- **Assignment Box:** Assignment box is the container of New-Assign-Joins, and it is embedded only in the Pab. There are two sections in an assignment box: New participant section and Re-assignment section.

- **New Join:** New Join in Multichain MWPoW is a data set that contains a New-assign-Join and a Merkle branch. The Merkle branch must prove this New-Assign-Join has been written to a Pab of a particular chain.

- **Fuel-up Block:** Fuel-up block (Fub) of a chain records the New Joins that were assigned by a Pab to this chain after the previous Fub of this chain. The creator of the New Joins recorded in this Fub can start the mining game in this chain after the current block height (Try Ranges are assigned). The preceding block of a Fub is an Ob, where this Ob's preceding block is a Pab.

- **History / OffSptring Chain:** When a chain is merged/split, the new chain(s) is the Offspring chain of this chain. This chain becomes a history chain of its Offspring chain(s).

- **Duty Range:** A range of transactions/New-Assign-Joins, which should be processed by a chain.

- **TransOnhold:** TransOnhold is a number added to the block header. This number stands for the number of transactions/New-Assign-Join received by the creator of the block. These transactions/New-Assign-Joins should be legal and within the Duty Ranges. In the meantime, they should have not yet been written into a block in the mainchain of this chain.

- **Chainpower:** The amount of the overall registered power (in PoW difficulty form) inside a chain.

- **Threshold Chainpower:** Rank the CP of the participants inside a chain in ascending sequence, place the ranked sequence in a list $RCP_{0...NPC-1}$. NPC is the number of registered participants inside this chain. Threshold Chainpower is the sum all the values from $RCP_{0...\lfloor\frac{2}{3}\times NPC-1\rfloor}$

- **Bl_candidate:** An integer array of Sg items indicated in the block header,

$$Bl\_candidate(i) = RCP_{\lfloor i\times(NPC/Sg)\rfloor}, i \in [0.Sg) \tag{21}$$

- **Global Block Header:** Global block header is a Merkle root of the hash of all the latest finally accepted block of all chains.

- **Crosschain Section:** When one transfers a transaction between chains, the transaction is written into the Crosschain section.

4.1.2. *Amendment to the designs of MWPoW*

The following amendments are made to the designs of MWPoW:

1. Three types of blocks: Ordinary block (Ob), Power-assignment block (Pab), and Fuel-up block (Fub) take turns to be written into the chain, repeating the sequence of Ob, Pab, Ob, Fub.
2. Fub records New Joins; however, Ob and Pab do not record New Joins.
3. The Block interval time of every chain is set to be the same number. Thus all chains generate blocks in an approximately same time window.
4. $TransOnHold$ is placed into the block header.
5. $Share$ is signed by the private key of the Indentity_Key of its creator.
6. $Chainpower$ is added to the block header.
7. $Bl\_candidate$ is added to the block header.
8. $Threshold\ chainpower$ is added to the block header.
9. $Number\ of\ participants$ is added to the block header, a number which states the number of valid registered miners inside a chain.
10. The block in every chain records a list of valid registered miners inside its chain. [1]
11. Every block embeds a Global block header; the hash of the Global block header is written into the block header of every block. Global block header records the hashes of the latest finally accepted block of all the chains. We allow these hashes to be the second latest one because the block generation among chains goes incomplete synchronised.
12. Nodes only hear the blocks of the chain they were assigned into as well as the block header of the announced blocks from other chains. When a block is announced, miners in all chains should download the block header of this block.
13. Apart from the rules of MWPoW regarding final accepting a block. A block reached Acceptance Difficulty is finally accepted when the chance for the block to be evil is lower than the security threshold. This chance is calculated using equation 12. If there are two blocks which reached the Acceptance Difficulty at the same epoch in a chain and the chance for them to be evil are lower than the security threshold, the one (*Alice*) with more Support Rate is finally accepted if the differences between the Support Rate of the two blocks is more significant than a specific value. This value is defined as one that the adversary can gain with the pre-defined security threshold probability. The chance for the adversary to control this particular value of Support Rate differences can also be calculated using equation 12 by enumerating some voters of *Alice* and assuming the enumerated voters are evil (only use the enumerated votes to calculate the $DG(i)$ and $NgS(i, j)$). The enumerated evil voters together should contribute the amount of differences between the Support Rate of *Alice* and the other block. The chance for the enumerated voters to be evil should be lower than the security threshold probability.

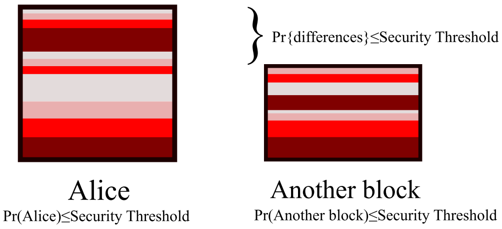

Pr{differences}≤Security Threshold
Alice
Pr(Alice)≤Security Threshold
Another block
Pr(Another block)≤Security Threshold

Figure 11: The explanation of the additional rules for accepting a block

Figure 12 shows the structure of Multichain MWPoW.

[1] In the original MWPoW, the participant list is not written in the block, which can be derived by counting the New Joins and Shares since the beginning of the system. Including participant list does not largely increase the bandwidth demand because the block is encoded using Graphene. Nodes do not need to swap any clear text of the participant list unless a discrepancy is detected.

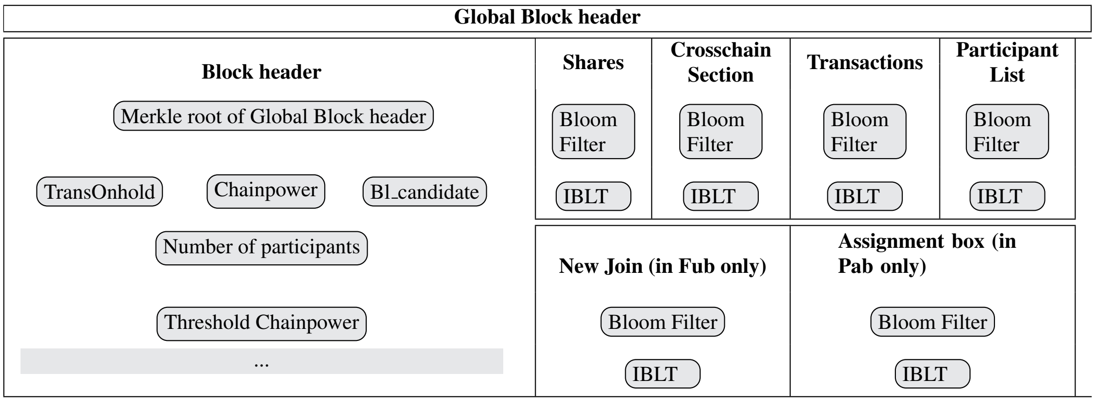


Figure 12: The Block of Multichain MWPoW

4.1.3. Game procedure

- **Register power:** The participant creates a New-Assign-Join based on a Fub of a chain (HashPrevblock should be the hash of that Fub). After the New-Assign-Join is constructed, usually Ti iterations of the game have passed as the hash difficulty of this New-Assign-Join must reflect T times of its Calculation Power Claim. The participant then sends the New-Assign-Join to that chain.

- **Wait for the power assignment:** In every four iterations (whenever a Pab is created), up to K number of qualified New-Assign-Joins of new participants is selected by miners in a chain. A random assignment protocol is used to place all the selected New-Assign-Joins into the assignment box of the new Pab.

- **Register with the chain assigned to:** After a Pab *Alice*, which embedded the participant's New-Assign-Join, is announced (reached Acceptance Difficulty), the participant then creates a New Join, which contains that New-Assign-Join and a Merkle branch. The Merkle branch should prove this New-Assign-Join has been assigned to a specific chain by *Alice*. Finally, the participant should submit this New Join to the chain assigned by *Alice*.

- **Get a Try Range:** Miners in the assigned chain check if the New Join they received is valid. They also check if the Pab (*Alice*), which made the assignment, is the latest finally accepted Pab in its chain. A Try Range is given to the participant at the next Fub, then the original rules of MWPoW begin to apply.

- **Split and merge the chains:** When the chain violated chain limits, the latest block will indicate if the chain should be split or merged. The miner then entered the split or merged chain following the rule of split/merge.

- **Reassignment:** When a miner has inside a chain for Ti rounds of the game, it is reassigned to another chain.

- **Expel:** The same rule as MWPoW, if a miner did not send at least three valid Shares per iteration which successfully embedded to the block, it is expelled.

Figure 13 shows an example of the game procedure of Multichain MWPoW.

4.2. Global parameters

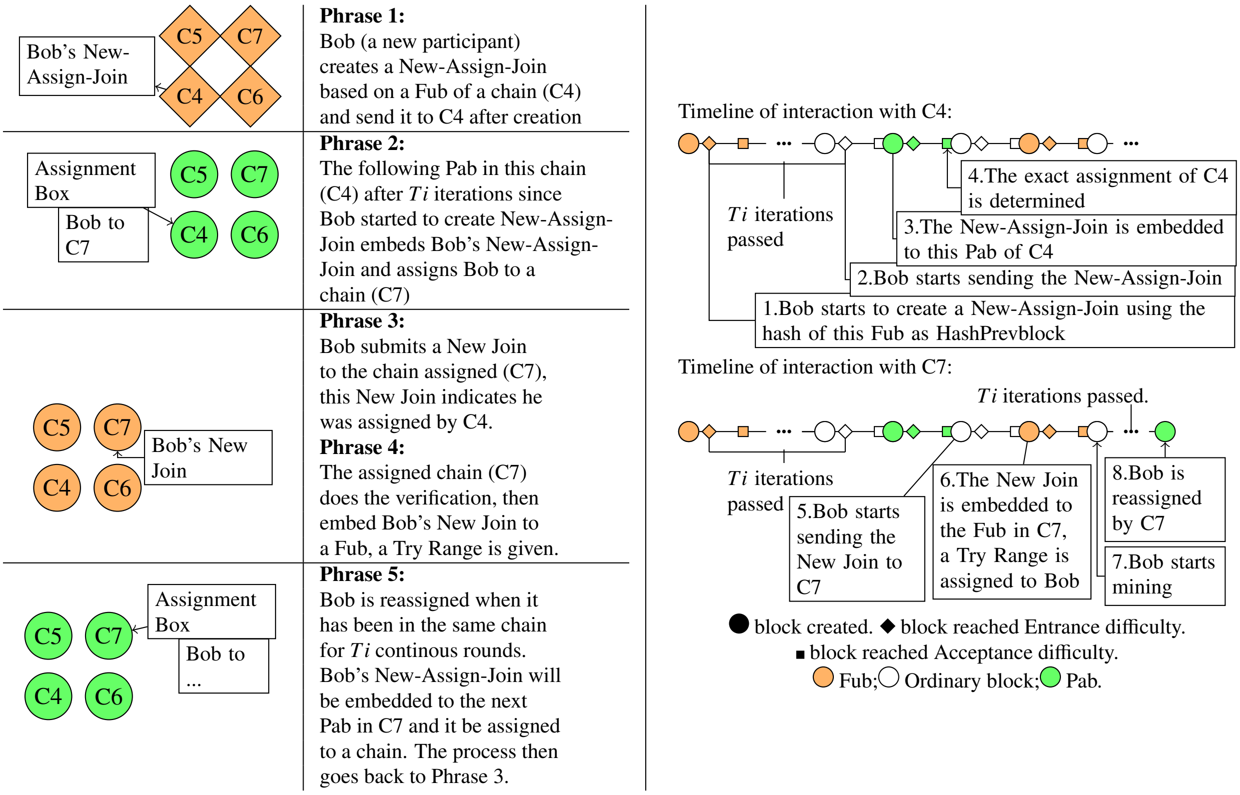
4.2.1. Group boundary

The number of nodes in the system can be derived by adding the number of participants indicated in the block headers of the latest finally accepted blocks in every chain together. Let there be NC number of chains,

$$bl(i) = min(Bl\_candidate(i, j)), i \in [0, sg), j \in [0, NC) \tag{22}$$

where $Bl\_candidate(i, j)$ refers to the $Bl\_candidate(i)$ of the latest finally accepted block in chain j. Miners of a chain j then classify the nodes inside the chain j according to the bl derived. Every time the group boundary is determined, miners should examine if some restrictions are met.

The restriction includes:



Phrase 1:
Bob (a new participant) creates a New-Assign-Join based on a Fub of a chain (C4) and send it to C4 after creation

Phrase 2:
The following Pab in this chain (C4) after Ti iterations since Bob started to create New-Assign-Join embeds Bob's New-Assign-Join and assigns Bob to a chain (C7)

Phrase 3:
Bob submits a New Join to the chain assigned (C7), this New Join indicates he was assigned by C4.

Phrase 4:
The assigned chain (C7) does the verification, then embed Bob's New Join to a Fub, a Try Range is given.

Phrase 5:
Bob is reassigned when it has been in the same chain for Ti continous rounds. Bob's New-Assign-Join will be embedded to the next Pab in C7 and it be assigned to a chain. The process then goes back to Phrase 3.

Timeline of interaction with C4:

1.Bob starts to create a New-Assign-Join using the hash of this Fub as HashPrevblock

2.Bob starts sending the New-Assign-Join

3.The New-Assign-Join is embedded to this Pab of C4

4.The exact assignment of C4 is determined

Timeline of interaction with C7:

5.Bob starts sending the New Join to C7

6.The New Join is embedded to the Fub in C7, a Try Range is assigned to Bob

7.Bob starts mining

8.Bob is reassigned by C7

● block created. ◆ block reached Entrance difficulty.
▪ block reached Acceptance difficulty.
● Fub;○ Ordinary block;● Pab.

Figure 13: The procedure of Multichain MWPoW

1. $2 \times Threshold\ Chainpower \geq Chainpower$.
2. There is at least one node from every group in this chain.
3. $Max(Pr_j) \leq Threshold$, where $Threshold$ is the predefined security threshold.

If the restrictions are not met, the chain j should be merged with others.

4.2.2. Global block header and dispute resolution

Because nodes only sync information of its chain and the block headers of the announced blocks of other chains, nodes are unable to determine if a block of another chain is genuine and finally accepted. To solve this, we propose a mechanism:

1. When a block is announced or finally accepted, relevant miners should broadcast this information to miners of other chains.
2. Miners should periodically ask several miners in other chains to see if the announced blocks have been finally accepted.
3. If a conflict is known to a node, this node should sync the participant list in the last block before suspicious one. It should then determine the genuine Shares and calculate the support rate of the blocks.
4. A block carried a wrong finally accepted block hash in its global block header should be rejected. The miner should not mine on this block in any circumstances.

Because the calculation power is distributed in chains, it is easier for Byzantines to over-write specific blocks in a chain using a greater calculation power. To prevent unregistered nodes affecting the generation of blocks, we rule that:

1. The Shares sent to the network should be signed by the Identity_Keys which were claimed in the New-Assign-Joins;

2. The Nonces should be within the Try Range that associates to the Identity_Keys.

Under this mechanism, the first block of a fraud chain of blocks can be determined as invalid because byzantine cannot provide the correct Shares which signed by the previous participants.

When a new block of a chain which fulfills the Entrance difficulty comes out, nodes of that chain should check the global block header of that block before contributing Shares for it. In this way, when a block is announced, at least a certain amount of calculation power agrees on the global block header attached. Because nodes sync all the block headers of the announced blocks of all the chains, they can see the differences between the Merkle root of all the global block headers. A node will request and verify the relevant global block headers if it cannot construct the same Merkle root of the global block header. Figure 14 is an example of a global block header, where NC is the Chain ID, and LASH is the hash of the latest finally accepted the block.

NC	LHASH
C2	EA232341AEAFEWER2EKWFL23EWRKL
C6	FB1113A122FIAQFXWSLEEF23ERK1LR4
C7	CCA313A152FIAQF1AWLEWAE3WFETQ

Figure 14: Global block header

In a brief summary, if a Byzantine attempts to change a finally accepted block of a chain, it must place enough power to the inside of this chain through the normal procedure. If the power is not registered before, it cannot generate valid Shares, nodes inside the chain will not recognise an invalid block which reached the Acceptance Difficulty. When nodes of other chains ask which block is finally accepted, or the honest miners inside a chain received a fraud block of that chain from the network, the honest registered power inside that chain will appoint another block to the network. When a conflict of finally accepted block occurred, the Byzantine's block cannot pass the verification of other chains.

4.2.3. Duty Range, Chain split and merge

Duty ranges for a chain include all the New-Assign-Joins, the transactions, and New Joins which:

- The HashPrevBlock of the New-Assign-Joins is a block inside this chain.
- The HashPrevBlock of the New-Assign-Joins indicates a block in the history chain of the current chain. The hash of this New-Assign-Join is within a specific range.
- All the Input transactions of the transactions were committed to any block of this chain.
- All the Input transactions of the transactions were embedded in the history chains of the current chain. The hash of these Input transactions is within a specific range.
- The New Joins which indicated their creators are assigned to this chain.
- The New Joins, which indicated their creators are assigned to the history chains of this chain, and the hash of the New Joins are within a range.

The valid New-Assign-Joins, transactions and New Joins of a chain complies:

- They are under the government of this chain (inside the Duty Range).
- The INPUT transactions are not used before.

When $TransOnhold$ indicated in the latest finally accepted block of a chain is more substantial than $2 \times K$, then this block is split into two since the next block height. However, a chain cannot be split when either of the split chains will not meet the chain restrictions stated in section 4.2.1.

When a chain $C1$ is split:

- **Duty Ranges:** According to the hash of the transactions written in the blocks of chain $C1$, if the hashes of the transactions are within the range of 0 to 2^{255} then these transactions are governed by chain $C2$. Otherwise, the transactions are governed by chain $C3$. The duty ranges inherited from chain $C1$ are also equally split into two. Chain $C2$ will take the duty ranges of $C1$ with lower half hashes while the chain $C3$ will take the upper half. The rule also applies to the New Joins on hold. If the hashes of which are within 2^{255}, then the New Joins are processed by $C2$. Otherwise, they are processed by $C3$.
- **Participants:** Rank all the participants by the amount of their *Calculation Power Claim* in ascending order, a participant is relocated to $C2$ if $Pindex\ mod\ 2 =: 0$ where $Pindex$ is the index number of this participant inside the ranked participant sequence, otherwise this participant is relocated to $C3$.

Figure 15 shows an example of the chain split and merge, where the system starts from one chain $C1$, the squares in orange are blocks in the chains that are currently existing while the squares in gray are blocks in history chains.

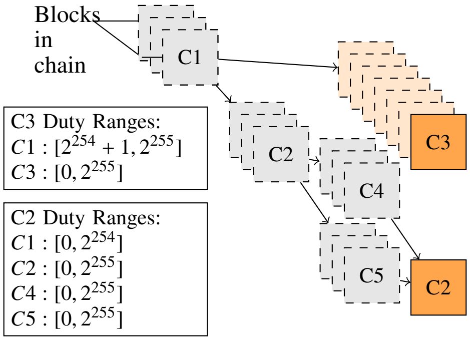


Figure 15: Chains overview

When merging, a chain will be merged with another that is closest to it in Chain ID. If there are two chain candidates, select the one of the smaller Chain ID. The duty ranges of the chains are also merged.

Assume chain $C5$ is merged into another chain $C3$ after a block *Alice* in $C5$ is announced. When *Alice* is announced, the miners in $C3$ is aware of this merging because they sync the block header of all the announced blocks. The miners in $C3$ then sync the data in $C5$ between the block interval of the last finally accepted block of $C5$ and *Alice*. The miners use this data to verify *Alice*. If they believe $C5$ should be merged with $C3$ according to the rules, they will mine on the merged chain. When a safe number of nodes in both $C3$ and $C5$ has approved this merge through mining in the merged branch, then the merge is completed. This safe number can be calculated using equation 12. When chain $C5$ has not generated a finally accepted block for five continuous block interval, the chains which the chain $C5$ is possible to merge into should sync the data from C5 and determine if they should merge with $C5$. The merged chain starts at the next block height of the highest block height in its history chains. When a chain $C3$ seeking to merge to $C5$, $C5$ is also seeking to merge; if $C5$ is trying to merge with another chain $C6$, then three or more chains merge to one at the same time. Figure 16 shows an example of the chain merge and split, squares in gray are abandoned blocks. A sufficient number of nodes in $C3$ and $C5$ agree on merging in block height 14, other branches of them are then abandoned.

4.3. Crosschain operation

Because every chain can confirm the situation of blocks in other chains (has been / not yet finally accepted), we take advantage of that to conduct crosschain operations. When a user wants to transfer a transaction to another chain, it first sends the cross-chain-request to the chain that governs the transaction (Origin chain). If this cross-chain-request is written into the crosschain section of a finally accepted block afterward, the user then sends a cross-chain-confirm to the transfer destination chain. The cross-chain-confirm is a Merkle branch that can prove the cross-chain-request has been written into the crosschain section. The destination chain should write the cross-chain-confirm into its crosschain section, and then the transaction is transferred. The difference in block height between the cross-chain-request and the cross-chain-confirm embedded the blocks should be less than three. If the cross-chain-confirm is not able to be written into the destination chain on time, the user will ask the origin chain to cancel the cross-chain request. The miners in the original chain will acquire the cross-chain section of relevant blocks of that destination chain and determine if the transfer should be cancelled. If the user does not send the cancel request, the transaction is being transferred to the destination chain. Figure 17 shows an overview of the cross-chain operation. New transactions of the destination chain can refer to the cross-chain-confirms written in the cross-section of this destination chain as the INPUT transactions.

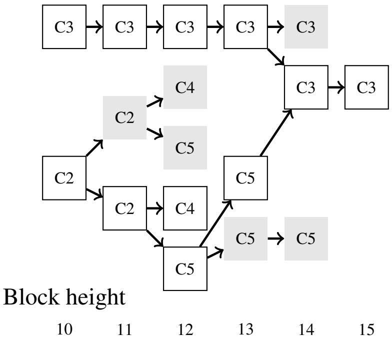


Figure 16: An example of chain merge/split

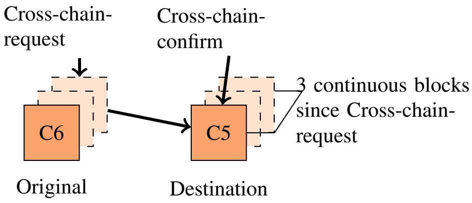


Figure 17: Crosschain operation overview

4.4. Power assignment block

In this section, we show the procedures of forming a Pab for a chain $C5$. There are two parts in forming a Pab: Periodical power reassignment and New power adding. The structure of the Assignment Box and New-Assign-Join is shown in Figure 18 and Figure 19 respectively.

4.4.1. Periodical power reassignment

Select the nodes which were added to $C5$ at the block height $BH - Ti$, where BH is the current block height. Place the selected nodes' New-Assign-Join into a list PSL by the ascending order of the calculation power claim indicated in their New-Assign-Joins. Create a new sequence $RPSL$,

$$RPSL_i = Hash(MGBH \oplus Hash(PSL_i)) \tag{23}$$

where $MGBH$ is the Merkle root of the global block header indicated in the latest block of $C5$. Link PSL_i with $RPSL_i$ and rank $RNAJ$ by alphabetical order. After that, a new index of PSL can be reached. Let there be a NC number of sub-sections in the "Re-assignment section" of the Assignment box. $Acs(i)$ represents the number i sub-section,

$$Acs(i) = \bigcup_{(hash(PSL_j)+j) \ mod \ NC=i} PSL_j \tag{24}$$

Nodes in $Acs(i)$ are assigned to chain i. If chain i becomes a history chain right after, nodes in $Acs(i)$ are assigned to its OffSpring chains according to the Duty Range.

4.4.2. Adding New Power

Miners in $C5$ takes the New-Assign-Joins from all the unassigned New-Assign-Joins received, which fulfilled the following criteria :

- The Nonce inside can make the hash of this New-Assign-Join fulfill the Intended_difficulty.
- The HashPrevBlock is the hash of the Fub at Ti iterations before the current block height.

After selecting the New-Assign-Joins, the following procedures are carried out:

1. Let InD be the Intended Difficulty indicated in a New-Assign-Join. If $bl(i + 1) > InD >= bl(i)$ then place this New-Assign-Join to list i. $bl(Sg) = +\infty$.
2. Rank the New-Assign-Joins in every list i by the ascending order of $abs(InD - tt)$, where $tt = \frac{bl(i+1)+bl(i)}{2}$. Specially, in this step, $bl(Sg) = bl(Sg - 1)$.
3. Select K/Sg number of New-Assign-Joins from the top of every list. If a list has less than K/Sg number of New-Assign-Joins, then take all the New-Assign-Joins in them.
4. Rank the selected in descending order of their power claim, sum the front $\frac{1}{3}$ fractions the power claims. If that is larger than the half of the overall power of the selected New-Assign-Joins, then delete the New-Assign-Joins from the top until the front $\frac{1}{3}$ fractions of the power claims are not more significant than the half of the overall power of the selected New-Assign-Joins.
5. Rank the remaining New-Assign-Joins according to the alphabetical order of their hashes and place them into a list NAJ. Create a new sequence $RNAJ$,

   $$RNAJ_i = Hash(MGBH \oplus Hash(NAJ_i)) \tag{25}$$

   Link NAJ_i with $RNAJ_i$ and rank $RNAJ$ by alphabetical order. After that, a new index of NAJ can be reached.
6. Let the "New participant section" in the assignment box assigns New-Assign-Joins to $min(NC, K)$ number of chains. $NAJ_{i\ mod\ min(NC,K)=j}$ is assigned to the number j chain indicated in the assignment box, NC is the number of chains.
7. Write the assignment plan into the "New participant section" in the assignment box.

Assignment Box		
New participant section		
Ll	New-Assign-Joins	Intended_Difficulty
0	$[NAJ_3]$	$[CP_3]$
1	$[NAJ_1]$	$[CP_1]$
2	$[NAJ_2]$	$[CP_2]$
Re-assignment section		

Figure 18: Assignment box

New-Assign-Join	
HashPrevBlock	The hash of the latest block in the mainchain of the chain.
Intended_Difficulty	Calculation Power Claim.
Wallet_address	Used for receiving rewards.
Identity_Key	A public key of a public-private key pair.
Nonce	Number (256bits) that makes the hash of this New-Assign-Join fulfill the Intended_Difficulty.

Figure 19: New-Assign-Join

4.4.3. Determine the exact assignment

Any chains accept the New-Assign-Joins which assigned to them if these New-Assign-Joins are written in a "Re-assignment section". If a New-Assign-Join *Alice* claimed she was assigned to a chain *Ben* by a Pub *Gary* of *C*5 in "New participant section", *Ben* verify this information by:

1. Let there is a *Ll* number of subsections in the "New participant section" of the Assignment box of *Gary*. Let there are *NC* number of chains, rank chains by the alphabetical order of the Chain ID.
2. The New-Assign-Joins in the number $i, i \in [0, Ll)$ subsection of the Assignment box of *Gary* is assigned to the number $Hash(Gary)\ hash(MGBH + i)\ mod\ NC$ chain in the ranked sequence.

If it is verified by the above procedure that *Alice* is assigned to *Ben*, then *Ben* should accept *Alice*.

4.5. Fuel-up block

Miners need to send a New Join to the chain which they were assigned in. The New Joins are embedded in the Fuel-up block, and Try Ranges are assigned afterward. Figure 20 is the structure of New Join. The New join for any chain *Ben* is a valid one when:

1. The Merkle Branch and the hash of the New-Assign-Join attached can form the Merkle root of the Assignment Box of the chain who made the assignment.
2. The New-Assign-Join is assigned to *Ben*.
3. The Pab which made this assignment is the latest finally accepted Pab of that chain.
4. This New Join is not used previously.

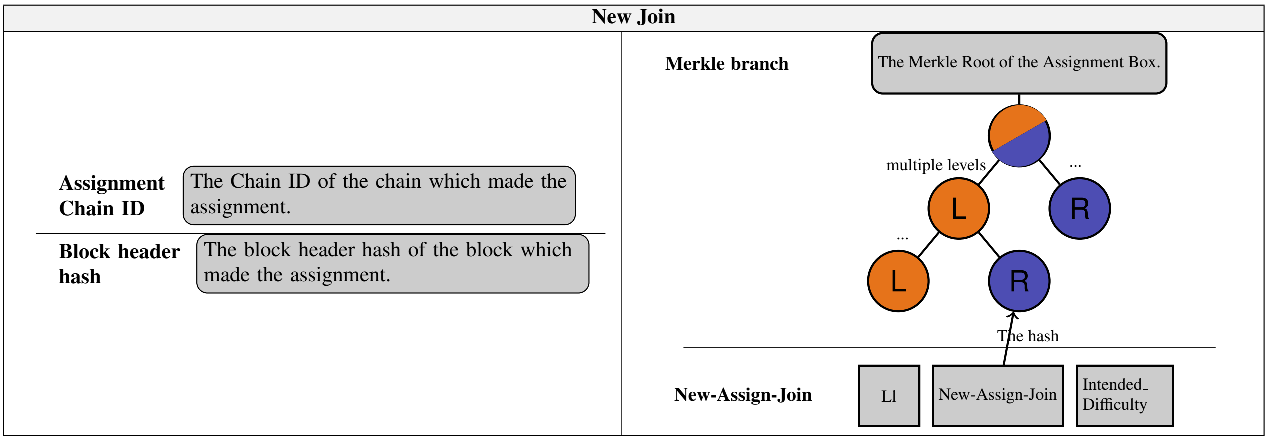

New Join
Assignment Chain ID
The Chain ID of the chain which made the assignment.
Block header hash
The block header hash of the block which made the assignment.
Merkle branch
The Merkle Root of the Assignment Box.
multiple levels
...
L
R
...
L
R
The hash
New-Assign-Join
Ll
New-Assign-Join
Intended_ Difficulty

Figure 20: New Join

4.5.1. Power adjustment

When the New Join is valid, it is recorded into a Fub of chain *Ben*. The miners of chain *Ben* acquire the Intended_Difficulty of the New Joins and add the nodes into the chain. However, after adding, if the restriction $2 \times Threshold\ Chainpower >= Chainpower$ is not met, then the Intended_Difficulty of the new assigned nodes in $group(Sg - 1)$ is lowered to $bl(Sg - 1)$. If the restriction is still not met, the chain will be merged with others. The adjusted Intended_Difficulty is then used as the reference for assigning a Try Range.

5. Security analysis

5.1. Solving the hypothesis challenges

5.1.1. Random distribution

The randomness of the assignment is safeguarded by *MGBH* in each step. *MGBH* is changed by every operation, every transaction embedded to the blocks at the time. An attacker cannot control everything that happened in the system and as a result, making *MGBH* impossible to be pre-calculated. Also, *Bl* is hard to be predicted because nodes can join in the system and leave the system freely at any time. A node added or dropped causes a shift in *Bl*. It is also impossible to predict when the New-Assign-Joins that fulfill the selection criteria would appear on the internet because that is up to the participants globally. For the above result, the attackers cannot control the rank of its New-Assign-Joins in the sequences, making it impossible to pre-calculate which chains their New-Assign-Joins will be assigned. Also, as it requires Ti times of Calculation power claim to qualify a new miner, it is of no gain to quit a chain and repeat the assignment procedure. It takes the same effort to get the miners reassigned to other chains regardless it is a chain or out of a chain currently.

5.1.2. Determine block legibility of other chains

The protocol discussed in section 4.2.2 provides a "detec" and "verify" then "synchronization" procedure. When a block is announced, the block header of it then flows to the whole network. When a conflict is detected, miners of other chains can distinguish and recognise the genuine block by acquiring Shares during the conflict and the participant list before the conflict. The recognition is written in the block header as the global block header, which is synchronized and verified by miners globally. We secure the gateway to the inside of every block by the random assignment. Meanwhile, the blocks created by the power outside a chain are not recognised globally, this secures (2) and (3) of the blockchain hypothesis Challenges.

5.1.3. The number of honest participants and the proportion of honest power

We use the indicator $Threshold\ chainpower$ to describe if the power distribution inside a chain is balanced and secured. If the number of participants in a chain cannot guarantee a safe result, we will merge this chain to others.

5.1.4. Categorisations

We use the property of ranked sequence to make the nodes into different categorisations. Nodes can use different power to make a rough selection of the group, but it is up to the situation of the contemporary nodes to determine the groups eventually. The queues are automatically divided into equal length, few operations to maintain the system is needed, not like previous $n/2$ approaches requiring strict restrictions. The design not only satisfies the challenge of the $n/2$ BS hypothesis, but it also brings flexibility and stability.

6. Data analysis

Miners are required to sync block headers of the announced blocks of all the chains and the blocks inside the chains they were assigned. Table 4 shows the minimum size of a Block header, Share, New Join, or New-Assign-Join in Multichain MWPoW.

Table 4: The minimum size of structures in Multichain MWPoW

Name	Size	Description
Block header	124 + Sg× 4 bytes	Three 256-bits hashes: the hash of the preceding block,*MGBH*, and the Markle root of the transactions. Seven 32-bits integers: Chainpower,TransOnhold, Threshold Chainpower, Number of participants, Timestamp,Entrance Difficulty,Acceptance Difficulty. Sg number of integers: Bi_candidate.
Share	64.5 bytes	A 4-bits integer(the last four bits of the block hash),a 256-bits integer (Nonce),and a 256-bits signature.
New-Assign-Join	132 bytes	A 256-bits hash (HashPrevBlock), a 32-bits integer (Intended_Difficulty), three 256-bits integers (Wallet address, Identity_Key and Nonce).
New Join	12+32+$log_2(K)$× 32 bytes	Three 32-bits integers (*Ll*,Intended_Difficulty and Assignment Chain ID). A 256-bits hash (Block header hash) $log_2(K)$ number of 256-bits hashes (Merkle branch).

The participants send shares during every iteration after joining in a chain. The majority of New-Assign-Joins are only broadcasted at block heights before a Pab in one iteration interval. The New Joins are broadcasted between a Pab is finally accepted, and before the next Fub comes out (also one mining interval). Thus, the minimum upload bandwidth required for a miner in the most data heavily iteration is $Max(Size_{New\ Join}, Size_{Share} \times 4, Size_{New-Assign-Join})$. The download bandwidth for a participant in the most data-heavy iteration in a system with NPC number of participants inside the chain is $NPC * Max(Size_{New\ Join}, Size_{Share} \times 4, Size_{New-Assign-Join}) + Size_{Transactions} \times K$. Figure 21 shows the download bandwidth requirement and transaction throughput globally with $n = 8000$ and different number of NC and K. NC is ranged from 1 to 400 ($\frac{n}{20}$, 20 participants per chain).$Sg = 20, NPC = \frac{n}{NC}$, while K ranged from 2 to 1000.

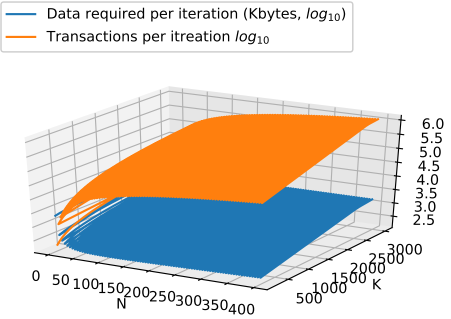


Figure 21: Data requirement and Throughput per iteration with different K and n when $N = 8000$.

7. Experiment

In this section, we experimentally evaluate the overall performance of Multichain MWPoW, test whether it genuinely scales. We compare its performance with RapidChain [16] and $n/2$ Byzantine node resistant blockchain sharding approaches [28, 29] regarding throughput and transaction confirmation time with different percentage of adversary power in the system. In this experiment, we maintain a 10^{-6} failure chance for every approach. We use a regulated layout of Distributed Ledger Network [32] as the communication protocol used for the essential P2P connections. The connections and the network structure are dynamically adjusted to fit into the data flow to make data propagation fast.

7.1. Experimental setup

7.1.1. Multichain MWPoW

We simulate 8000 nodes in a network with $10 Mbytes/s$ bandwidth per node. We give every connection a random delay time ranged from $1ms$ to $200ms$; the distribution of the connection delay time is shown in Figure 22. We have three scenarios A, B, and C of calculation power for every node, which are shown in Figure 23. In the experiment, we set K to be 2000, meaning blocks can contain up to 2000 transactions per block, and a Pab can contain up to 2000 New-Assign-Joins in the New participant section of the assignment box. When blocks are broadcasted inside a chain, the blocks will be encoded by Graphene [5] as like original MWPoW. If a block is requested by nodes outside the chain or is requested by a new participant when it is syncing data, the block sent will be the one which decoded from Graphene.

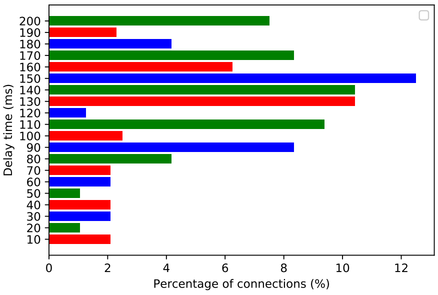


Figure 22: Delay time distribution

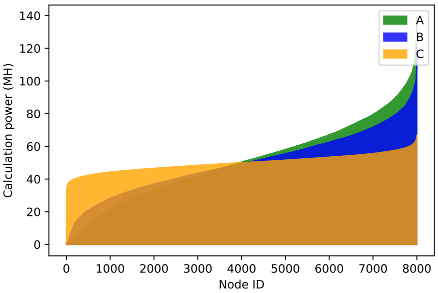


Figure 23: Power distribution

7.1.2. Other approaches

The simulated RapidChain and the two existing n/2 blockchain sharding approaches are implemented in a complete copy of the Multichain MWPoW network settings introduced in section 7.1.1, except the protocol runs on every node is not Multichain MWPoW and every node is equal in voting.

7.2. Experiment design

We send 10^6 transactions per iteration to the network by random nodes (equal allocation). The size of one transaction is $500 bytes$ fixed. We record the number of chains (shards) in the system, and the throughput (the number of transactions that were processed globally per iteration). We also record the transaction confirmation time and the frequency of data refreshing (the frequency of a node being reassigned). For Multichain MWPoW, we increase the number of evil power from zero to 50% of the overall power during the attack, the number of malignant nodes may be higher than half of the node population. For other approaches, we increase the number of evil nodes from zero to 50% of all the nodes during the attack. We set the block interval to be 10 seconds globally; the experiments were conducted during 1000 block intervals. Ti is set to be 20 block intervals and $Sg = 20$. For n/2 blockchain sharding and flexible n/2 blockchain sharding, the m is set to be 33 at the beginning, and T is $0.7 \times m$. The flexible n/2 blockchain sharding approach may adjust this number during the experiment. The number of shards is set as 55 for RapidChain to maintain the security threshold. For RapidChain, 20% of the transactions are multiple input Shard transactions: a transaction must be confirmed by all the input shards to proceed. All the approaches used in this experiment maintains a 10^{-6} failure probability.

The adversary node in Multichain MWPoW will function as an honest node if it does not have enough evil compansions in the chain. When there are enough evil nodes inside to halt this chain, the evil nodes will function maliciously by attempting to create corrupt fork branches. The adversary node in RapidChain has a 50% chance to drop out in every ten iterations. The dropped the node will apply to join in the RapidChain again immediately. The adversary nodes will start to create evil blocks when they controlled the Shard. The adversary nodes for the two n/2 blockchain sharding approaches will correctly function when they do not have enough nodes to halt the shards. They will halt the Shard immediately when having enough number of evil companion. We set a transaction in the first block as the initial transaction. The inputs of transactions are randomly selected from the transactions in previous blocks. In the experiment, Multichain MWPoW starts with one chain named $C1$. Nodes were added to the system followed the rule of Multichain MWPoW as soon as possible. Nodes in other approaches are also added following the rules as quickly as possible. When increasing the adversary percentage, we randomly select the honest nodes in the system and turn them into evil nodes to make the percentage of evil power as the set percentage.

7.3. Experiment results

The experiment lasted 1000 block intervals. Figure 24 shows the changes of the Chain (shards) in the progress of block interval. As can be seen from the results, for Multichain MWPoW, the number of chains and the processability are dynamically adjusted to fit into the data flow and conquer the adversary power from halting the chains. C power distribution scenario is generally more steady than A and B, mostly because the power is more balanced. From 25, we see that the $n/2$ BS approach stops function after the adversary took 33% of the nodes, although the result in them are still correct. We let RapidChain stop functioning after 33% of power taken by the adversary because the security of RapidChain is wholly broken. Flexible n/2 approach also uses K to indicate the pending transactions. We see the curve of the transaction processed by flexible n/2 is drastic, and this is different from the pattern in Figure 24, which can indicate what number of the processed transactions should be. This difference is because when a Shard is halted in both n/2 and flexible n/2 approach, the Shard is frozen until new memberships replaced the old nodes in this Shard. The halting problem is also why n/2 approach has slight fluctuation when functioning. When a global halt occurred, the transaction per second reduced to zero, and the system takes a few intervals to recover from the halting. However, in Multichain MWPoW, the system would not stop processing transactions; the halting is only about when the blocks would be finally confirmed. Figure 28 shows the times of data refreshing during the experiment. As the experiment setup stated, there is a $\frac{1}{20}$ chance for the adversary nodes in RapidChain to quit and rejoin the Shard immediately. Because when some nodes assigned to a Shard at one moment, the same number of old nodes in this Shard will be reassigned to other shards. This design causes the majority of refreshing in RapidChain. There is no limitation of how many times a node can join or leave the system so that the attacker can make this attack in reality on a larger scale. This attack could also work for both n/2 and flexible n/2. However, in the experiment, the adjustment for n/2 approaches are used mainly to solve the halting problem. For Multichain MWPoW, the adjustments are primarily occurred for solving the local halting and adjust to the data flow (changes in the number of pending transactions). Figure 26 shows the average transaction per iteration during the experiment. The transaction per iteration is zero for

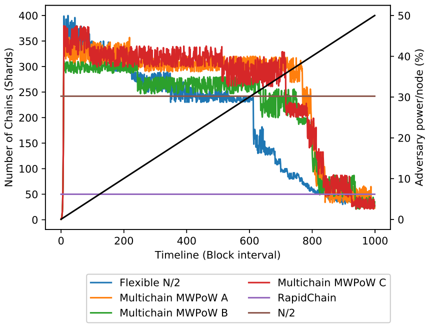


Figure 24: The number of chains/ shards.

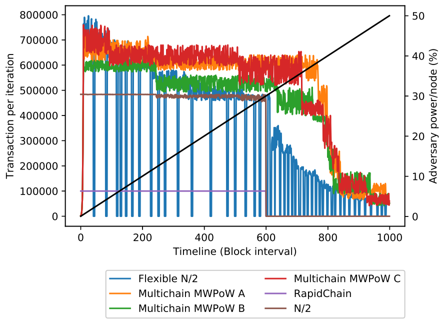


Figure 25: The number of transactions processed per iteration.

n/2 blockchain sharding approach and RapidChain after the adversary taken 33% of the nodes. Figure 27 shows the average pending time for nodes to accept a transaction finally.

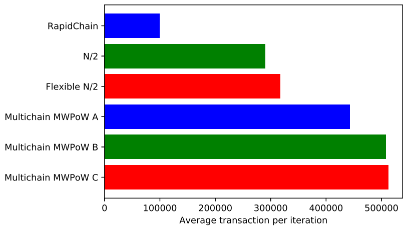


Figure 26: The number of transaction per iteration.

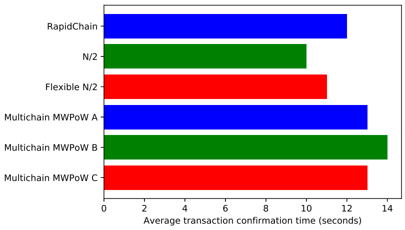


Figure 27: Transaction confirmation time. Recorded from the time a transaction is embedded to a block, and this block is finally accepted.

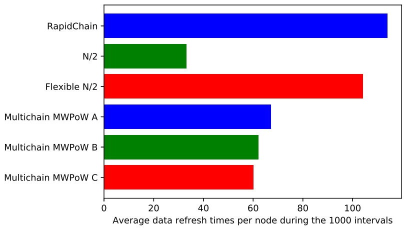


Figure 28: Data refreshing times.

8. Conclusion

We presented an implement of MWPoW in a multichain scenario, reached a quick and robust decentralised autonomous organisation architecture. Multichain MWPoW is the first blockchain sharding approach that can withstand up to 50% of adversary power without assuming the honest people create as many nodes in the system as possible. By experiment, we showed that Multichain MWPoW largely outperforms Rapidchain, n/2 blockchain sharding approach [28] as well as the flexible n/2 blockchain sharding approach [29] in terms of stability, throughput, and transaction confirmation time. We secured a random distribution mechanism and maintained a threshold distribution of power inside every chain. We categorise nodes into different classes dynamically and require at least one node per class per chain, the number of participants per chain (shard) is mostly reduced; more chains can be split out. This brings a significant improvement in terms of scalability.

References

[1] Satoshi Nakamoto et al. Bitcoin: A peer-to-peer electronic cash system. 2008.

[2] Gavin Wood et al. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper*, 151(2014):1–32, 2014.

[3] Joseph Poon and Thaddeus Dryja. The bitcoin lightning network: Scalable off-chain instant payments, 2016.

[4] Jacob Eberhardt and Stefan Tai. On or off the blockchain? insights on off-chaining computation and data. In *European Conference on Service-Oriented and Cloud Computing*, pages 3–15. Springer, 2017.

[5] A Pinar Ozisik, Gavin Andresen, George Bissias, Amir Houmansadr, and Brian Levine. Graphene: A new protocol for block propagation using set reconciliation. In *Data Privacy Management, Cryptocurrencies and Blockchain Technology*, pages 420–428. Springer, 2017.

[6] Peter Tschipper. Buip010: Xtreme thinblocks. In *Bitcoin Forum (1 January 2016). https://bitco. in/forum/threads/buip010-passed-xtreme-thinblocks*, volume 774, 2016.

[7] Matt Corallo. Bip 152: compact block relay. *See https://github.com/bitcoin/bips/blob/master/bip-0152.mediawiki*, 2016.

[8] Bitcoin, developer-guide. https://bitcoin.org/en/developer-guide#simplified-payment-verification-spv, 2019.

[9] Xinxin Fan and Qi Chai. Roll-dpos: a randomized delegated proof of stake scheme for scalable blockchain-based internet of things systems. In *Proceedings of the 15th EAI International Conference on Mobile and Ubiquitous Systems: Computing, Networking and Services*, pages 482–484, 2018.

[10] Yibin Xu. Section-blockchain: A storage reduced blockchain protocol, the foundation of an autotrophic decentralized storage architecture. In *2018 23rd International Conference on Engineering of Complex Computer Systems (ICECCS)*, pages 115–125. IEEE, 2018.

[11] Serguei Popov. The tangle, 2016.

[12] Gerard De Roode, Ikram Ullah, and Paul JM Havinga. How to break iota heart by replaying? In *2018 IEEE Globecom Workshops (GC Wkshps)*, pages 1–7. IEEE, 2018.

[13] Loi Luu, Viswesh Narayanan, Chaodong Zheng, Kunal Baweja, Seth Gilbert, and Prateek Saxena. A secure sharding protocol for open blockchains. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pages 17–30. ACM, 2016.

[14] Eleftherios Kokoris-Kogias, Philipp Jovanovic, Linus Gasser, Nicolas Gailly, Ewa Syta, and Bryan Ford. Omniledger: A secure, scale-out, decentralized ledger via sharding. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 583–598. IEEE, 2018.

[15] George Danezis and Sarah Meiklejohn. Centrally banked cryptocurrencies. *arXiv preprint arXiv:1505.06895*, 2015.

[16] Mahdi Zamani, Mahnush Movahedi, and Mariana Raykova. Rapidchain: Scaling blockchain via full sharding. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*, pages 931–948. ACM, 2018.

[17] Adam Back et al. Hashcash-a denial of service counter-measure. 2002.

[18] Aggelos Kiayias, Alexander Russell, Bernardo David, and Roman Oliynykov. Ouroboros: A provably secure proof-of-stake blockchain protocol. In *Annual International Cryptology Conference*, pages 357–388. Springer, 2017.

[19] Yibin Xu and Yangyu Huang. Mwpow: Multiple winners proof of work protocol, a decentralisation strengthened fast-confirm blockchain protocol. *Security and Communication Networks*, 2019, 2019.

[20] Yibin Xu and Yangyu Huang. Mwpow-multi-winner proof of work consensus protocol: an immediate block-confirm solution and an incentive for common devices to join blockchain. In *2018 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Ubiquitous Computing & Communications, Big Data & Cloud Computing, Social Computing & Networking, Sustainable Computing & Communications (ISPA/IUCC/BDCloud/SocialCom/SustainCom)*, pages 964–971. IEEE, 2018.

[21] Conrad Burchert, Christian Decker, and Roger Wattenhofer. Scalable funding of bitcoin micropayment channel networks. *Royal Society open science*, 5(8):180089, 2018.

[22] Christian Decker and Roger Wattenhofer. A fast and scalable payment network with bitcoin duplex micropayment channels. In *Symposium on Self-Stabilizing Systems*, pages 3–18. Springer, 2015.

[23] Matthew Green and Ian Miers. Bolt: Anonymous payment channels for decentralized currencies. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 473–489. ACM, 2017.

[24] Serguei Popov. The tangle. *cit. on*, page 131, 2016.

[25] Yibin Xu and Yangyu Huang. Segment blockchain: A size reduced storage mechanism for blockchain. *IEEE Access*, 2020.

[26] Yibin Xu, Yangyu Huang, and Jianhua Shao. Anchoring the value of cryptocurrency. *arXiv preprint arXiv:2001.08154, 3rd International Workshop on Emerging Trends in Software Engineering for Blockchain*, 2020.

[27] Miguel Castro and Barbara Liskov. Practical byzantine fault tolerance and proactive recovery. *ACM Transactions on Computer Systems (TOCS)*, 20(4):398–461, 2002.

[28] Yibin Xu and Yangyu Huang. An n/2 byzantine node tolerate blockchain sharding approach. In *Proceedings of the 35th Annual ACM Symposium on Applied Computing*, SAC '20, page 349352, New York, NY, USA, 2020. Association for Computing Machinery.

[29] Yibin Xu, Yangyu Huang, Jianhua Shao, and George Theodorakopoulos. A flexible n/2 adversary node resistant and halting recoverable blockchain sharding protocol. *arXiv preprint arXiv:2003.06990, Concurrency and Computation: Practice and Experience, DoI:10.1002/CPE.5773*, 2020.

[30] James K Mullin. A second look at bloom filters. *Communications of the ACM*, 26(8):570–571, 1983.

[31] Michael T Goodrich and Michael Mitzenmacher. Invertible bloom lookup tables. In *2011 49th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 792–799. IEEE, 2011.

[32] Yibin Xu and Yangyu Huang. Contract-connection:an efficient communication protocol for distributed ledger technology. *2019 IEEE 38th International Performance Computing and Communications Conference (IPCCC)*, Oct 2019.