A Decentralized Mining Pool for General PoX-Based Blockchains

Niffler Network

October 2021

Abstract. Blockchain mining involves attempts of solving a computationally-hard hash puzzle. Miners mine blocks by attempting to solve the puzzle and obtain revenues. To amortize the revenue, the demand for pooled mining has emerged since the growth of puzzle difficulty. In pooled mining protocols, miners solve the same puzzle but with a looser difficulty and have all their mined coins shared according to their solutions (known as shares). To date, most mining pools are controlled by centralized parties and the decentralizing principle of blockchains is undermined. To solve the issue, decentralized pooled mining schemes are proposed. Two major schemes are P2Pool and SmartPool. However, the throughput of share submission is limited or the protocol is not incentive-compatible due to expensive gas fees in the two designs. Moreover, their mining is restricted to one or a small set of existing blockchains. To provide a more generalized decentralized mining pool with a decent throughput and incentive compatibility, we propose our construction based on a newly designed blockchain.

The newly designed blockchain protocol can be executed in either a proof-of-stake paradigm or our newly proposed proof-of-mining paradigm, which delicately motivates greater participation in mining. The basic safety and liveness are guaranteed in both paradigms. Based on such a blockchain, we are allowed to qualify the mining power of all participants and revenue them in a decentralized manner in two ways. Namely, apart from sharing coins from the mined chain, they are also rewarded by coins of our newly spawned chain. Thereby, participants of centralized pools are motivated to switch to the decentralized protocol. The throughput of share submissions is only bounded by the throughput of the blockchain which is sufficiently large. In particular, we provide an aggregation of share submissions via non-interactive zero-knowledge proofs and hence the communication overhead can be further reduced.

It is not trivial to determine the address of revenue receipt for share mining and to allocate all mined revenues fairly. To tackle this, we maintain a dynamically rotating committee from miners in a game-theoretically fair and decentralized manner. The committee switches by the growth of our chain and reassembles for every certain block generations. For each committee, a revenue receipt address, manifesting as a public key, is negotiated according to a threshold signature scheme and is released to all participating miners. Every certain round, all mined revenues are allocated to miners in two steps. Namely, revenue releasing transactions are arranged by the block assemblers of our chain according to share submissions and then signed and aggregated by the committee.

Our description of the protocol is based on the proof-of-work mining of Bitcoin (before the Taproot update) with an ECDSA-based public key infrastructure. However, our methodology can be extended to support more than one chain simultaneously and each of them can be based on any proof-of-X mining with any public key infrastructure with an appropriate threshold signature scheme.

Keywords: Blockchain, Pooled Mining, Distributed Consensus, Proof-of-Work, Proof-of-Stake, Threshold Signature.

1 Introduction

Since the emergence of Bitcoin and its underlying consensus [30], known as the blockchain consensus, various novel distributed ledgers have been launched as cryptocurrencies. A blockchain is an append-only linearly ordered log of blocks and each block can be considered as a linearly ordered log of transactions. The blockchain as a whole realizes the functionality of an append-only ledger manifesting as a sequence of transactions. To extend the ledger, a new block is appended to the chain. The proposer of each new block is rewarded by *block rewards* and *transaction fees*. To compete for the chance of proposing the next block and hence earning the rewards, a mechanism should be brought about to elect the proposer of each block.

Most mainstream blockchains launched in the early stage solve the above issue based on *proof-of-work* (PoW), a technique firstly introduced in the 1980s to prevent e-mail spam. A PoW-based blockchain involves a hash puzzle that requires a substantial amount of hash attempts to solve. The puzzle is sufficiently difficult such that, on average, only one node solves the puzzle within every certain time interval, say, 10 minutes. This lucky node is entitled to the proposal right of the new block. The procedure of trying to solve the hash puzzle is known as *mining* and the mining nodes are thereby called *miners*.

However, due to the growing difficulty of the puzzles of existing PoW-based blockchains, it is too hard for most lightweight miners to solve a hash puzzle even within ten years. In another word, the variance of the mining revenue is too great to sustain. To amortize the revenue, *mining pools* emerge. Most mining pools are centralized nodes. Pools issue their *mining templates* to participating miners. The miners can be rewarded if they find out a "partial" solution (known as *shares*) to the puzzle. Therefore, lightweight miners have the chance to receive revenues with a significantly smaller variance.

However, due to their capability of determining the mining template, centralized mining pools are entitled to excessively great decision power in the three aspects.

- The mining template specifies the sequence of transactions to be included manifesting as a *Merkle tree*. Thereby, they are allowed to selectively accept transactions of their own favor, instead of having each actual miner make the decisions. This potentially allows pools to cooperatively make their own rules for transactions.
- Some blockchains include a field for community voting. This field is also determined by the mining template. This contradicts the decentralizing principle of "one hash one vote".
- Moreover, certain pools may even have the capability of launching a "51% attack" to a blockchain with certain strategies like *selfish mining* [17].

To resolve the issue, attempts are made to devise decentralized pooled mining protocols, like P2Pool and SmartPool. However, existing solutions are far from suc-

cessful due to their respective limitations. For example, P2Pool introduces a parallel blockchain that grows for each share. This limits the throughput of shares due to the scalability nature of blockchains. SmartPool leverages an Ethereum smart contract to receive shares. However, their throughput is also limited by the throughput of Ethereum and the high gas fee of Ethereum transactions. SmartPool introduces the methodology of submitting shares as bundles to solve the throughput issue, while each submitter still causes one Ethereum transaction to exert burdens on the smart contract virtual machine. As a mainstream layer-one cryptocurrency, Ethereum can only provide the contract with a small fraction of total storage and computation resource. This is not sufficient to realize a giant mining pool even with delicate optimizations.

Based on the above observations, we devise a novel pooled mining mechanism based on a newly launched blockchain. The functionality of pooled mining is realized by this blockchain while the miners can still participate in the mining without involving deep into the executions of the blockchain.

To distinguish our chain and the chain our participating miners are mining, we refer to our chain as the A-chain and the latter one the B-chain. We assume a single PoW-based and ECDSA-based B-chain throughout most paragraphs of the paper. However, in fact, our scheme can be extended to support more than one B-chain and general PoX scheme rather than PoW only.

1.1 Technical Roadmap

Our decentralized pooled mining protocol consists of the following components.

A group of miners. Similar to classical mining pools, the essential mining of any B-chain is performed by participating B-chain miners. We maintain a list of *admissible* B-chain miners that are allowed to mine in our pool, with the interval of hash attempts determined by our decentralized system. Lightweight miners can simply regard the system as a blackbox and only take care of block templates issued by the A-chain. Apart from additional A-chain coin rewards, the essential difference for lightweight miners is the fact that the block assembling phase is not finished by the pool, but by themselves or their trusted third parties.

A novel consensus. The consensus of the A-chain starts with PoS, then transformed to PoM (see details in Sec. 4). The miners of our A-chain are stakeholders[1] (to PoS) and B-chain miners (to PoM). Also, we support an intermediate state where both PoS and PoM are adopted simultaneously if required by the actual market scenario. Such an intermediate consensus is a hybrid consensus where the possibility of having each miner propose the next block is proportional to a weight determined by a certain hybrid function of its stake and mining power.

[1] Most of the stakeholders are, or once were, B-chain miners.

Simulating a mining pool. The ultimate purpose of A-chain is the decentralized simulation of centralized mining pools. The consensus asks A-chain block assemblers to execute the following subroutines in the form of specialized A-chain transactions.

1. To prevent DDOS and to maintain a list of admissible B-chain miners[2], each newly participating B-chain miner should finish a qualification of hash power before entering. The qualification phase consists of two steps recorded and responded in two separate blocks with the difference in block height Δ. The first phase is the submission of a qualification request, which is responded to by a random hash puzzle to solve in Δ time. The second phase is the submission of the solution and the response is the approval of pooled mining and the updating of the admissible miner list.
2. Each A-chain block specifies the admissible mining interval (the interval that the nonce can take from in B-chain mining) for each miner in the admissible mining list. This intuitively takes a large amount of block size to realize but in fact, can be implemented by storing only a random seed and specifying a pseudorandom function family.
3. The regular opening of B-chain rewards asks the assemblers of certain blocks to have a set of unsigned B-chain transactions recorded in the corresponding Merkle tree.

The committee and the DAO. We adopt a committee consisting of recent mining contributors to authorize the opening of B-chain rewards for all share submitters. In particular, we adopt a *decentralized autonomous organization* (DAO) to direct critical protocol updates. Most functionalities of the committee and the DAO have their soundness guaranteed by the security of the threshold ECDSA scheme. More implementation details are shown in Sec. 3.6 and Sec. 6.2.

Generalizing to multiple PoX-based chains. As to be shown in Sec. 5.1, our scheme can be generalized to support the decentralized pooled mining of more than one chain. Moreover, a more generalized version of our scheme can support the decentralized pooled mining of more PoX-based chains rather than only PoW-based ones.

Smart Contracts. We adopt an EVM-liked virtual machine infrastructure to support general smart contracts. Moreover, via a specialized opcode, we allow for contract-level access to certain oracles which returns the mining difficulty and the coin value of each supported chain.

1.2 Related Works

Blockchain consensus. Starting from the emergence of Bitcoin [30], recent research of consensus mechanism has evolved from a permissioned environment (see [9, 18, 21, 23]) to a permissionless environment. Bitcoin's PoW consensus mechanism effectively defends against *Sybil attacks* [14] and double-spending [30]. However, due to the huge hash power and hence the energy

[2] The list is not necessarily short, it can be append-only and continuously grow by time.

waste brought about by the PoW mechanism, multiple PoX solutions have emerged. *Proof-of-Stake* (PoS, like [4, 11, 20, 24]) elects leaders (who extend the ledger) from stakeholders according to their amounts of coins. *Proof-of-Space* (see [15, 29, 31, 34]) and its variants base the security on the storage resources contributed by participants. However, most PoX mechanisms are essentially similar to PoW in the sense that leader election is attached with ledger extension in the form of blocks. To more fundamentally improve the scalability, committee-based consensus schemes (see [25–27, 33, 40, 42]) are introduced to separate leader election from ledger extension and have one or few (via *sharding*) dynamically rotating committees verify and linearize transactions through a permissioned protocol like PBFT of [8, 9]. Meanwhile, alternative improvements on the scalability are proposed like novel blockchain structures [36]. Efforts have been made to cryptographically model and analyze the security of blockchains [19, 32].

Pooled mining. Since the emergence of mining pools, the drawbacks, and risks of centralized pooled mining have been recognized. As attempts to solve the issue, P2Pool [1] and SmartPool [28] are proposed as two decentralized mining pools.

Cryptography. Threshold ECDSA has been studied as a circuit-specific multiparty computation over the past years [7, 12]. Efforts are made to create a publicly verifiable source of randomness in blockchains [2, 3, 6, 10, 22]. One way is to adopt verifiable delay functions [5, 13, 16, 35, 41] based on which the randomness of a block originates from a hard-to-compute function of previous blocks.

2 Terminologies

We assume the B-chain adopts an ECDSA-based public-key infrastructure (PKI). The protocol execution is divided into *rounds*. Round i is defined as the interval between the generation of the generation between A_{i-1} and A_i. Note that, due to the existence of potential forks, different participants of our protocol may have different views on the current round number.

For most classical PoW-based consensus, the miner is also the block assembler. Thereby, the terminologies of a miner and a block assembler can be utilized interchangeably. However, to avoid ambiguity, we often regard miners of A-chain block assembler and miners of B-chain miners in our work. To facilitate descriptions, we assume that any proposed valid transaction of A-chain is included in the block of the next round.

$A := B$ assigns B to A. $A \xleftarrow{\$} B$ stands for selecting an element uniformly randomly from B if B is a countable set, or selecting an element randomly accordingly to B if it is a distribution, or having an outcome returned from B is it is a protocol with randomness. $\|$ concatenates two bit-strings or two tuples. ϵ stands for the empty string while $\emptyset$ stands for the empty set. $[m]$ stands for $\{1, 2, \ldots, m\}$ for any positive integer $m \in \mathbb{Z}^+$. For a tuple $\mathbf{A} = (a_0, a_1, \ldots, a_\ell)$, $\mathbf{A}[i : j]$ stands

for $(a_i, a_{i+1}, \ldots, a_j)$ if integers i, j satisfy $1 \leq i < j \leq \ell$. $\mathbf{A}[: -k]$ stands for $(a_{\ell-k+1}, a_{\ell-k+2}, \ldots, a_\ell)$ if the natural number k is no greater than ℓ. For two sequences A and B, $A \preccurlyeq B$ means A is a prefix of B.

We assume a security parameter κ. We assume a cryptographic hash function $H(\cdot)$ and a public key infrastructure (PKI). We assume a digital signature scheme $\mathscr{S} = (\mathsf{Gen}, \mathsf{Sig}, \mathsf{Vrf})$ with the key generation protocol $(\mathsf{sk}, \mathsf{pk}) \xleftarrow{\$} \mathsf{Gen}(1^\kappa)$, the signature protocol $\sigma \leftarrow \mathsf{Sig}(\mathsf{sk}, m)$ and a verification protocol $0/1 \leftarrow \mathsf{Vrf}(\mathsf{pk}, m, \sigma)$. As in most cryptocurrencies, the identity of any node is referred to as its public key. Therefore, the same notation of pk may refer to a node or its public key interchangeably. For a tuple $(n_1, n_2, \ldots, n_\ell)$, we denote $(n_1, n_2, \ldots, n_\ell)_{\mathsf{pk}}$ or $(n_1, n_2, \ldots, n_\ell, \mathsf{sig}_{\mathsf{pk}})$ as the same tuple appended with a proper signature from pk. We assume a pseudorandom function family $\mathsf{PRF}_k(\cdot)$ keyed by k with a sufficiently large range. $\mathsf{L}(T)$ takes a Merkle tree T as input and outputs the linearized log of transactions in T.

m stands for the total number of existing admissible B-chain miners. We denote an A-chain as a sequence

$$\mathsf{Achain} = (A_0, A_1, \ldots, A_\ell),$$

a B-chain as a sequence

$$\mathsf{Bchain} = (B_0, B_1, \ldots, B_\ell).$$

Let C be any A-chain or B-chain block, $\mathsf{miner}(C)$ stands for the miner of the block C. Our abstraction regards each block as $C = (C.\mathsf{header}, C.\mathsf{Merkle}, C.\mathsf{nonce})$, where $C.\mathsf{header} = (\mathtt{type}, \mathtt{height}, H(C_{-1}), aux)$, $C.\mathsf{Merkle}$ is the root of a Merkle tree consisting of the ledger (for both A-chain and B-chain blocks) and a lookup table T (for A-chain blocks). The block hash of C is defined as

$$H(C) := H(C.\mathsf{header} \| C.\mathsf{nonce}).$$

Block height is an indicator of the distance from a block to the genesis block. For instance, the height of the genesis block is 0, its succeeding block is of height 1. We say that "block A is within d blocks away from block B" if (1) A and B are on the same chain branch, (2) and the block heights of A and B differ no greater than d. In addition, for a block B, we notate its previous block as B_{-1} when there is no ambiguity.

Although the term "mining" often refers only to PoW participants, in our article, however, we refer to all block assemblers as miners to simplify descriptions. *Block hash* refers to the hash of the block header. In PoW, hash power refers to the number of hash attempts that one miner can take within a certain time interval. In this article, we generalize the notion of hash power to the number of hash attempts that one block assembler can take for each block assembling in PoS or PoM.

3 The Protocol

This section brings about the full protocol of our decentralized pooled mining. We at fist provide building blocks for the protocol, an overview of the protocol, and then detailed constructions.

3.1 Building Blocks

Threshold ECDSA. We adopt a t-out-of-n threshold ECDSA scheme to provide soundness and liveness of certain message deliveries under the assumption that a certain fraction of participants can be byzantine or crashed nodes. Threshold ECDSA can be regarded as a secure multiparty computation protocol realizing a specific functionality, i.e., producing a valid digital signature assuming the cooperation of over t participants. Also, we can regard a threshold ECDSA scheme as a specialized signature scheme without delving too deep into communication details within the protocol execution. A threshold ECDSA scheme involves the following interactive protocols.

- *Setup.* In the setup phase, all participants cooperate to generate their own secret key via a shared protocol $\Pi_{\text{sig-setup}}$, their own public key and also their aggregated key pairs $(\mathsf{SK}, \mathsf{PK})$.
- *Signing.* The signature of a message is produced by an interactive signing protocol $\Pi_{\text{sig-sign}}$. After the interactions, one final signature is aggregated as the output of the protocol. This signature is equivalent to a signature from the naive ECDSA signing produced by PK. The protocol can be considered as a blackbox in the latter descriptions.
- *Verification.* The signature of any message produced by the above signing protocol can be verified by the signature verification protocol of the naive ECDSA with the public key PK. For a message m and its signature σ, it is a valid aggregated signature if and only if $\mathsf{Vrf}(\mathsf{PK}, m, \sigma) = 1$, except for a negligible probability.

The most rigorous description of the threshold ECDSA functionality involves a rich background of cryptography, especially compositional security analysis techniques. Therefore, we omit this for simplicity.

Verifiable Delay Function. Verifiable delay functions (VDFs) are verifiable computations that are easy to verify but hard to compute. Namely, the output of the computation can be efficiently verified, however, the computation (or the evaluation) involves a substantial amount of sequential computations. Due to the nature of these sequential computations, the evaluation process cannot be speeded up by parallel or concurrent computations. Since the per-thread computation power of existing CPUs is limited, the evaluation essentially guarantees a certain length of time delay before the valid result of the computation is outputted by any node. Formally, a VDF puzzle is parameterized by a random number r which is unpredictable until the problem is released, we notate its correct result as $\mathsf{Eval}(r)$.

A Random Beacon. Verifiable delay functions are often leveraged to produce an unpredictable random beacon for blockchains. The generation of each block issues a verifiable delay function to be evaluated by all miners. We let the random seed of each block be the xor-summation of the two components, i.e., the previous block hash and the valid evaluation result of the verifiable delay function from the block d rounds earlier than the current block. The detailed protocols and the cryptographic reduction from the randomness of the beacon to the soundness of verifiable delay functions are omitted in our article. However, we are allowed to safely assume that the random beacon has realized the functionality that each block has a seed outputted by a random oracle taking the block as the input and outputs only by the generation of a block.

3.2 Protocol Overview

There are two blockchain objects to be considered in our system. The first one is the B-chain, i.e., the blockchain that our participating miners are mining. Note that our system can actually support multiple B-chains simultaneously. However, we only consider one B-chain with an ECDSA-based PKI for the sake of simplicity. The other chain is A-chain, the chain we launched to simulate a centralized mining pool in a decentralized approach. We assume that the block interval of the A-chain is relatively shorter than that of the B-chain. This makes sense since our consensus is not based on PoW and a relatively long block interval is the nature of PoW.

The protocol is described in two layers. The first layer is the consensus of our launched A-chain, i.e., how the proposer of each block is determined and how blocks are generated, this layer is described as a separate section in Sec. 4. Thereby, we refer to the existence of a sound consensus scheme (concerning leader elections and ledger extensions) as a blackbox in this section. The second layer is the description of how A-chain realizes the functionality of a decentralized mining pool when abstracting A-chain block assemblers as a consistent and ideal third party.

The protocol consists of the following parties within two communities (the A-chain community and the B-chain community). There are two parties in the A-chain community.

Block Assemblers. A-chain is a blockchain extending one block by one block assembler for each round. The detailed leader (block proposer) election and blockchain extension protocols are shown in Sec. 4. In this part, we are allowed to ideally regard all A-chain block proposers as one consistent party realizing our described functionalities of a decentralized mining pool. Such an ideal and consistent party executes the following subroutines.

1. *Qualification.* Each B-chain miner should finish a qualification test before enrolling in the system. From the perspective of A-chain block assemblers, the qualification manifests as two specialized transactions of two separate blocks of a certain predetermined distance (i.e., the difference in block heights). The first is a qualification request, the A-chain block assembler

should specify a mining task for each request, manifesting as an additional field appended to the transaction in the Merkle tree. The second is the qualification submission, it includes all its found shares within the predetermined number of A-chain block generation intervals. If the number of shares is greater than the threshold and all shares are valid with respect to the assigned task, the corresponding miner is enrolled into a list of admissible miners notated as mining_list.

2. *Range Division.* A naive approach is to allow miners to have hash attempts in any slot of the nonce range. However, this brings about two major issues. The first is the issue of solution eavesdropping. Nonce solutions are easy to eavesdrop since the submission channel is not secure in P2P networks. The second is the issue of pre-mining. Pre-mining refers to the adversary behavior that mines shares in secret for a long time and then releases all shares to gain a high chance of controlling over $1/3$ committee slots. An intuitional solution to pre-mining is to make shares mined after an old B-chain block invalid. However, due to the possibility of B-chain forks, it is infeasible to force miners to mine after one certain block or even a few blocks. Thereby, a long range of tolerable B-chain block height is a must while this brings about a great chance of adversary pre-mining within such a range. Our solution is to have A-chain determine an admissible range for each miner in mining_list. Since A-chain is not PoW-based, it has a faster rate of block generation and hence a fast rate of admissible range alternation. The alternation is not predictable due to the soundness of our random beacon. Miners can only try hash attempts within the range specified by the latest confirmed A-chain block. However, when realizing the protocol, we should allow for nonces within slots specified by multiple recent A-chain blocks due to the asynchronous network. This does not compromise our intention of range divisions since even the union of few admissible slots is significantly small compared with the whole nonce range.
3. *Share Collection and A-chain Rewarding.* Once a valid share is found by any miner, the share (including the B-chain block header) is submitted to the A-chain network. Block assembler should have the Merkle tree include a transaction releasing certain instant A-chain coins to the miner. Each A-chain block contains a field recording all submitters of shares recorded in the current Merkle tree.
4. *Timely Releasing and B-chain Rewarding.* Every W blocks, a set of B-chain transactions are generated by the corresponding A-chain block assembler to release B-chain revenue to share submitters. The set of transactions are not signed. The signing and proposing of these transactions are executed by the committee of B-chain miners.
5. *The A-chain Ledger.* Similar to most cryptocurrency blockchains, A-chain is a ledger of A-chain coins. Normal A-chain transactions are collected and sorted in each block.

The DAO. The DAO is a party of A-chain stakeholders for realizing critical protocol updates. We provide a brief description of the DAO in Sec. 6.2. However, for generality and tunability, we do not specify the detailed rule of DAO election and we do not formally describe the DAO protocols in the rest of the whitepaper. We advise most readers to omit all DAO-related descriptions to take a quick glance at our protocol and grasp our key innovations.

The B-chain community includes the following parties.

Non-Admissible Miners. To prevent DDOS and facilitate range divisions, B-chain miners cannot participate in our pooled mining without finishing a qualification test. The qualification test asks a miner to mine for certain shares within certain A-chain block generation intervals. After finishing the test, the miner is enrolled into $\mathsf{mining\_list}$.

Admissible Miners. Each A-chain block specifies an admissible nonce slot. Miners should try nonce solution within the nonce slot specified in the latest confirmed A-chain block. Once a valid share is found, the share (including the block header, not the whole Merkle tree of transactions) is submitted to the A-chain network for certain instant A-chain rewards and a certain share of B-chain revenues to be released later. If the share meets the target of proposing a B-chain block, the whole block is also proposed to the B-chain network. Different from traditional centralized mining, the miners have the right to assemble the Merkle tree of transactions.

It is possible that a valid share is eavesdropped on and copied by adversary nodes in the P2P network. To avoid this, we ask for all miners to include a specialized transaction with its public key as the only receiving address in the right-most node of the Merkle tree.

The Committee. The committee is constituted randomly from recent miners according to their submitted shares. The purpose of the committee is to jointly generate a public key PK and each obtains a share of the corresponding secret key SK. Thereby, as the mining template specifies the address of PK, the committee can jointly compute signatures for revenue releasing transactions of the B-chain. This is provided by a threshold ECDSA scheme.

3.3 The Aggregated Address

We maintain a dynamically rotating committee of miners. The committee of round R is notated as $\mathsf{com}_R = (\mathsf{pk}_1, \mathsf{pk}_2, \ldots, \mathsf{pk}_\ell)$. The committee rotates every Q rounds by enrolling miners from share miners in the recent rounds. The current committee is recorded in the block header. The address to be included in all Q rounds with the same committee share the same *mining address*. The mining address is the address that all B-chain miners should take as the only output of block rewards and transaction fees. Such an address is formed as an aggregated address of all ℓ committee members and can be opened up by any $2/3$ fraction of ℓ members. To this end, we adopt a $\frac{2}{3}\ell$-out-of-ℓ threshold ECDSA scheme.

3.4 Zero-Knowledge Proofs

To decrease the variance of mining, a significant number of shares are probably mined within every second. For example, if the difficulty of mining a share is 10^6 times easier than the difficulty of mining a block and a block is mined every ten minutes on average, $10^6/600 \approx 1666$ shares are globally mined every second. Thereby, great communication overhead is caused by share submissions of miners. This is a great burden to the system since the throughputs of most existing permissionless distributed consensus are limited to a few thousand or less. To reduce the communication overhead caused by share submissions, We leverage non-interactive zero-knowledge (NIZK) proofs to "aggregate" share submissions.

In general, zero-knowledge proofs are techniques for proving that the prover holds a witness satisfying a certain formula without leaking the witness. A zero-knowledge proof scheme is non-interactive if it does not require the prover to interact with the verifier to obtain challenges.

$$\mathsf{PoK}\{w : P(w) = 0\}$$

is the notation for a NIZK proof that the prover holds a witness w such that the predicate $P(w) = 0$ is satisfied. For instance, if $P(w) := g^w - r$, this is a NIZK proof for the fact that the prover holds a logarithm of r with the base g. There are various different NIZK proof schemes (with a proof size sublinear to its witness size) applicable to our system. However, in this article, we do not specify any one of them for generality. Since the formal definition is far from trivial, we only describe the verification phase and the security of a NIZK proof verbally.

For a miner to prove that it holds a list $(x_1, x_2, \ldots, x_K)$ of shares to the puzzle $H(B.\mathsf{header}\|x) < D_s$ with $\hat{x} \le x < \hat{x} + \Gamma$, it generates

$$\mathsf{pf} = \mathsf{PoK}\left\{ x_1, x_2, \ldots, x_K \ : \begin{array}{l} H(B.\mathsf{header}\|x_1) < D_s \ \wedge \ \hat{x} \le x_1 < x_2 \ \wedge \\ H(B.\mathsf{header}\|x_2) < D_s \ \wedge \ \hat{x} \le x_2 < x_3 \ \wedge \\ \ldots \\ H(B.\mathsf{header}\|x_K) < D_s \ \wedge \ \hat{x} \le x_K < \hat{x} + \Gamma \end{array} \right\},$$

which is sublinear or constant if an applicable NIZK proof scheme is leveraged.

3.5 Block Assembler Perspective

The assembling of an A-chain block involves the regular leader election (see Sec. 4) and ledger extension of A-chain transaction, and specialized transactions for the following functionalities. The full protocol of A-chain block assemblers is shown in Fig. 1.

Qualification Phase. To join our pool with total mining resource R of type τ, a miner submits a specialized transaction[3] to start up a qualification phase. This phase qualifies whether the miner has its claimed hash power.

[3] Considering incentive compatibility issues, this transaction may require a substantial amount of fees.

For a miner of identity pk claiming a moderate hash rate, its qualification phase consists of $\Delta + 2$ rounds to be described below. Only after the qualification, it is enrolled into the mining list manifesting as pk added to mining_list.

Submission. Suppose that the qualification phase starts from round R, a qualification request is sent to the chain as a specialized transaction

$$(\texttt{qualification\_request}, \mathsf{pk})_{\mathsf{pk}}.$$

The transaction is collected by a block assembler of A-chain afterwards. To facilitate descriptions, we ideally regard that this transaction is included in the block of the next round.

Mining. The block assembler adds the specialized transaction into the block by assigning a mining task of difficulty D_s. The task consists of a pair (s, t) where $s \xleftarrow{\$} [2^\kappa]$ and $t = s + \Omega'$ for a constant Ω'. After receiving the task, the miner attempts to mine a B-chain block B, i.e., to solve the hash puzzle

$$H(B.\mathsf{header} \| B.\mathsf{nonce}) < D_s,$$

where B.nonce is enumerated starting from s. All valid solutions are recorded in

$$\mathsf{rec} = \begin{pmatrix} B.\mathsf{header}_1 & B.\mathsf{nonce}_1 \\ B.\mathsf{header}_2 & B.\mathsf{nonce}_2 \\ \cdots & \cdots \end{pmatrix}.$$

Verification. In round $R+\Delta$, all valid solutions are submitted as another specialized transaction

$$(\texttt{qualification\_answer}, R, \mathsf{pk}, (s, t), \mathsf{rec}).$$

Each row of rec is verified by the block assembler of round $R + \Delta + 1$, and the miner is enrolled into the mining list if and only if the number of total solutions in rec is greater than a certain threshold.

Mining Task Assignment. For each round R, the block assembler of the round assigns mining tasks in the following approach.

Range Division. The range of all nonces Ω is divided into $M := |\mathsf{mining\_list}|$ segments, we denote them as $(s_1, t_1) = (0, \frac{\Omega}{M}), (s_2, t_2) = (\frac{\Omega}{M} + 1, \frac{2\Omega}{M}), \ldots, (s_M, t_M) = (\frac{(M-1)\Omega}{M} + 1, \frac{M\Omega}{M})$.

Task Assignment. The mining task of round R is determined as M tuples included as a part of the Merkle tree

$$(R, \mathsf{pk}_i, (s_i, t_i))$$

Note that since the total range of Ω is sufficiently large, we are allowed to assume that no miner could possibly enumerate all elements within (s_i, t_i) for any $i \in [M]$. The reason for asking each miner to try nonces within a predetermined range is to avoid cheating by overlapping.

In fact, the actual implementation of the mining task assignment can be significantly simplified. With the randomness seed r_R of the round R, s_i, t_i can be locally computed by each admissible miner as

$$(s_i, t_i) = \left(\frac{(k-1)\Omega}{M} + 1, \frac{k\Omega}{M}\right),$$

where

$$k = \mathsf{PRF}_{r_R}(i) \mod M.$$

Do we need to worry about the magnitude of M? Intuitively, a great magnitude of M casts a great burden on the system. However, such an intuition is false for two reasons.

- The list of all admissible miners is maintained by a persistent data structure rather than having it stored by every block. This resembles the balances in account-based ledgers where each block only needs to store the updates via a persistent structure. In particular, we do not need to record any task assignment as a tuple of size M.
- The range Ω is sufficiently large, say, $\Omega = 2^{256}$. Even if $M = 2^{32}$, the nonce range $\frac{\Omega}{M} = 2^{224}$ for each participant is still sufficiently large. Therefore, we do not need to worry about too small a nonce range caused by a great M.

Share Collection and A-Chain Rewarding. After receiving the mining task, each miner pk_i tries to solve the hash puzzle by enumerating nonces starting from s_i. Different from centralized mining pools, we allow miners to assemble the block by creating its Merkle tree of transactions $B.\mathsf{Merkle}$. The block header $B.\mathsf{header}$ should be assembled such that the rewarding address is the aggregated address mentioned in Sec.3.3. After finding a solution that

$$H(B.\mathsf{header}||B.\mathsf{nonce}) < D_s,$$

- the block B is sent to the P2P network of B-chain if $H(B) < D$,
- the tuple of $(\texttt{share\_submission}, R, \mathsf{pk}_i, B.\mathsf{header}, B.\mathsf{nonce})_{\mathsf{pk}_i}$ is sent to the network of A-chain,
- and the nonce value $B.\mathsf{nonce}$ is in the range of its permitted interval of blocks on the chain branch within d blocks away.

The assembler of the next A-chain block adds the tuple into the Merkle root of its assembled A-chain block. Certain A-chain coins are minted instantly to the miner.

Also, a tuple of nonce solutions $(x_1, x_2, \ldots, x_K)$ for the same header can be submitted simultaneously with a sublinear or constant size by NIZK proofs in the form of

$$\mathsf{pf} = \mathsf{PoK}\left\{ x_1, x_2, \ldots, x_K \ : \begin{array}{l} H(B.\mathsf{header}||x_1) < D_s \ \wedge \ s_i \le x_1 < x_2 \ \wedge \\ H(B.\mathsf{header}||x_2) < D_s \ \wedge \ x_2 < x_3 \ \wedge \\ \ldots \\ H(B.\mathsf{header}||x_K) < D_s \ \wedge \ x_K < t_i \end{array} \right\}.$$

Thereby, the submission is $(\texttt{share\_submission\_agg}, R, \mathsf{pk}_i, B.\mathsf{header}, K, \mathsf{pf}, s_i, t_i)_{\mathsf{pk}_i}$.

Timely Releasing and B-Chain Rewarding. Mining a valid nonce share brings about both A-chain rewards and B-chain rewards. This asks the A-chain block assembler to pack up a set of B-chain transactions (we refer to them as *open-up transactions*) to allocate the B-chain revenues to corresponding A-chain contributors according to our rule. These transactions are not signed, they are valid only with the proper aggregated signature from the committee.

3.6 Committee Perspective

The committee is a group of B-chain miners participating in our protocol which rotates every Q rounds. The execution of their routine consists of the following components.

Committee Subroutine. The committee has an aggregated address PK. For each A-chain block of height $kW - d$ with a natural number $k \in \mathbb{N}^+$, the committee observes its $\mathcal{L}_k$. $\mathcal{L}_k$ is parsed to a tuple $(\mathsf{tx}_1, \mathsf{tx}_2, \ldots, \mathsf{tx}_\ell)$. For each tx_i with $i \in [\ell]$, all committee members cooperate to compute the signature sig_i for it. Note that for each $i \in [\ell]$, sig_i is equivalent to a signature from PK on the message tx_i. Thereby, broadcasting $\widetilde{\mathsf{tx}}_i = (\mathsf{tx}_i, \mathsf{sig}_i)$ to the B-chain network releases B-chain coins to all corresponding miners with shares recorded in A-chain blocks from $A_{(k-1)W-d+1}$ to A_{kW-d}. The above broadcasting is performed by the first committee members in the lexicographic order.

Committee Switchover. The committee switches for each Q A-chain block generations. Namely, the switchover happens on the confirmation of each A-chain block of height kQ, where $k \in \mathbb{N}^+$ is a natural number. All committee slots are randomly selected from all share submitters within A-chain block height of $(k-1)Q+1$ to kQ (we refer to these shares as *round-k shares*). Specifically, $|\mathsf{com}_{k+1}|$ *lucky shares* are uniformly randomly chosen from the list of all round-k shares, then the committee members are set as the submitters of the lucky shares. The randomness comes from our publicly verifiable random source of block A_{kQ}. Note that it is possible that one identity takes up two committee slots, this is tolerable and is a must to guarantee the basic fairness of our scheme.

Importantly, to guarantee that all our mined B-chain revenues with the current PK are shared before the termination of the round, we ask Q to be a multiple of W. After the switchover, the new committee instantly negotiates the new PK according to and issue it to the A-chain network.

The Mining Template. The mining template specifies the address receiving the block rewards. Therefore, after the negotiation of the new PK, the committee assembles and broadcasts the newest template with the newest receiving address of PK. Afterwards, all miners of our scheme mine with the newest template. Due to the possibility of network delay, miners should stop mining for a certain period of

time before and after the committee switchover, or the risk of mining a invalid share should be taken.

3.7 B-Chain Miner Perspective

B-chain miners participate in the system by the following subroutines.

The qualification. To become an admissible miner of the system, the miner should finalize the two-step qualification phase described in Sec. 3.5.

The mining. After the qualification, the miner is allowed to mine on B-chain shares. It can regard the system as the classical pooled mining with two major differences. Firstly, the miner does not receive an assembled Merkle tree of transactions from the pool. The assembling is performed by the miner itself or its trusted third parties. Secondly, the interval of the nonce is limited to an interval that varies every A-chain block generation.

PoM consensus. B-chain miners who submitted a substantial amount of valid shares can participate in the A-chain consensus after the consensus is switched to the PoM mode (see Sec. 4).

4 The Underlying Consensus of A-Chain

The above descriptions of our scheme focus on the high-level technique of making pooled mining decentralized by assuming an existing A-chain realizing certain functionalities. The consensus of the A-chain itself, however, is not further discussed. In fact, the consensus of A-chain is not trivial. In this section, we describe the underlying consensus of the A-chain.

A-chain starts with a PoS, while adopts a novel consensus after a start-up phase. In the early phase, we adopt a *proof-of-stake* (PoS) consensus to finish the initial allocation of A-chain tokens. Afterward, a novel consensus is leveraged to replace the initial consensus. We name the novel consensus as *proof-of-mining* (PoM).

4.1 Proof of Stake

In the early stage, we adopt a proof-of-stake consensus to finalize the initial setup of the A-chain protocol before having a rich number of participants and hash power.

The genesis block. The genesis block includes the initial allocation.

The mining. For each round, assuming the hash of the previous block header as h, the hash puzzle is

$$H(h\|\mathsf{pk}\|\mathsf{nc}\|\mathsf{rd}) < \mathsf{target}_{PoS},$$

where pk is the public key of the A-chain miner, nc is a natural number no greater than the A-chain coins held by the miner. rd is a natural number to be enumerated from 0. When multiple valid blocks with different rd's are proposed, only the block with the smallest rd is accepted as the next block. If multiple blocks with the same rd occur, a fork and a chain competition happen.

A key difference from classical blockchains. One key difference lies in the structure of the hash puzzle. In our puzzle, we take only the hash of the previous block header (including the previous Merkle root) and the public key as the parameter. However, in classical blockchains, the block header, including the current Merkle root, is taken as the parameter as a whole. This essentially delays the confirmation of data integrity to a later block. In Sec. 7.3, we show that the system is still secure in an ad-hoc approach.

Forks. In the case of a chain fork, an honest A-chain miner only accepts the chain branch with the longest newest block height and (secondly) the least *chain weight*, which is defined as

$$\mathsf{weight}(A_0, A_1, \ldots, A_\ell) = \sum_{i=0}^{\ell} A_i.\mathsf{rd}.$$

The switchover. The switchover from PoS to PoM starts from a command from the DAO in the form of

$$(\texttt{switch\_PoS\_to\_PoM}, k, \mathsf{sig}_{DAO})$$

where k specifies the block height from which all blocks should be generated by the PoM consensus, sig_{DAO} is the aggregated signature of the DAO.

4.2 Proof of Mining

When the latest block reaches a predetermined block height, the blockchain is switched to the second phase, namely, the PoM phase. As mentioned in Sec. A.1, all known permissionless consensus schemes can be considered as one kind of resource proof. On a philosophical level, this is a must for a distributed consensus allowing dynamical entering and quitting of untrusted public nodes. This argument applied to PoM as well. In our proposed PoM consensus, a node can be elected as a block proposer with the probability proportional to its involved B-chain hash power.

Specifically, in our account-based leger, the total amount of devoted hash power within a certain interval of blocks is recorded for each participating B-chain miner. The generation of each block brings about a random seed, which determines a list of miners who can propose the next block. Each miner of the list is allowed to propose the next block or fork the block in case of malicious behaviors of a block proposer. The reward of the next block is shared by this list of miners evenly.

The PoM protocol. The detailed description of PoM unfolds below.

The mining. For each round, assuming the hash of the previous block header as h, the hash puzzle is

$$H(h\|\mathsf{pk}\|\mathsf{nc}\|\mathsf{rd}) < \mathsf{target}_{PoM},$$

where pk is the public key of the A-chain miner, nc is a natural number no greater than a predetermined factor times the number of its mined shares within the previous W blocks. rd is a natural number to be enumerated from 0. When multiple valid blocks with different rd's are proposed, only the block with the smallest rd is accepted as the next block. If multiple blocks with the same rd occur, a fork happens to be solved by a chain competition.
Note that, similar to that of PoS, our hash puzzle is parameterized by the previous block hash, rather than the hash of the current block header. This is critical to the security of our system.

Forks. In the case of a chain fork, an honest A-chain miner only accepts the chain branch with the longest newest block height and (secondly) the least *chain weight*, which is defined as

$$\mathsf{weight}(A_0, A_1, \ldots, A_\ell) = \sum_{i=0}^{\ell} A_i.\mathsf{rd}.$$

Interestingly, we observe the PoM consensus has a similar description with the PoS one.

An interesting observation. Note that, philosophically, PoM is PoW for the following reason. At the first glance, PoM gracefully has the same mining resource securing up two different blockchains, namely, the explicitly mined B-chain and the implicitly mined A-chain. However, this is not simple multiplexing.

Consider the following problem: is it possible that two blockchains, say, two PoW-based blockchains, share the same mining resource? This is theoretically possible if two blockchains share the same hash function and one specially designed mining template. However, this does not imply that one hash power is securing up two chains. In a game theoretical level, considering the hash power securing up one chain as h_A, the hash power securing up the other chain as h_B, the actual total hash power required to reach the same security of both chains is $h_A + h_B$ rather than $\max\{h_A, h_B\}$. Thereby, such multiplexing is not actually saving mining power, or the security of both chains is undermined.

For the same reason, our PoM is not actually saving mining power, but bringing about a greater mining power to B-chains.

Why a PoS-based early phase is required? In fact, the security of our PoM-based consensus can hardly be guaranteed in the early stage when the number of participants (B-chain miners) and the total hash power are limited. Even after entering the PoM phase, PoS is possibly adopted again. During the execution of the whole protocol, when the total hash power is lowered than a certain predetermined level, a command from the DAO will be issued to switch the consensus back to PoS.

4.3 An Intermediate Consensus

We provide the DAO with an option called the *intermediate consensus*. This is a hybrid consensus of PoS and PoM. This consensus differs from PoM by its hash

puzzle

$$H(h||\mathsf{pk}||\mathsf{nc}||\mathsf{rd}) < \mathsf{target}_{PoM},$$

where nc is enumerated in a range according to its submitted shares within certain previous blocks. In the intermediate consensus, however, we have such a range determined by a function $G(s, w)$ where s is its stake and w is its contributed shares.

The switchover from any consensus to an intermediate consensus starts with an instruction from the DAO in the form of

$$(\texttt{switch\_to\_hybrid}, k, G, \mathsf{sig}_{DAO}),$$

where k is the specified block height to start with the protocol update, G is the hybrid function, and sig_{DAO} is the valid aggregated signature.

5 Extension to Multiple B-Chains and Smart Contracts

5.1 Supporting Multiple B-Chains

In fact, our scheme can be extended to support multiple B-chains. In the extended scheme, all miners of different B-chains are allowed to participate in A-chain like the B-chain miners of our naive scheme except for the following difference.

Different mining difficulties. We assume that the difficulty of finding a share for each B-chain is the same constant fraction of that of finding a nonce solution to mine a block. When multiple B-chains are supported, it is no longer trivial to evaluate the computational resource required to find a share in different B-chain networks. To guarantee fairness, participants in our system (specifically, A-chain block assemblers) should have access to the same oracle for mining difficulty queries.

Different B-chain coin values. Likewise, a publicly accessible oracle for querying B-chain coin values is required to guarantee the fairness in A-chain rewarding for each share submission and fairly weighting the contribution of each participant during committee and DAO elections.

Different public-key infrastructures. Importantly, for B-chains not adopting the ECDSA-based public-key infrastructure, different threshold signature schemes have to be leveraged accordingly.

5.2 The Exchange Oracle

Due to the issues described in Sec. 5.1, in our A-chain network, we realize an exchange oracle to provide public access to different mining difficulties and coin values of different B-chains. Two methods can be adopted.

– The first way is to realize the oracle purely by the consensus of the A-chain. Assemblers of each A-chain block have the obligation to maintain a field describing the latest mining difficulties and coin values of each B-chain. Otherwise, honest block assemblers would not assemble blocks after their block and a chain fork would expel the block from the main chain.

– The second way, which is safer theoretically but harder to implement, is to solve the issue by truthful auctions. For every certain round, we provide a one-time sealed-bid truthful auction on certain B-chain shares and coin futures. The truthfulness of the auction scheme would bring the system insights into the actual value of each B-chain coin. This methodology originates from [38].

5.3 Smart Contracts

Our A-chain provides a smart contract language similar to *Solidity*. Smart contracts applicable to Ethereum can be directly deployed and executed in our virtual machine. In particular, via a specialized opcode, our smart contract virtual machine allows access to our exchange oracle at the contract level.

6 Token Allocation and The DAO

6.1 Token Allocation

The allocation of all mined B-chain tokens resembles that of classical mining pools and is thereby omitted. All A-chain tokens are minted and allocated in the following venues.

1. *Initial Allocations.* A certain portion of coins are initially offered to a list of early contributors. This offering is realized by specialized transactions in the genesis block.
2. *Block Assemblers.* A certain portion of coins are offered to miners or block assemblers of A-chain blocks. They are PoS participants in the early stage and PoM miners afterward. This portion can be regarded as block rewards of most existing blockchains.
3. *Share Submitters.* For participating B-chain miners, they can receive not only B-chain rewards according to the number of their submitted valid shares but also A-chain coins offered for each B-chain share submission.
4. *Committee members.* Committee members are rewarded by newly minted coins after the termination of each Q A-chain block generations.
5. *DAO members.* Likewise, the DAO members are offered a certain fraction of coins. Detailed protocols are realizing the DAO reward are omitted to simplify our discussions.

Note that the above allocations only have specified the ways A-chain coins are minted. Participating revenues not bringing about coin minting are not comprehended in the above discussion, say, transaction fees and share submission fees.

6.2 The DAO

We maintain a *decentralized autonomous organization* (DAO) to have a group of stake holders issue instructions on protocol updating and forks from certain chain branches.

The DAO includes a relatively static set of stake holders with an aggregated address of PK_{DAO}. The protocol updating instruction from the DAO is in the form of

$$(\texttt{instruction}, k, \mathsf{sig}_{DAO}),$$

where `instruction` specifies the instruction of protocol updating, k is the A-chain block height from which the protocol is updated, sig_{DAO} is the signature of PK_{DAO} on the instruction concatenating the specified block height.

The maintaining of DAO members. For a DAO non-member, to enter the DAO, it should collect approvals from more than $1/3$ DAO members. The approval is in the form of a signature and can be the aggregation of these signature forms the witness. The committee non-member can enter the DAO with such a witness and start the DAO subroutine when the witness is confirmed by the A-chain. Likewise, a DAO member can be evicted with approvals from more than $1/3$ members.

More implementation details. When implementing the protocol, more non-trivial details should be considered. The most significant one is the storage of the DAO list. There are several potential ways of storing the list, like storing the list as a data structure of A-chain blocks, as part of a smart contract in another chain, and as the data which should be locally maintained by all network nodes. The first solution seemingly takes up a large segment of storage for each A-chain block. The second solution has the security of our system depend on other chains. Also, the third solution pushes excessive burden and reliance on lightweight network nodes. To solve the issue, we leverage a persistent data structure to store the list on A-chain blocks, but with no much storage cost.

Note that to enable instructions from the DAO, when realizing the protocol, all DAO-related messages should be listened from the A-chain P2P network. DAO instructions are effective only when the majority of A-chain power (stake for the beginning, and hash power afterward) is held by participants receiving the instructions before their specified block height.

7 Incentive Compatibility and Security Analysis

In this section, we analyze the incentive compatibility and the security of our scheme in an ad-hoc approach.

7.1 Incentive Compatibility

The incentive compatibility of our system is discussed from three aspects, i.e., the perspective of A-chain block assemblers, B-chain miners, and committee members.

A-Chain Block Assemblers. The A-chain block assemblers should have the motivation to honestly finish their task of block assembling, including the classical block assembling routine, the qualification of new participants, and the maintaining of other data structures related to the scheme. An incentive-compatible system should provide motivations for their honest behaviors and discourage their potential misbehaving.

B-Chain Miners. The motivation of B-chain miners is relatively trivial. Miners have the incentive to participate in qualifications and mining. All their submissions for the qualification or share mining should be in the correct form or their efforts are in vain.

Committee Members. Committee members have the incentive to negotiate the new aggregated public key PK according to the threshold ECDSA scheme, the incentive to sign on all valid B-chain rewarding transactions for share submitters, and the incentive to submit all signed B-chain opening transactions to the B-chain network and to broadcast the newest mining template.

- The setup phase of our adopted threshold ECDSA scheme asks all participants to finish certain local computations and few rounds of broadcasting. The computation and communication cost is relatively light compared with rewards as committee members.
- Most committee members are miners and are benefited by B-chain reward openings. Therefore, they have the incentive to sign on B-chain opening transactions.
- The cost of broadcasting the newest mining template is light. The cost of submitting all B-chain opening transactions to the B-chain network is also small. This contradicts the intuition that submitting the transaction takes certain transaction fees. In fact, all transaction fees are deducted from the balance of PK rather than the submitter. These opening transactions are put into effect in the B-chain network as any committee member has submitted the list of opening transactions.

7.2 Security Definitions

In the context of the related literature, we analyze the security of a blockchain (in fact, applicable to any distributed consensus) in the following way. The security consists of two major properties, i.e., *consistency* and *liveness*. Consistency consists of *common prefix* and *self-consistency*. Common prefix means that for any two honest nodes of the system, their view of the blockchain main branch should be identical, after truncating a certain number of new blocks in the rear of the chain. Self-consistency means that for each honest node participating in the consensus, its view of the blockchain main branch in time t should be a prefix of that of t' if $t < t'$, except for a few new blocks in the rear. Liveness means that any newly

proposed transaction[4] is always comprehended into the ledger log after a certain time bounded by a polynomial of the security parameter and the network latency.

Formally, we denote the view on an A-chain main branch $\mathsf{Achain} = (A_0, A_1, \ldots, A_\ell)$ of a node p as $\mathsf{view}_A(p) = (A'_0, A'_1, \ldots, A'_\ell)$, that of a B-chain main branch as $\mathsf{view}_B(p) = (B'_0, B'_1, \ldots, B'_\ell)$. We occasionally specify the time of the view by explicitly showing the time t in $\mathsf{view}^t_A(p)$ or $\mathsf{view}^t_B(p)$. For both A-chain and B-chain, say, A-chain, the two consistency properties are thereby formally defined as follows.

Common prefix. For any two participants p and q of the system, for the same time t, let $\mathrm{LOG}_p = \mathsf{view}^t_A(p)[: -\delta]$ and $\mathrm{LOG}_q = \mathsf{view}^t_A(q)[: -\delta]$, either $\mathrm{LOG}_p \preccurlyeq \mathrm{LOG}_q$ or $\mathrm{LOG}_q \preccurlyeq \mathrm{LOG}_p$ holds. Here δ is a polynomial of the security parameter and the network latency bound.

Self-consistency. For the same participant p of time t and t', if $t < t'$, let $\mathrm{LOG}^t = \mathsf{view}^t_A(p)[: -\delta]$ and $\mathrm{LOG}^{t'} = \mathsf{view}^{t'}_A(p)[: -\delta]$, $\mathrm{LOG}^t \preccurlyeq \mathrm{LOG}^{t'}$ must hold for a δ which is a polynomial of the security paramter and the network delay.

Note that our definition of self-consistency is slightly different from that of related literature due to practical reasons. Some related works define the same property without having the rear of the chain truncated. This definition is applicable to their theoretical models but hard to be explained in our scenario.

7.3 Security Analysis

Block withholding and selfish mining. Block withholding and selfish mining [17] undermine the security of most PoW-based blockchains. On a philosophical level, the underlying principle of block withholding and selfish mining is actually to cheat honest miners to spend time mining on blocks that would not remain on the main chain after the proposal of certain adversary blocks. Therefore, selfish mining enables the adversary of α total hash rate to emulate an honest miner with at most $\frac{\alpha}{1-\alpha}$ total hash rate. However, selfish mining is not applicable to PoS and PoM since the time cost is negligible to miners.

Branch Decision Tree. Before the formal analysis, we define the branch decision tree to describe the probability space of mining an own branch after a certain block. The branch decision tree is a tree of infinite depth. Each of its nodes is a pair in the form of (d, w, h) where d is its depth in the tree, w stands for a block weight, and h stands for a block hash. The branch decision tree $\mathsf{tree}_{h,T}$ of a miner which is mining after a block of hash h and is allowed to have T hash attempts for each block is assembled recursively as follows.

– $\mathsf{tree}_{h,T}$ has a root of $(0, 0, h)$.

[4] Shares, qualification requests and qualification submissions are generalized transactions to A-chain.

– For each node (d, w, h) and each $i \in \mathbb{N}$,
  – for each $\mathsf{nc} \in [T]$, if $H(h\|\_\|\mathsf{nc}\|i) < \frac{\Omega}{D_0}$, the node has a child node of $(d+1, i, H(h\|\_\|\mathsf{nc}\|i))$. Note that it is possible that there are multiple children of the same weight.

Next, we define the least weight summation of depth d as

$$LS\left(\mathsf{tree}_{h,T}, d\right) = \min\left\{\left(\left(d_{i-1}, w_{i-1}, h_{i-1}\right), \left(d_i, w_i, h_i\right)\right) \in \mathsf{tree}_{h,T} : \sum_{i=1}^{d} w_i\right\}.$$

For instance, the branch with the least weight summation of the branch decision tree of Fig. 4 is the branch of $((0,0,h),(1,2,h_1),(2,1,h_8))$, rather than the left-most branch ending with $(2,3,h_5)$.

Secure Convergence Gap of PoM. In this paragraph, we consider the naive "51%" attack without block withholding and selfish mining. We show that our system is secure against the naive "51%" attack if a secure convergence gap is reached before confirming each block, based on the methodology from [37, 39]. Secure convergence gap[5], parameterized by the security parameter κ, is the least difference of two chain weights such that, if the two chains have the same length, the adversary with an α ratio of global hash rate can not win the following competition with any integer $L > 0$ except for a negligible probability (say, $\mathsf{negl}(\kappa) = 2^{-\kappa}$).

– The randomness of the game originates from the previous block hash h. We assume a PoM blockchain $\mathbf{A}_0 = (A_0, A_1, \ldots, A_h)$ and ideally assume that the adversary always takes a α fraction of total share submission workload for each block generation. We notate the global hash attempts for each rd number to be D and the difficulty is set to be $\frac{\Omega}{D_0}$. Suppose the adversary attempts to create a fork from a certain block of height h, it attempts to extend the chain to $\widetilde{\mathbf{A}} = \mathbf{A}_0\|\widetilde{\mathbf{A}}[h+1, h+\ell]$ when the honest miners are extending the chain to $\mathbf{A} = \mathbf{A}_0\|\mathbf{A}[h+1, h+\ell]$.
– The adversary takes $C = \left((i, w_i, h_i)_{i=1}^{L}\right)$ as the branch of $\mathsf{tree}_{h,T}$ with the least branch weight summation of $LS\left(\mathsf{tree}_{h,T}, L\right)$. Let $\widetilde{\mathbf{A}}[h+i]$ be a block leaving the Merkle root empty, with $\mathsf{rd} = w_i$, and the nonce such that the block hash is h_i.
– The adversary wins if $\mathsf{weight}(\widetilde{\mathbf{A}}) < \mathsf{weight}(\mathbf{A})$.

We notate the Win_L as the indicator of the event of having the adversary win the game.

[5] With a certain loss of rigorousness, the notion of a secure convergence gap resembles the notion of the "six block confirmation" rule in Bitcoin, if we consider the weight of a chain branch to be (the opposite of) the height of the newest block. There are two differences concerning the Bitcoin convergence gap of 6 and our convergence gap. (1) Our gap is parameterized by κ, while the "six block confirmation" rule guarantees the safety except for a small probability of 10^{-3} assuming a 10% rate of adversary mining power. (2) In Bitcoin, two chain branches are competed by length. Similarly, we firstly compare two chain branches by length. However, for two chain branches of the same length, we compete them by the weight, and the branch with the least weight wins.

With a Monte Carlo-based approach, we are allowed numerically calculate $\Pr[\mathsf{Win}_L]$ for each L and hence the secure convergence gap for a determined tuple of (D, D_0, α, κ). We notate the gap as $\Gamma_{D,D_0,\alpha,\kappa}$.

The above definition ignores the Merkle root which differs in each block. However, by assuming H as an ideal cryptographic hash function emulating the random oracle, the secure convergence gap can guarantee that our system is secure against the 51% attack if the adversary hash rate is bounded by a modest constant and blocks are only confirmed if the current branch excels the others by the gap.

Chain quality of PoS and PoM. The chain quality of a blockchain is defined as the expected number of blocks generated by honest miners within a certain interval of blocks. This term is hard to be formally defined in the version of the paper since we have not formally defined our execution model at a theoretical level. However, without considering block withholding and selfish mining, we may intuitively regard chain quality as the guarantee that each miner has its probability of proposing each block in the final main chain proportional to its resource, i.e., stake holding for PoS or hash power for PoM. This is actually achieved in our system with a constant α, due to the hardness of modeling the distribution of the maximum branch summation of a tree of values following a geometric distribution, we could only provide a loose upper bound of the expected adversary block percentage, which is $\sum_{i=1}^{\infty} i \cdot \Pr[\mathsf{Win}_i]$. This bound is derived from the union bound. For each block of height h and each k, Win_k possibly happens after each $B \in \mathbf{A}[h-k+1:h]$, which consists of k blocks.

Consistency of PoS and PoM. Please refer to our full version for the formal proof of common prefix and self-consistency.

Liveness of PoS and PoM. The liveness of PoS and PoM is provided in two aspects. Firstly, as discussed in Sec. 7.1, the incentive compatibility motivates block assemblers to include transactions (including qualification requests, qualification submissions and share submissions) in their assembled Merkle tree. Secondly, due to the chain quality, blocks from honest assemblers appear on the chain with a high frequency. Therefore, liveness is guaranteed even if certain malicious nodes attempt to ignore certain transactions.

DDOS Attacks. Practically, a robust system should be DDOS-secure. Potential DDOS attacks to our system are analyzed as follows.

- *Excessive Invalid Share Submissions.* The verification of a share submission is expensive due to the fact that the assembling of a Merkle tree of transactions is done by miners, instead of centralized mining pools. To prevent such a DDOS attack, and also to improve the overall performance of the system, we ask miners to submit only the Merkle root, instead of the whole tree.
- *Excessive Qualification Requests.* Qualification requests should be handled by miners with a high computational cost of generating the problem and storing

the problem in the block. Therefore, a relatively high cost of applying a qualification is required.
- *Sybil Participating.* An adversary node may pretend to be a large amount of participating nodes and have each of them take up a slot in the admissible mining list and receive an admissible mining slot for each round. The methodology of preventing excessive network burden from such Sybil attacks is still to increase the cost of a qualification request.
- *Invalid A-chain Blocks.* Invalid A-chain blocks are not forwarded by other honest participants in the P2P network. Due to the nature of a P2P network, the network burden caused by an invalid message is light.

8 Conclusion

In this article, we have proposed a novel and practical decentralized pooled mining protocol based on a newly launched blockchain. The consensus of the blockchain switches from PoS to our newly proposed PoM to secure up decentralized pooled minings by the mining power of existing blockchains. The incentive compatibility is illustrated and the security is described to be guaranteed based on security assumptions some of which are in fact guaranteed by the incentive compatibility nature of the system. We have also implemented the system and shown the performance analysis based on our implementation.

In the future, we look forward to seeing similar decentralized pooled mining protocols and implementations with different advantages and disadvantages. We also expect the formal verification of our implementations, the cryptographic modeling and proving of our system in the universal composition security model.

References

1. P2Pool: Decentralized bitcoin mining pool. http://p2pool.in/.
2. M. Andrychowicz and S. Dziembowski. PoW-based distributed cryptography with no trusted setup. In *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part II*, pages 379–399, 2015.
3. I. Bentov, A. Gabizon, and D. Zuckerman. Bitcoin beacon. arXiv CoRR abs/1605.04559, 2016.
4. I. Bentov, R. Pass, and E. Shi. Snow white: Provably secure proofs of stake. *IACR Cryptology ePrint Archive*, 2016:919, 2016.
5. D. Boneh, J. Bonneau, B. Bünz, and B. Fisch. Verifiable delay functions. In H. Shacham and A. Boldyreva, editors, *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part I*, volume 10991 of *Lecture Notes in Computer Science*, pages 757–788. Springer, 2018.
6. B. Bünz, S. Goldfeder, and J. Bonneau. Proofs-of-delay and randomness beacons in ethereum. In *IEEE Security & Privacy on the Blockchain (IEEE S&B)*, 2017.
7. R. Canetti, R. Gennaro, S. Goldfeder, N. Makriyannis, and U. Peled. UC non-interactive, proactive, threshold ECDSA with identifiable aborts. In J. Ligatti, X. Ou, J. Katz, and G. Vigna, editors, *CCS '20: 2020 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, USA, November 9-13, 2020*, pages 1769–1787. ACM, 2020.
8. M. Castro, B. Liskov, et al. A correctness proof for a practical byzantine-fault-tolerant replication algorithm. Technical report, Technical Memo MIT/LCS/TM-590, MIT Laboratory for Computer Science, 1999.

9. M. Castro, B. Liskov, et al. Practical byzantine fault tolerance. In *OSDI*, volume 99, pages 173–186, 1999.
10. S. Das, V. Krishnan, I. M. Isaac, and L. Ren. SPURT: scalable distributed randomness beacon with transparent setup. *IACR Cryptol. ePrint Arch.*, 2021:100, 2021.
11. B. David, P. Gaži, A. Kiayias, and A. Russell. Ouroboros praos: An adaptively-secure, semi-synchronous proof-of-stake blockchain. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 66–98. Springer, 2018.
12. J. Doerner, Y. Kondi, E. Lee, and A. Shelat. Threshold ECDSA from ECDSA assumptions: The multiparty case. In *2019 IEEE Symposium on Security and Privacy, SP 2019, San Francisco, CA, USA, May 19-23, 2019*, pages 1051–1066. IEEE, 2019.
13. N. Döttling, S. Garg, G. Malavolta, and P. N. Vasudevan. Tight verifiable delay functions. In C. Galdi and V. Kolesnikov, editors, *Security and Cryptography for Networks - 12th International Conference, SCN 2020, Amalfi, Italy, September 14-16, 2020, Proceedings*, volume 12238 of *Lecture Notes in Computer Science*, pages 65–84. Springer, 2020.
14. J. R. Douceur. The sybil attack. In *International workshop on peer-to-peer systems*, pages 251–260. Springer, 2002.
15. S. Dziembowski, S. Faust, V. Kolmogorov, and K. Pietrzak. Proofs of space. In *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part II*, pages 585–605, 2015.
16. N. Ephraim, C. Freitag, I. Komargodski, and R. Pass. Continuous verifiable delay functions. In A. Canteaut and Y. Ishai, editors, *Advances in Cryptology - EUROCRYPT 2020 - 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10-14, 2020, Proceedings, Part III*, volume 12107 of *Lecture Notes in Computer Science*, pages 125–154. Springer, 2020.
17. I. Eyal and E. G. Sirer. Majority is not enough: Bitcoin mining is vulnerable. In N. Christin and R. Safavi-Naini, editors, *Financial Cryptography and Data Security - 18th International Conference, FC 2014, Christ Church, Barbados, March 3-7, 2014, Revised Selected Papers*, volume 8437 of *Lecture Notes in Computer Science*, pages 436–454. Springer, 2014.
18. J. Garay and A. Kiayias. SoK: a consensus taxonomy in the blockchain era. Technical report, Cryptology ePrint Archive, Report 2018/754, 2018.
19. J. Garay, A. Kiayias, and N. Leonardos. The bitcoin backbone protocol: Analysis and applications. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 281–310. Springer, 2015.
20. Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich. Algorand: Scaling byzantine agreements for cryptocurrencies. In *Proceedings of the 26th Symposium on Operating Systems Principles*, pages 51–68. ACM, 2017.
21. J. N. Gray. Notes on data base operating systems. In *Operating Systems*, pages 393–481. Springer, 1978.
22. R. Han, J. Yu, and H. Lin. Randchain: Decentralised randomness beacon from sequential proof-of-work. *IACR Cryptol. ePrint Arch.*, 2020:1033, 2020.
23. F. P. Junqueira, B. C. Reed, and M. Serafini. Zab: High-performance broadcast for primary-backup systems. In *2011 IEEE/IFIP 41st International Conference on Dependable Systems & Networks (DSN)*, pages 245–256. IEEE, 2011.
24. A. Kiayias, A. Russell, B. David, and R. Oliynykov. Ouroboros: A provably secure proof-of-stake blockchain protocol. In *Annual International Cryptology Conference*, pages 357–388. Springer, 2017.
25. E. K. Kogias, P. Jovanovic, N. Gailly, I. Khoffi, L. Gasser, and B. Ford. Enhancing bitcoin security and performance with strong consistency via collective signing. In *25th USENIX Security Symposium (USENIX Security 16)*, pages 279–296, 2016.
26. E. Kokoris-Kogias, P. Jovanovic, L. Gasser, N. Gailly, E. Syta, and B. Ford. Omniledger: A secure, scale-out, decentralized ledger via sharding. In *2018 IEEE Symposium on Security and Privacy (SP)*, pages 583–598. IEEE, 2018.
27. L. Luu, V. Narayanan, C. Zheng, K. Baweja, S. Gilbert, and P. 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.

28. L. Luu, Y. Velner, J. Teutsch, and P. Saxena. Smartpool: Practical decentralized pooled mining. In E. Kirda and T. Ristenpart, editors, *26th USENIX Security Symposium, USENIX Security 2017, Vancouver, BC, Canada, August 16-18, 2017*, pages 1409–1426. USENIX Association, 2017.
29. A. Miller, A. Juels, E. Shi, B. Parno, and J. Katz. Permacoin: Repurposing bitcoin work for data preservation. In *2014 IEEE Symposium on Security and Privacy, SP 2014, Berkeley, CA, USA, May 18-21, 2014*, pages 475–490, 2014.
30. S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008.
31. S. Park, A. Kwon, G. Fuchsbauer, P. Gazi, J. Alwen, and K. Pietrzak. SpaceMint: A cryptocurrency based on proofs of space. In *Financial Cryptography and Data Security - 22nd International Conference, FC 2018, Nieuwpoort, Curaçao, February 26 - March 2, 2018, Revised Selected Papers*, pages 480–499, 2018.
32. R. Pass, L. Seeman, and A. Shelat. Analysis of the blockchain protocol in asynchronous networks. In J. Coron and J. B. Nielsen, editors, *Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part II*, volume 10211 of *Lecture Notes in Computer Science*, pages 643–673, 2017.
33. R. Pass and E. Shi. Thunderella: Blockchains with optimistic instant confirmation. In *Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part II*, pages 3–33, 2018.
34. W. J. Paul, R. E. Tarjan, and J. R. Celoni. Space bounds for a game on graphs. *Mathematical Systems Theory*, 10:239–251, 1977.
35. K. Pietrzak. Simple verifiable delay functions. In A. Blum, editor, *10th Innovations in Theoretical Computer Science Conference, ITCS 2019, January 10-12, 2019, San Diego, California, USA*, volume 124 of *LIPIcs*, pages 60:1–60:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.
36. D. Skeen. Nonblocking commit protocols. In *Proceedings of the 1981 ACM SIGMOD international conference on Management of data*, pages 133–142. ACM, 1981.
37. S. Tang. A few explanations for <fast-to-finalize Nakamoto-like consensus>. *IACR Cryptol. ePrint Arch.*, 2020:810, 2020.
38. S. Tang and S. S. M. Chow. Systematic market control of cryptocurrency inflations. In S. V. Lokam, S. Ruj, and K. Sakurai, editors, *Proceedings of the 2nd ACM Workshop on Blockchains, Cryptocurrencies, and Contracts, BCC@AsiaCCS 2018, Incheon, Republic of Korea, June 4, 2018*, pages 61–63. ACM, 2018.
39. S. Tang, S. S. M. Chow, Z. Liu, and J. K. Liu. Fast-to-finalize Nakamoto-like consensus. In J. Jang-Jaccard and F. Guo, editors, *Information Security and Privacy - 24th Australasian Conference, ACISP 2019, Christchurch, New Zealand, July 3-5, 2019, Proceedings*, volume 11547 of *Lecture Notes in Computer Science*, pages 271–288. Springer, 2019.
40. M. Vukolic. The quest for scalable blockchain fabric: Proof-of-work vs. BFT replication. In *Open Problems in Network Security - IFIP WG 11.4 International Workshop, iNetSec 2015, Zurich, Switzerland, October 29, 2015, Revised Selected Papers*, pages 112–125, 2015.
41. B. Wesolowski. Efficient verifiable delay functions. *J. Cryptol.*, 33(4):2113–2147, 2020.
42. M. Zamani, M. Movahedi, and M. Raykova. Rapidchain: Scaling blockchain via full sharding. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018*, pages 931–948, 2018.

A Backgrounds for Beginners

A.1 Blockchain Consensus

The core of any blockchain is the underlying distributed consensus. Most existing consensus can be classified into two categories.

- *Permissioned Consensus.* Before the proposal of Bitcoin and blockchain in 2008, most distributed consensus schemes assume a predetermined set of participants. Such consensus schemes are known as permissioned consensus. After the emergence of blockchains, permissoined consensus schemes are not only adopted by certain blockchains but also improved and thoroughly studied in both distributed consensus and security literature.
- *Permissionless Consensus.* Permissionless consensus is defined as the set of distributed consensus allowing the dynamical entering and quitting of participants. The Bitcoin consensus, which is the first blockchain, is known as the first permissionless consensus for a public ledger.

After a decade of academic and industrial progress, a rich variety of different permissionless consensus schemes are proposed to improve or alternate blockchains, including *proof-of-work*, *proof-of-stake*, and *proof-of-space*.

- *Proof-of-Work.* Most early cryptocurrencies adopt proof-of-work (PoW) as the underlying consensus of the blockchain, including Bitcoin and Ethereum. In short, PoW makes the probability of having each participant mine a new block in a certain period of time proportional to its hash power (i.e., the average greatest possible number of trying to solve a hash puzzle per second). At a higher level, PoW is essentially a proof of computational resource for a certain hash function and an allocation of rewards according to its proved resource holding.
- *Proof-of-Stake.* Likewise, proof-of-stake (PoS) is essentially proof of stakeholding and an allocation of rewards according to the amount of proved (or checked) stakeholding of each participant. Different from PoW, the proof is explicit in most cryptocurrencies since the balance of each participant is recorded in the public ledger.
- *Proof-of-Space.* There are also other forms of resource proofs, like proof-of-space (or proof-of-capacity, PoC). PoC is the proof of the physical storage space of each participant. We omit discussions of any specific PoC protocol since more tricky probabilistic methods are required to introduce a PoC protocol.

In summary, all known permissionless distributed consensus can be considered as a certain type of *resource proof*.

Actually, on a philosophical level, resource proof is a must for any permissionless consensus allowing dynamical entering and quitting of untrusted public nodes. Informally, with a certain loss of rigorousness, we may consider

$$\text{permissionless consensus} \approx \text{leader election (resource proof)} + \text{ledger extension (classical consensus)}.$$

We argue that, at a high level, any permissionless distributed consensus can be abstracted as such two phases.

- *Leader Election.* Any permissionless distributed consensus selects one or a group of leaders for each round (no matter how the term *round* is defined). For example, a classical Bitcoin-liked blockchain selects the leader as the person successively mining a block.

– *Ledger Extension.* The one or a group of leaders have the right and the duty to extend the public ledger for each round. There are two major forms of the ledger. Some distributed ledgers take the configuration, i.e., the whole mapping from account addresses (public keys) to their balances. Such form of the ledger is known as the *account-based* ledger. Note that this does not imply that all blocks with account-based ledgers have to record the whole mapping, there are specifically designed *persistent data structures* to reduce the space complexity of each block to $O(m \log n)$ where m is the number of total transactions of the block and n stands for the number of total node in the Merkle tree. Another form of the ledger, which is originally adopted in Bitcoin, is known as *UTXO-based ledgers*. Such ledgers only record transactions rather than the whole mapping. The data structure of UTXO guarantees that no user can pay more than its balance. However, for lightweight nodes, UTXO-based ledgers can hardly support frequent queries to the total balance of one account.

In most classical blockchains, the two phases above are somewhat combined – the mining of a new valid block is both a leader (the block proposer, in this particular case) election and a ledger extension. The miners do not assemble blocks after successive mining, but before the mining. However, in some hybrid consensus schemes, two phases are explicitly separated. In fact, the ledger extension phase can be somewhat regarded as the same functionality as any permissioned consensus. Informally, this implies that the must of a fair leader election makes the key difference between permissionless and permissioned consensus. However, a fair leader election comes from "outer forces" or it is vulnerable in face of Sybil attacks. Due to the decentralized nature of most cryptocurrencies, "outer forces" provided by centralized authorities are prohibited. Thereby, such forces must come from the proof of certain resources.

A.2 Chain Forks and The Main Chain

A chain fork is defined as the configuration where two or more blocks are generated from the same previous block. Due to the possibility of having two nodes finding a valid hash puzzle solution almost simultaneously, and the necessity of avoiding explicitly malicious blocks, forks would happen. A sound blockchain resolves forks by *chain competitions*. For example, in Bitcoin, honest miners always mine after the longest branch, and only the branch with a length six blocks greater than all other branches win the competition. The *main chain* refers to the current longest chain branch.

A.3 Centralized Pooled Mining

Due to the hardness of the hash puzzle of most existing PoW-based cryptocurrencies, miners have to face a great variation of revenue. Most miners are almost

incapable of mining one valid block within one year. Therefore, pooled mining becomes the optimal choice for them to amortize the same expected revenue with a significantly smaller variation.

Mining pools are often centralized nodes providing a mining template, mostly including the set of transactions to be included in the block. Most miners try to solve the same puzzle but with a smaller difficulty according to the template. Once a valid solution is found, it submits the solution (known as a *share*) to the pool to obtain certain rewards. Even if a valid solution to the puzzle with the original difficulty is found, the miner cannot take the revenue for itself since the mining template has specified the receiving address. Such centralized mining pools are notorious for several reasons.

- The decentralizing principle of cryptocurrencies is compromised. Centralized pooled mining brings about the phenomenon that major hash power is in fact under the control of few pools. This deviates from the decentralized principle of cryptocurrencies. Moreover, the following issues arise along with such a deviation.
- Centralized mining pools are entitled to excessive power. In fact, due to the power of deciding mining templates, pools are allowed to choose only transactions of their favor. Pools with excessive total controlled hash power may even have the power to launch a "51% attack" to cause a fork of their taste.
- Even the expected revenue of miners can be reduced by major pools. In centralized pooled mining, the expected revenue is lessened due to the deduction from pools. The consistent revenue of mining is guaranteed by the soundness of its chosen mining pool. Therefore, pools with a rich total hash power are attracting more participating nodes and hence few giants are formed. To guarantee the consistency of mining revenue, most nodes choose to join such a few giant pools. This provides the giants more confidence to raise the revenue deduction.

A.4 Decentralized Pooled Mining

To solve the above issues brought about by centralized mining pools, solutions to decentralize the pooled mining procedure are proposed, including P2Pool and SmartPool.

- P2Pool leverages another blockchain with a smaller mining difficulty. Any block meeting the difficulty can be accepted as one block of this chain, as long as certain conditions of the block structure are met.
- SmartPool is a decentralized pooled mining protocol based on smart contracts. Though realized by smart contracts of Ethereum, it can support both Ethereum mining and the mining of other PoW-based blockchains.

Protocol Π_{A}

Each A-chain block $A = (A.\mathsf{header}, A.\mathsf{Merkle}, A.\mathsf{nonce})$ consists of a header, a Merkle tree, and a nonce. Traditionally, the nonce value is part of the block header. However, we move it out of the scope of a block header to facilitate descriptions.

$A.\mathsf{header} = \mathsf{pk} \,||\, A.\mathsf{Merkle.root} \,||\, r_A \,||\, \mathsf{next\_com} \,||\, \mathsf{param} \,||\, \mathsf{aux}$ includes the public key of pk the block assembler, the Merkle root $A.\mathsf{Merkle.root}$, a random source r_A, the next committee list $\mathsf{next\_com}$ for every Q blocks, a set of parameters to maintain params, and certain auxiliary information.

$\mathsf{L}(A.\mathsf{Merkle}) = A.\mathsf{transaction} \,||\, \mathsf{qualify} \,||\, \mathsf{share} \,||\, \mathsf{mining\_list} \,||\, \mathcal{L}$, where $A.\mathsf{transaction} = \mathsf{norm}||\mathsf{reward}$ includes normal A-chain coin transactions and rewarding transactions for the block proposer and share submitters, $\mathsf{qualify}$ specifies the mining task for each qualification request, $\mathsf{mining\_list}$ is the set of admissible miners, $\mathcal{L}$ is empty for most blocks while includes unsigned B-chain revenue allocating transactions in each block of height $kW - d$ ($k \in \mathbb{N}$).

– **Block Mining.**

• The A-chain miner of address pk keeps track of all valid A-chain block extensions, it locally maintains a sequence of blocks containing potential forks, thereby it is a tree of blocks. Its local main chain is the chain branch with the greatest total weight in its view.

• For each new block A of the main chain with block hash h:

1. It obtains the Merkle root corresponding to h. Note that since A-chain mining does not take the current Merkle root into the puzzle, the Merkle tree is not hash-chained by A, but hash-chained by the next block mining via h. Therefore, a separate signature σ_m is required to guarantee the integrity of $A.\mathsf{Merkle}$. It receives $(\texttt{separate\_Merkle\_sig}, \sigma_m, A.\mathsf{Merkle.root})$ and checks whether $\mathsf{Vrf}(A.\mathsf{header.pk}, A.\mathsf{Merkle.root}, \sigma_m) = 1$.
2. It fetches its admissible range Ran of nc, the rule of which differs to PoS and PoM, see details in Sec. 4.
3. Let $k = 0$.
4. If there exists a solution $\mathsf{nc} \in \mathsf{Ran}$ such that $H(h||\mathsf{pk}||\mathsf{nc}||k) < \mathsf{target}_A$, then jump to execute the block assembling phase. Else, increase $k := k + 1$ and repeat the current step.

– **Block Assembling.**

• The block assembler calculates the random source $r_A := h_{-1} \oplus \mathsf{Eval}(h_{-d})$ of the block to assemble, where h_{-i} stands for the hash of the i-th previous block.

• Maintain the list of parameters params, including the current PK.

• It arranges the list of qualification requests and submissions to be tackled by the current block. Set $\mathsf{qualify} := \epsilon$.

- For each qualification request $(\texttt{qualification\_request}, \mathsf{pk}_c)_{\mathsf{pk}_c}$ in the form of a specialized transaction, if the transaction fee is sufficient, let $\mathsf{qualify} := \mathsf{qualify} \,||\, (\texttt{qualification}, R, \mathsf{pk}_c, (s,t))$ where R is the current block height, (s,t) restricts its admissible nonce slot. Here $s := H(\texttt{qualification}, R, \mathsf{pk}_c, r_A) \mod 2^\kappa$ and $t := s + \Omega'$.
- For each qualification submission $(\texttt{qualification\_answer}, R', \mathsf{pk}_c, (s,t), \mathsf{rec})_{\mathsf{pk}_c}$, check whether there is an item of $(\texttt{qualification}, R', \mathsf{pk}_c, (s,t))$ in the $\mathsf{qualify}$ of block height R', abort if not. Fetch the current block height R and check whether $R - R' \le \Delta + \delta$, abort if not. Parse $\mathsf{rec} = ((\mathsf{header}_1, \mathsf{nonce}_1), (\mathsf{header}_2, \mathsf{nonce}_2), \ldots, (\mathsf{header}_\ell, \mathsf{nonce}_\ell))$, check if $s \le \mathsf{nonce}_i \le t$ and $H(\mathsf{header}_i, \mathsf{nonce}_i) < D_s$ for each $i \in [\ell]$, abort if not. Abort if $\ell < K$ where K is a predetermined constant threshold. Then update the mining list $\mathsf{mining\_list} := \mathsf{mining\_list}||\mathsf{pk}_c$.

• For each $(\texttt{share\_submission}, R, \mathsf{pk}, B.\mathsf{header}, B.\mathsf{nonce})$, check whether $H(B.\mathsf{header}||B.\mathsf{nonce}) < D_s$ and $B.\mathsf{nonce}$ is in the range of the admissible nonce slot of any block in $\mathbf{A}[:-d]$. Sort all valid share submissions into share.

• For each $(\texttt{share\_submission\_agg}, R, \mathsf{pk}, B.\mathsf{header}, K, \mathsf{pf}, s_i, t_i)$, check whether pf is a valid NIZK proof and (s_i, t_i) is the range of the admissible nonce slot of any block in $\mathbf{A}[:-d]$. Add into share K shares.

• If the height of A equals $kW - d$ for any $k \in \mathbb{N}$, assemble $\mathcal{L}$ according to share in each block in $\mathbf{A}[:-W]$.

• If the current height R is a multiple of Q, then assemble $\mathsf{next\_com}$ by the following procedure:

1. Let S be the sequence of the submitters of all shares within $\mathbf{A}[R - Q + 1, R]$ in the lexicography order. Note that there can be nodes submitting more than one share, then it repeatedly appears in S for the time equal to the number of its shares.
2. For each $i \in |\mathsf{com}|$, let $j := \mathsf{PRF}_{r_A \oplus 1}(i) \mod |S|$, let $\mathsf{com}_i := S[j]$.
3. Let $\mathsf{next\_com} := (\mathsf{com}_1, \mathsf{com}_2, \ldots, \mathsf{com}_{|\mathsf{com}|})$.

• It packs $A.\mathsf{transaction} = \mathsf{norm}||\mathsf{reward}$, where norm consists of ordinary A-chain coin transactions and reward consists of its block reward and instant A-chain rewards for each share submitter.

• It assembles $A = (A.\mathsf{header}, A.\mathsf{Merkle}, A.\mathsf{nonce})$, where $A.\mathsf{header} = \mathsf{pk} \,||\, A.\mathsf{Merkle.root} \,||\, r_A \,||\, \mathsf{next\_com} \,||\, \mathsf{param} \,||\, \mathsf{aux}$, $\mathsf{L}(A.\mathsf{Merkle}) = A.\mathsf{transaction} \,||\, \mathsf{qualify} \,||\, \mathsf{share} \,||\, \mathsf{mining\_list} \,||\, \mathcal{L}$, and $A.\mathsf{nonce} = \mathsf{nc}$.

• It produces a separate signature σ_m for the Merkle root and broadcasts the signature $(\texttt{separate\_Merkle\_sig}, \sigma_m, A.\mathsf{Merkle.root})$.

Fig. 1. The Full Protocol for A-Chain Block Assemblers

Protocol Π_{B}

We ideally regard that each B-chain block $B = (B.\mathsf{header}, B.\mathsf{Merkle}, B.\mathsf{nonce})$ consists of a header, a Merkle tree of transactions, and a nonce. Note that, differently from related literature, we do not consider the nonce as part of the header to facilitate descriptions.

– **Qualification.**

• For a miner not yet participating in our system, it submits a qualification request $(\texttt{qualification\_request}, \mathsf{pk})_{\mathsf{pk}}$ to the A-chain network in attempt of joining our pooled mining.

• After receiving the response in the A-chain block Merkle tree $(\texttt{qualification}, R, \mathsf{pk}_c, (s,t))$, it sequentially enumerates nonce values starting from s for Δ A-chain block generation intervals. Assemble all its discovered nonce meeting the share difficulty into $\mathsf{rec} = ((\mathsf{header}_1, \mathsf{nonce}_1), (\mathsf{header}_2, \mathsf{nonce}_2), \ldots, (\mathsf{header}_\ell, \mathsf{nonce}_\ell))$ such that $H(\mathsf{header}_i, \mathsf{nonce}_i) < D_s$ for each $i \in [\ell]$. Submit $(\texttt{qualification\_answer}, R, \mathsf{pk}, (s,t), \mathsf{rec})_{\mathsf{pk}}$ to the A-chain network if $\ell \geq K$.

– **Pooled Mining.**

• The thread of pooled mining starts when pk appears in $\mathsf{mining\_list}$ of any confirmed A-chain block.

• The mining protocol involves three threads with a tuple of shared variable $(h_{-1}, \mathsf{PK}, B.\mathsf{header}, B.\mathsf{Merkle}, R, U)$. Note that, for fairness, all mining attempts are suggested to be withheld within A-chain rounds of $kQ - d$ to kQ.

- One thread keeps track of the longest chain branch of B-chain, maintains h_{-1} as the hash of the latest B-chain block in the longest valid branch. It also keeps track of PK which switches for each Q rounds. Based on h_{-1}, PK, and the collection of B-chain transactions, it assembles $B.\mathsf{header}$ and $B.\mathsf{Merkle}$. Moreover, it maintains R as the height of the latest confirmed A-chain block and maintains $U = \left(\frac{(i-1)Q}{M} + 1, \frac{iQ}{M}\right)$ $(M = |\mathsf{mining\_list}|, i = \mathsf{PRF}_{r_{A'}}(\mathsf{index}_{\mathsf{pk}}) \mod M)$ as the admissible nonce slot specified by the latest confirmed A-chain block A'.
- The other thread, without repetitions, enumerate elements $\mathsf{nc} \in U$. If $H(B.\mathsf{header}, \mathsf{nc}) < D_s$, submit $(\texttt{share\_submission}, R, \mathsf{pk}, B.\mathsf{header}, B.\mathsf{nonce})$ to the A-chain network. If $H(B.\mathsf{header}, \mathsf{nc}) < D$, then also submit the whole block $B = (B.\mathsf{header}, B.\mathsf{Merkle}, B.\mathsf{nonce})$ to the B-chain network.
- For each confirmed A-chain block A' of height kQ $(k \in \mathbb{N})$, check whether it is selected to the new committee by testifying whether $\mathsf{pk} \in A'.\mathsf{next\_com}$. If it holds, then execute a subroutine of the committee protocol Π_{com}.

Fig. 2. The Full Protocol for B-Chain Miners

Protocol Π_{com}

The committee protocol is executed jointly by $|\mathsf{com}|$ parties $\mathsf{com} = (P_1, P_2, \ldots, P_{|\mathsf{com}|})$. Each party initiates the participation in the protocol upon finding its address in $\mathcal{L}$ of the latest confirmed block.

– **Committee Initiation.**

• The parties in com jointly execute the protocol $\Pi_{\text{sig-setup}}$ to attain their own share of the key pairs $(\mathsf{SK}, \mathsf{PK})$.

• They jointly release PK in the newest block template with their joint signature produced by $\Pi_{\text{sig-sign}}$. The detailed description of the message data structure and the dynamic maintaining of PK in other protocols are not described formally for simplicity.

– **Committee Subroutine.**

• All parties in com locally maintain the A-chain blockchain. For each block of height $kW - d$ $(k \in \mathbb{N})$, it fetches its contained $\mathcal{L}$ and verify that it is a fair allocation to miners according to our protocol.

• If $\mathcal{L}$ is not empty, execute the shared protocol $\Pi_{\text{sig-sign}}$ with all members in com to produce a signature σ_i for each element $\mathsf{tx}_i \in \mathcal{L}$.

• Each member in com broadcasts $\left\{\widetilde{\mathsf{tx}}_i = (\mathsf{tx}_i, \sigma_i)\right\}$ to the B-chain network.

– **Committee Switchover.**

• Upon receiving the A-chain block of height kQ $(k \in \mathbb{N})$. The committee terminates by jointly broadcast $(\texttt{com\_end}, \mathsf{com})$ appended with an aggregated joint signature of it. Note that Q is a multiple of W such that all shared mined with the current mining template are rewarded before the committee rotation.

Fig. 3. The Committee Protocol

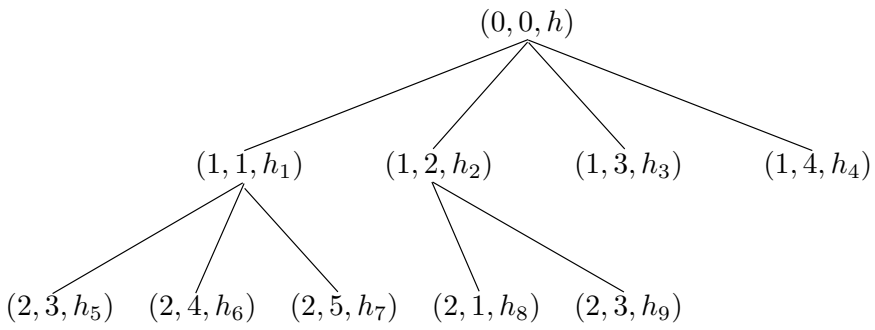

Fig. 4. Branch Decision Tree