PGChain

Technical White Paper
Version 1.0

May 18, 2022

PGChain R&D Team

Pangu Foundation.

Email: info@pgchain.org*

*Comments and feedback are highly appreciated, but all bugs and errors belong to the author.

Abstract

In this paper, we outline the architectural design of the PGChain technology and solution. PGChain is a public EVM (Ethereum Virtual Machine) compatible blockchain with the following advantages: low transaction fees, fast confirmation times, double verification and randomization for security. In particular, we propose Proof-of-Stake Voting (PoSV) consensus, a Proof-of-Stake (PoS)-based blockchain protocol with a fair voting mechanism, strict security guarantees, and fast finality. We also propose a novel reward mechanism and show that, under this mechanism, the blockchain fork probability is low, confirmation time is fast, plus the contributions and benefits of masternodes are fair because the probability distribution function is ultimately uniform of.

Index Terms

Blockchain, Ethereum, PGChain, Proof-of-Stake Voting, Masternode, Randomization, Security Protocol.

I. INTRODUCTION

The infrastructure of the blockchain industry and the Internet of Value is being built rapidly around the world, and in the eyes of many, the atmosphere is very similar to the Internet construction in the late 1990s, where pioneers and dreamers come together to innovate the future. PGChain can become a platform by seamlessly merging the application ecosystem with encrypted tokens used by millions of mainstream users, enabling fast, secure, frictionless payments and trusted storage of value through a unique blockchain infrastructure architecture. the dominant part of this phenomenon.

Distributed systems have been studied in "permissioned settings" where the number of participants in the system and their identities are well known. In 2008, Satoshi Nakamoto - "proposed his famous "blockchain protocol" which attempts to achieve consensus in a permissionless environment: anyone can join (or leave) the protocol execution (without obtaining a centralized or distributed licenses from established institutions), and the protocol instructions do not depend on the identity of the player". Later, Ethereum and its Ethereum Virtual Machine (EVM) proposed several major improvements over Bitcoin, including smart contracts. Both Bitcoin and Ethereum have some issues, especially transaction processing performance. In order to build an efficient and secure consensus protocol for PGChain, we solve the following main bottlenecks of classic blockchains:

- **Efficiency:** Existing blockchains as employed by major crypto-currencies(e.g., Bitcoin or Ethereum) do not scale well to handle a large transaction volume, e.g. Bitcoin and Ethereum can handle around 10 transactions/second. This small throughput severely hinders a wide-spread adoption of such crypto- currencies.

- **Confirmation time:** Bitcoin block time of 10 minutes is significantly greater than network latency. Additionally, a Bitcoin block requires 5 subsequent blocks to be confirmed; therefore, it takes an average of an hour to confirm a transaction (low confidence). While Ethereum uses smaller block times, the average confirmation time is still relatively high at around 13 minutes. These long confirmation times hinder many important applications (especially smart contract applications).

- **Fork Generation:** The problem of forked chains consumes computational energy, time, and creates potential vulnerabilities for different types of attacks.

Motivated by the above, our persistent and ultimate research goal is to propose a consensus protocol, focusing on the following key strategies:

- Two-factor verification to enhance security and reduce forks
- Randomization ensures fairness and prevents handshake attacks
- Fast confirmation times and efficient finality or rebasing checkpoints

To start dealing with these issues, in this article, we introduce the overall architectural design of the PGChain masternode. In particular, we propose Proof-of-Stake Voting (PoSV) consensus, a Proof-of-Stake (PoS)-based blockchain protocol with a fair voting mechanism, strict security guarantees, and fast finality. We also propose a novel reward mechanism and show that, under this mechanism, the blockchain fork probability is low, confirmation time is fast, plus the contributions and benefits of masternodes are fair because the probability distribution function is ultimately uniform of.

Structure of the rest of the paper. Section II-A explains the intuitive idea of masternodes, frameworks and background protocols and outlines the architectural design to help general readers (e.g. investors, traders, etc.) who may not have technical knowledge easily understand our mechanism.

Section II-B introduces the PGChain stakeholder policy, masternode committee voting system and reward mechanism. Section II-C explains the protocol's motivation and double verification process and finality checkpoint. In Section II-D, we mathematically present the formalization of our model to demonstrate the soundness of our model and protocol. Section III discusses security analysis and countermeasures against potential attacks. We discuss and compare PGChain with several existing blockchains in Section IV. Finally, we conclude this paper in Section V.

II. PGChain MASTERNODE DESIGN

A. The PGChain architecture

The PGChain blockchain is produced and maintained in a consistent manner by a set of master nodes through the PGChain consensus protocol, as shown in Figure 1. These masternodes are full nodes that hold PGCs. For a coin holder to become a masternode, two requirements must be met:

- Coin holders must hold at least the minimum required number of coins (see next section for details).
- The holder must be one of the most voted masternode candidates in the system. Token holders' votes are recorded through the Voting DApp, which allows token holders to send PGC through a smart contract mechanism.

In addition to an improved voting system over the current Bitcoin and Ethereum blockchains, PGChain also provides a new technology, a two-factor verification and randomization mechanism. This new technology significantly reduces the possibility of invalid blocks in the blockchain. These enhancements and components of PGChain will be detailed step by step below.

B. Stakeholders & Voting

Coin Holders, Masternodes

Token holders are as simple as their name: users who join the network, users who own and transfer PGC. Masternodes are full nodes that maintain a copy of the blockchain, generate blocks, and keep the chain consistent. It is worth noting that PGChain does not have miners in current proof-of-work based blockchain systems like Bitcoin and Ethereum. Only masternodes can generate and validate blocks.

Masternodes are selected through a voting system. The first requirement to become a masternode is to deposit 30,000 PGC into the voting smart contract. These depositors are then listed as masternode candidates in the Voting DApp, allowing holders to vote for them by sending PGC to the smart contract.

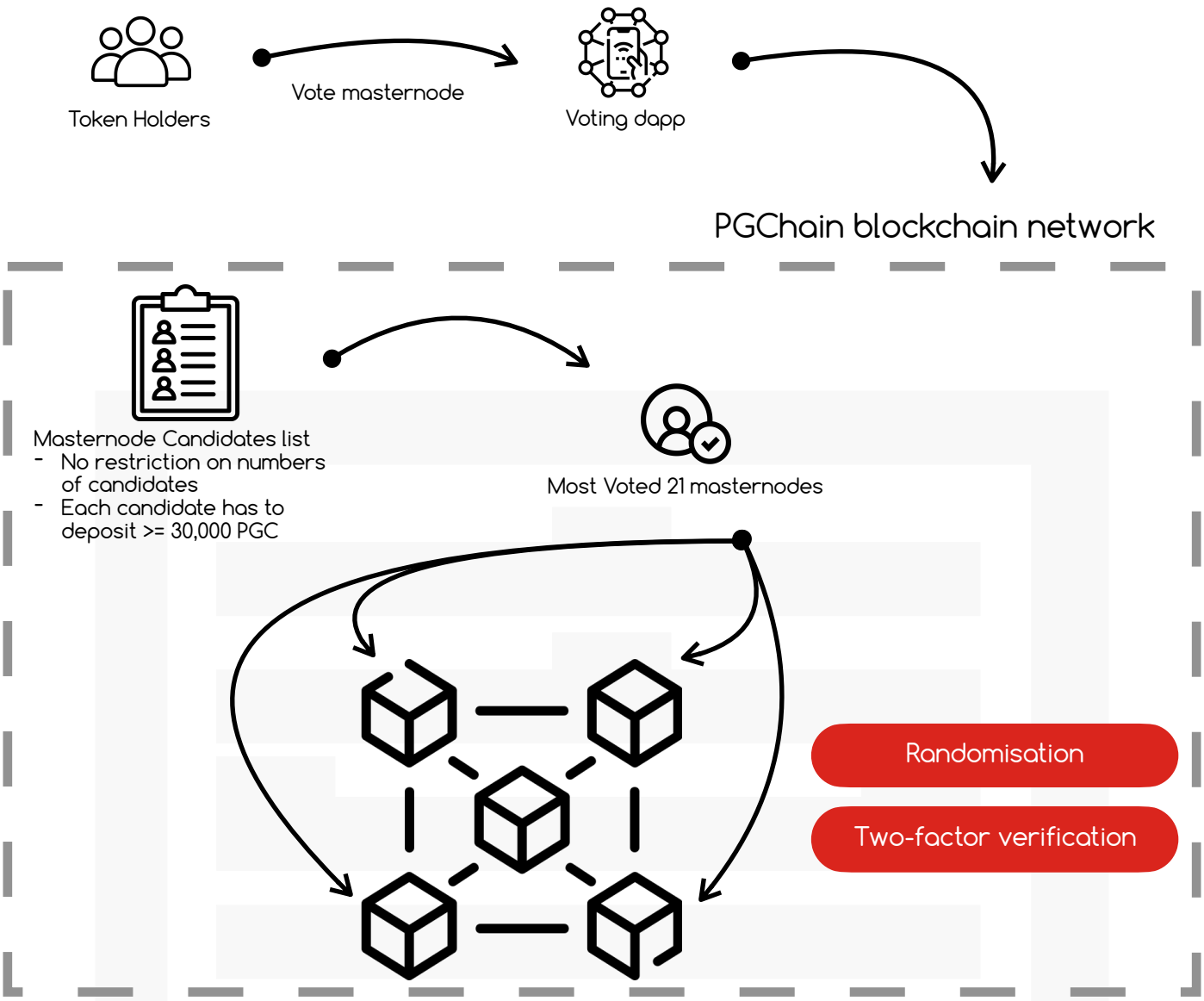

Fig. 1. PGChain architecture

Masternodes who work hard to create and validate blocks in the system will be incentivized by PGC. Additionally, token holders who vote for these incentivized masternodes will also receive PGC proportional to the amount of PGC they invested through voting. PGChain engineers are responsible for designing a fair, clear, automated and responsible reward mechanism.

Masternode candidates are dynamically sorted based on voted coins. The performance of masternodes will be tracked and reported to token holders based on three main metrics: a CPU/ memory graph that ensures the masternode's workload, the number of signature blocks indicating the performance of their work, and the last signature block to find out their last Activity. Holders can at any time un-vote a less-performing masternode and vote for another better-performing masternode. Token holders have an incentive to do so because the tokens they vote for are seen as an investment in the masternodes they support, so they should choose a voting strategy to maximize the profit they get from their investment.

This simple trick keeps the system healthy because masternodes always have to compete for their position in order to eventually eliminate all weak masternodes. Therefore, only the most powerful masternodes are voted to thrive.

Voting & Masternode Committee

The masternode committee elects a maximum of ninety-nine masternodes. The required deposit amount for the masternode role is set to 30,000 PGC. This amount is locked in the voting smart contract. Once a masternode is demoted (not remaining in the first twenty-one voting masternodes) or deliberately withdraws from the masternode candidate list/masternode committee, deposits will be locked for one month.

Holders can vote with any number of votes at any time (this is actually calculated based on the amount of PGC they stake on certain masternode candidates). They can vote using the masternode's performance statistics in the governance voting DApp as reference information. The masternode set is dynamically ordered based on the number of PGCs and counts up to 21 upon receipt of votes.

Reward Mechanism

For each iteration of 900 blocks (called an epoch), a checkpoint block is created, which only implements reward work. Masternodes that create blocks in a round-robin and sequential turn must scan all created blocks in an epoch and count the number of signatures. The design of the reward mechanism follows the following policy: the more signatures a masternode has, the more reward he gets. For example, Masternode A, which seals twice as many blocks as Masternode B, gets twice as many PGCs as Masternode B in one epoch.

In addition, there is also a reward sharing ratio between token holders and elected masternodes supported by token holders. Specifically, each epoch consists of 900 blocks, with a total of 250 PGC awarded for the first two years. This amount of 250 PGC will be distributed to all masternodes proportionally based on the number of signatures they signed during that epoch. After that, the reward received by each masternode will be divided into five parts.

- **Infrastructure Reward**: The first portion of 7% called Infrastructure Reward goes to the Masternode.
- **Staking Reward**: The second portion of 70% called Staking Reward goes to the pool of all voters for that Masternode which is shared proportionally based on the token stake.
- **Community Reward**: The third portion of 5% called Community Reward goes to the community contributors.
- **PG NFT Reward**: The forth portion of 8% called PG NFT Reward goes to the pool of all PG NFT Holders which is shared proportionally based on the total number of PG NFTs.
- **Foundation Reward**: The last portion of 10% called Foundation Reward goes to a special account controlled by the Pangu Foundation, which is run by PGChain foundation initially.

It is worth noting that coin-holders who unvote before the checkpoint block will not receive any shared reward in the **Staking Reward** portion.

C. PGChain Consensus Protocol

Two-factors (Double) Validation Process

In PGChain, master nodes share the responsibility of running the system and maintaining the stability of the system. A full node should run on a strong hardware configuration and a high-speed network connection to ensure the required block time (target is 2 seconds). Only masternodes can produce and seal blocks. To this end, PGChain consensus relies on the concept of double verification, which improves some existing consensus mechanisms, namely single verification. Let's first introduce Double Validation, and then analyze the differences and improvements between Double Validation and Single Validation.

Double Validation (DV): Similar to some existing PoS based blockchains (eg Cardano), each block is created by block producers (i.e. masternodes) who follow a predetermined and recurring masternode for each epoch The sequence takes turns creating blocks. However, unlike these existing blockchains, the DV in PGChain requires the signatures of two masternodes on a block to push the block to the blockchain. One of the masternodes is the block creator, while the other, the block validator, is randomly selected from a set of voting masternodes to validate and sign blocks. In the following, for more convenience, block creator and block validator are used interchangeably to represent masternode 1 (block producer) and the randomly selected masternode 2, respectively. The process of randomly selecting block validators is detailed in the next paragraphs. Note that in proof-of-work based blockchains such as Ethereum and Bitcoin, there is no mining in block creation. This means that a created block is valid if and only if it is signed by enough two from a block creator and a corresponding block verifier to confirm its correctness.

We believe this DV technology enhances the stability of the blockchain by reducing the likelihood of producing "garbage" blocks, while still maintaining the security and consistency of the system. The randomisation of block validators in DV is a key factor in reducing the risk of paired masternodes attempting to commit. Furthermore, by leveraging DV technology, PGChain requires only two signatures per block compared to some current public blockchains in the market . To demonstrate our improvement over existing PoS-based blockchains, we analyze the differences between DV and single verification mechanisms in some existing blockchains, as shown below.

The improvement of double-validation over single-validation: Let us demonstrate the improvement of DV compared to single-validation by analyzing some attack scenarios, as shown in Figure 2 and Figure 3.

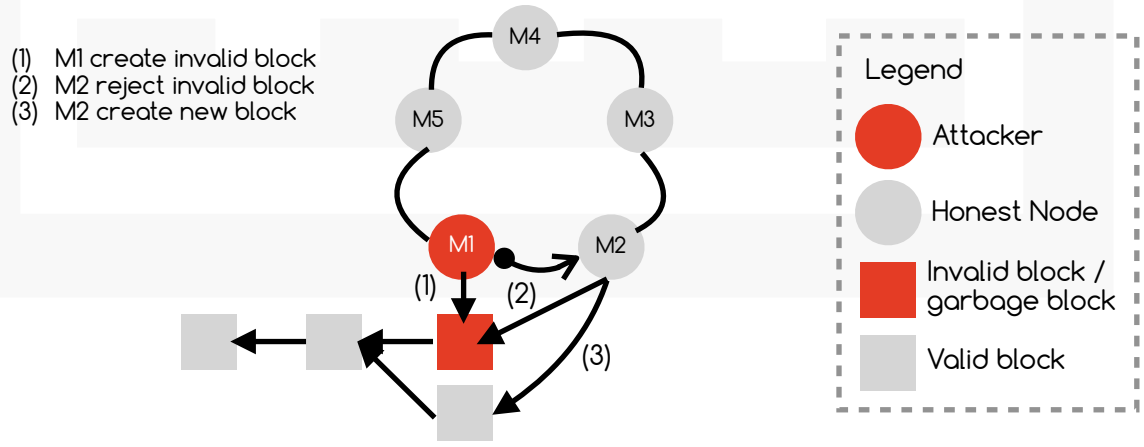

Fig. 2.a Single Validation (SV): SV with block creation masternode as an attacker

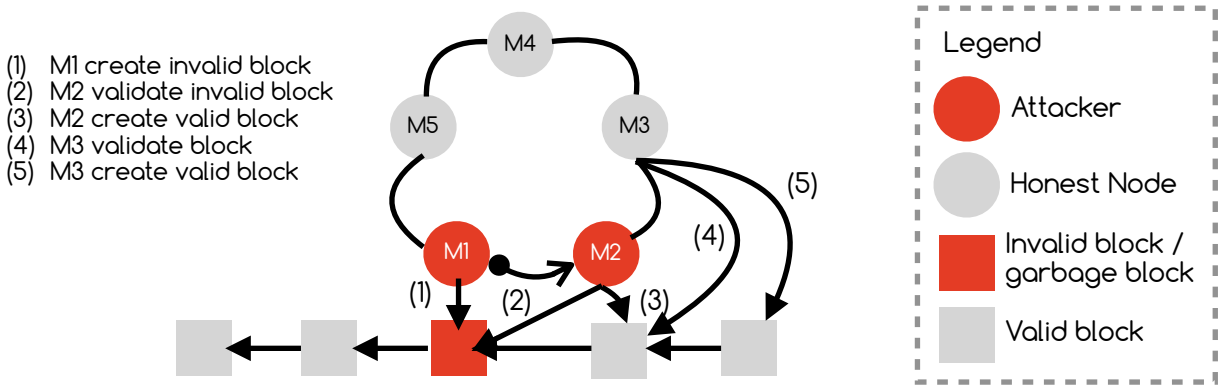

Fig. 2.b Single Validation (SV): SV with two consecutive block creation masternodes as attackers

- **Single Validation:** In Single Validation, in an epoch, each masternode, e.g. M1, sequentially takes its turn to create a block, e.g. block100. The next masternode, e.g. M2, in the sequence then validates the created block100. If block100 is invalid (that potentially means that M1 is an attacker) and contains a transaction that invalidly benefits M1, if M2 is honest (see Fig. 2.a), it rejects block100 and creates another block100 next to block99. But, if M2 is an attacker (see Fig. 2.b) that corporates with M1, M2 ignores the invalidation of block100, signs it and creates next block, namely block101 that is valid. Then, the next masternode M3 verifies that block101 is valid, M3 signs block101 and creates a block102. By this way, Single Validation potentially leaves the blockchain with "garbage" or invalid blocks which require a "rebase" to restore the validity of the blockchain.

- **Double Validation:** We claim that our DV technology significantly reduces the likelihood of garbage blocks in the blockchain. Suppose M1 and M2 are the block creator and block validator for block 100 in our DV, respectively. If block100 is invalid and M2 is honest (see Figure 3.a), M2 will not seal the block. So the next block creator M3 that creates block101 will see that block100 doesn't have enough 2 signatures, so reject block100 and create another block100 next to block99. On the other hand, if M2 is also the attacker pairing/handshaking with M1 (see Figure 3.b), M2 signs block100 despite Block100 being invalid (remember that block validator M2 is randomly chosen, the chance of a successful pairing Small M1 and M2). Next, even though M3 will verify that block100 has two valid signatures, M3 still rejects it because block100 is invalidated by M3, which will create another valid block100. In this case, in order to destroy the stability and consistency of the blockchain, M3 should be an attacker along with M1 and M2, but the probability is very low. In other words, DV strengthens the consistency of the blockchain, making it difficult to crack.

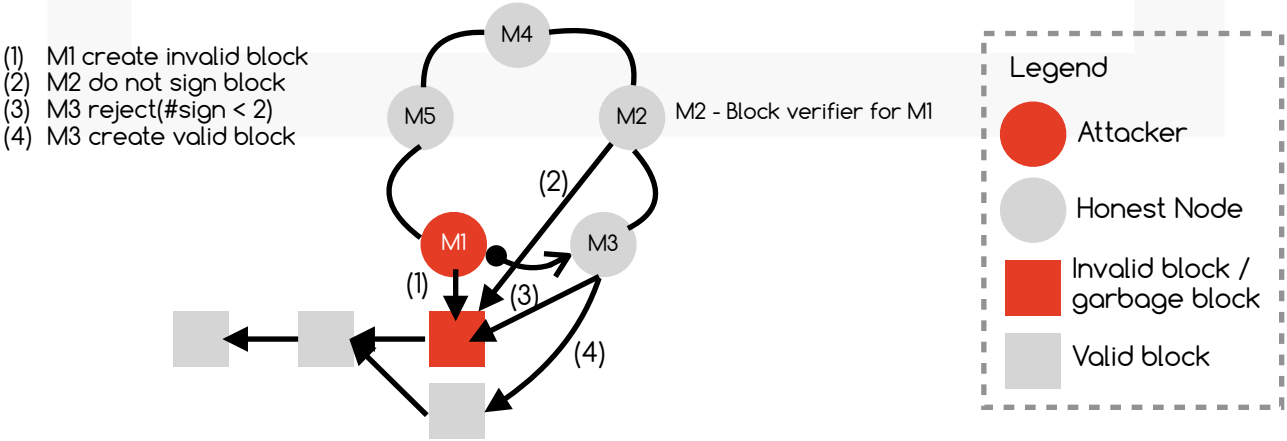

Fig. 3.a Double Validation (DV): DV with block creator as an attacker

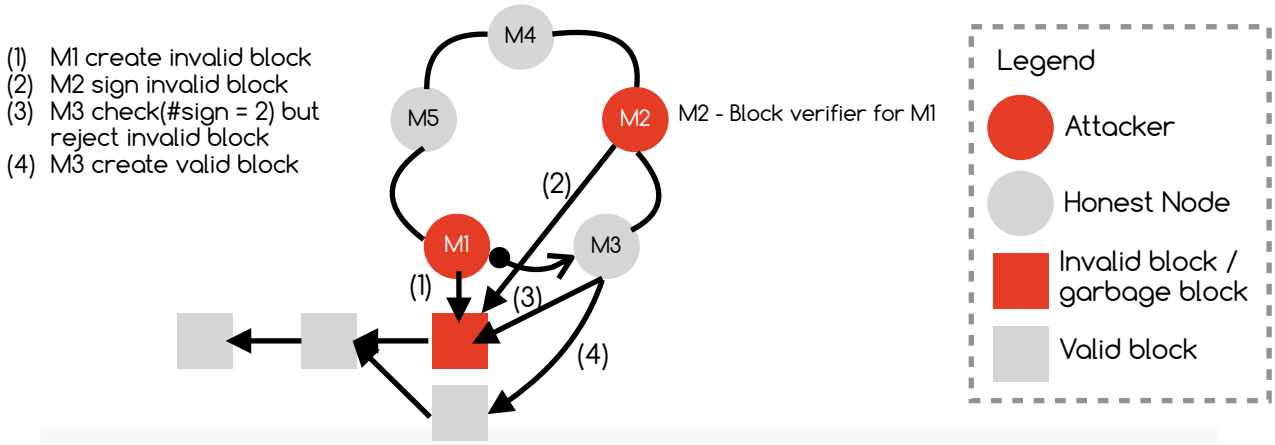

Fig. 3.b Double Validation (DV): DV with both block creator and block verifier as attackers

Randomisation for Block Verifiers for Double Validation
First masternode/block creator: The first masternode/block creator in a given epoch e can be selected by turn game and can be formally defined as an array:

$$\left[\nu_1\right] = \begin{bmatrix} V^e_{1.1} \\ V^e_{1.2} \\ . \\ . \\ . \\ V^e_{1.n-1} \\ V^e_{1.n} \end{bmatrix}$$

Random Matrix and Smart Contract: Let m be the number of masternodes and n be the number of slots in an epoch. To randomly generate block validators for the next epoch e+1, the process is performed through the following steps.

- **Step 1: Random Numbers Generation and Commitment Phase:**
  First, at the beginning of epoch e, each masternode V_i will securely create an array of n + 1 special random numbers $Recommend_i = [r_{i.1}, r_{i.2}, ..., r_{i.n}, \theta_i]$, where $r_{i.k} \in [1, ..., m]$ indicating the recommendation of ordered list of block verifiers for the next epoch of V_i, and $\theta_i \in \{-1, 0, 1\}$ is used for increasing the unpredictability of the random numbers. Second, each masternode V_i has to encrypt the array $Recommend_i$ using a secret key SK_i, say $Secret_i = Encrypt(Recommend_i, SK_i)$ as the encrypted array. Next, each masternode forms a "lock" message that contains encrypted array $Secret_i$; signs off this message with its blockchain's private key through the Elliptic Curve Digital Signature Algorithm (ECDSA) scheme currently used in Ethereum and Bitcoin along with the corresponding epoch index and its public key generated from its private key. By doing this, every masternode can check who created this lock message through ECDSA verification scheme and which epoch it relates to. Then, each node V_i sends their lock message

with its signature and public key to a Smart contract stored in the blockchain, so that eventually each masternode collects and knows the locks from all other masternodes.

- **Step 2: Recovery Phase:** The recovery phase is for each node to display its previous lock message so that other nodes can know the secret array it previously sent. Masternodes will only start showing their lock messages when all masternodes have sent their lock messages to the smart contract or some timeout event occurs. Each masternode then opens its lock message by sending an "unlock" message to the smart contract so that other masternodes can open the corresponding locks. Imagine a promise-like scheme, in which case the lock message is a promise message that locks the recommended array Recommendation_i it contains (so that no one can open or guess the contained array), and the unlock message provides other masternodes decrypted key box and retrieve the value of Recommend_i. Ultimately, the masternode has both the locks and unlocks of others. If an elector is an adversary and may issue its own lock, but not intend to send the corresponding unlock, other masternodes can ignore the adversary's lock and set all their random values to 1 by default. The idea is simple: the network can continue to function successfully even if some masternodes are adversaries.

- **Step 3: Assembled Matrix and Computation Phase:** At the point of the slot n^{th} of the epoch e, the secret arrays Secret_i in the smart contract will be decrypted by each masternode and return the plain version of Recommend_i. Each tuple of the first n numbers of each V_i will be assembled as the i^{th} column of an n × m matrix. All the last number θ_i forms a m × 1 matrix. Then each nodes will compute the block verifiers ordered list by some mathematical operations as explained below. The resulting output is a matrix n × 1 indicating the order of block verifiers for the next epoch e+1.

The Second Masternode/Block Verifier: Then, each node soon compute the common array v_2 for the order of the block verifiers by the following steps as in Equation 1.

$$\left[\nu_2'\right] = \begin{bmatrix} v_{2.1}^{e+1} \\ v_{2.2}^{e+1} \\ \vdots \\ v_{2.n}^{e+1} \end{bmatrix} = \begin{bmatrix} r_{1.1} & r_{2.1} & \cdots & r_{m.1} \\ r_{1.2} & r_{2.2} & \ddots & \vdots \\ r_{1.3} & \ddots & \ddots & r_{m.3} \\ \vdots & & r_{m-1.n-1} & r_{m.n-1} \\ r_{1.n} & \cdots & r_{m-1.n} & r_{m.n} \end{bmatrix} \begin{bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \vdots \\ \theta_m \end{bmatrix} \tag{1}$$

$$\left[\nu_2\right] = \left[\nu_2' \mod m\right] = \begin{bmatrix} \left|v_{2.1}^{e+1}\right| \mod m \\ \left|v_{2.2}^{e+1}\right| \mod m \\ \vdots \\ \left|v_{2.n}^{e+1}\right| \mod m \end{bmatrix} \tag{2}$$

Then, v_2 is obtained by modulo operation of element values of v_2' as in Equation 2:

Finality Analysis

There is a standard definition of "total economic finality": this happens when ¾ of all masternodes make a maximum odds bet that a given block or state will be finalized. This is the case Provides a very strong incentive for masternodes to never try to collude to recover a block: once a masternode makes such a max odds bet, in any blockchain where the block or state does not exist, the masternode will lose their entire savings."

PGChain maintains this standardization in its design, so if a block collects ¾ signatures of all masternode committees, it is considered irreversible. The timeline of the blockchain creation process, checking for finality, and marking blocks as immutable is shown in Figure 4 below.

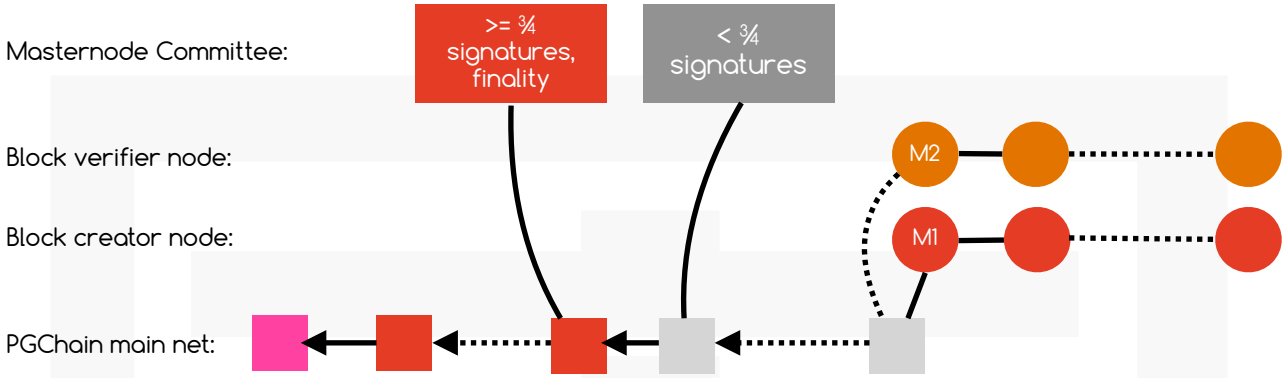

Fig. 4. Timeline of Blockchain Making Process

D. Consensus Protocol: Formalization

Basic Concepts & Protocol Description

In order to build a solid foundation for us to demonstrate that our blockchain can achieve what it claims, we first introduce a preliminary formalization of the concepts we will use later in the Yellow Paper. First, when we deal with proof-of-stake consensus algorithms, we follow formalizations in recent literature, such as Cardano and Thunder Token. In particular, we review the following concepts and definitions proposed in PGChain and adapt them to the context of PGChain.

Time, Slots, Epoch

As mentioned earlier, each epoch is ideally divided into 900 block times, called block slots. Only one block can be created in a slot. We assume that there is a roughly synchronized clock that allows the master to learn the current slot. This simplification would effectively allow masternodes to perform the signing and verification process of PoSV consensus, where each masternode must collectively create a block for the current slot. For more simplification, each slot sl_r is accessed by an integer $r \in \{1, 2, ...\}$, and suppose that the real time window that corresponds to each slot has the following properties, which are similar to what are specified in Cardano.

1. Every masternode can determine the index of the current slot based on the current time and "any discrepancies between parties' local time are insignificant in comparison with the length of time represented by a slot".

2. The amount of a slot time is sufficient to guarantee that any message transmitted by an honest party at the beginning of the time window will be received by any other honest party by the end of that time window. While this assumption similar to Cardano, PGChain requires it in order for a block creator to propagate its created block to the corresponding block verifier to ensure that the block is signed by both the masternodes before the next block creator builds another block on top of it.

As mentioned in Section II-A, in our setting, we assume that the fixed set of m (21) masternodes V_1, V_2, ..., V_m interact throughout the protocol to reach the consensus. For each V_i a public/private key pair (pk_i,sk_i) for a prescribed signature scheme, ideally ECDSA, is generated. Furthermore, we assume that the public keys pk_1,...,pk_m of the masternodes are distributed and known by all of them (that means a masternode knows all public keys of other nodes). Some notable definitions of the blockchain concepts are defined following the notation of [11].

Definition 1 (State): A state [6] is an encoded string $st \in \{0, 1\}^{\lambda}$. 11

Definition 2 (Block): A block [6] B generated at a slot sl_i contains the current state $st \in \{0, 1\}^{\lambda}$, data $d \in \{0, 1\}^{*}$, the slot number i and a signature $\Sigma = Sign_{ski}(st, d, sl_i)$ computed under sk_i corresponding to the masternode V_i generating the block.

Algorithm 1: Algorithm illustrated the consensus protocol

Input: m - Number of masternodes, n number of slots in an epoch

Output: The ledger of the blockchain C

begin

- Create the empty blockchain (stack) C;
- Initiate ICO; coinholders;
- Voting for the masternode committee (master nodes) $VC \leftarrow \{V_1; V_2; ..., V_m\}$;
- Initiate the first epoch $e_1 \leftarrow \{sl_1, sl_2, ..., sl_n\}$;
- Randomly generate the array of second masternodes for the first epoch $SV_1 \leftarrow [v^1_{2.1}, v^1_{2.2}, ..., v^1_{2.n}]$;
- Create the genesis block B_0;
- Update the blockchain $C \leftarrow C.push(B_0)$;
- **while** true **do**
  - **while** j is less than n **do**
    - Create block B_j by the first masternode;
    - Update the blockchain $C \leftarrow C.push(B_j)$;
    - Validate the block B_j by the second masternode;
    - Broadcast and validate the block B_j by VC_i;
    - **if** B_j has more than 3/4 masternode committee members sign **then**
      - FINALITY($B_j.ID$) = true;
    - **if** j = n **then**
      - $j \leftarrow 1$;
    - **else**
      - j++;
  - **if** len(C) mod n = 0 **then**
    - doCheckpoint();
    - Voting for the masternode committee for the next epoch $VC \leftarrow \{V_1; V_2; ..., V_m\}$;
    - Random generate the array of verifier masternodes for the next epoch $(i + 1)^{th}$;
    - $SV_{i+1} \leftarrow [v^{i+1}_{2.1}, v^{i+1}_{2.2}, ..., v^{i+1}_{2.n}]$;
    - $e_{i+1} \leftarrow i * n * 2 + e_1$;
    - i++;

Definition 3 (Blockchain): A blockchain [6] C is a sequence of blocks B_1 , ..., B_n associated with a strictly increasing sequence of slots for which the state st_i of B_i is equal to $H(B_{i-1})$, where H is a collision-resistant cryptography hash function. A blockchain has a number of properties, including the length of a chain len(C) = n, which is its number of blocks, and the block B_n is the head of the chain, denoted head(C).

As mentioned earlier, in our PGChain model, we set each time slot sl_i as 2 seconds; an epoch is a set R of 900 slots {sl_1, sl_2, ..., sl_{900}} (an epoch time duration equals to 1800 seconds).

To sum up, the consensus protocol of PGChain can be formalised in Algorithm 1. Algorithm 1 is simulated and interpreted as the process shown in Figure 5.

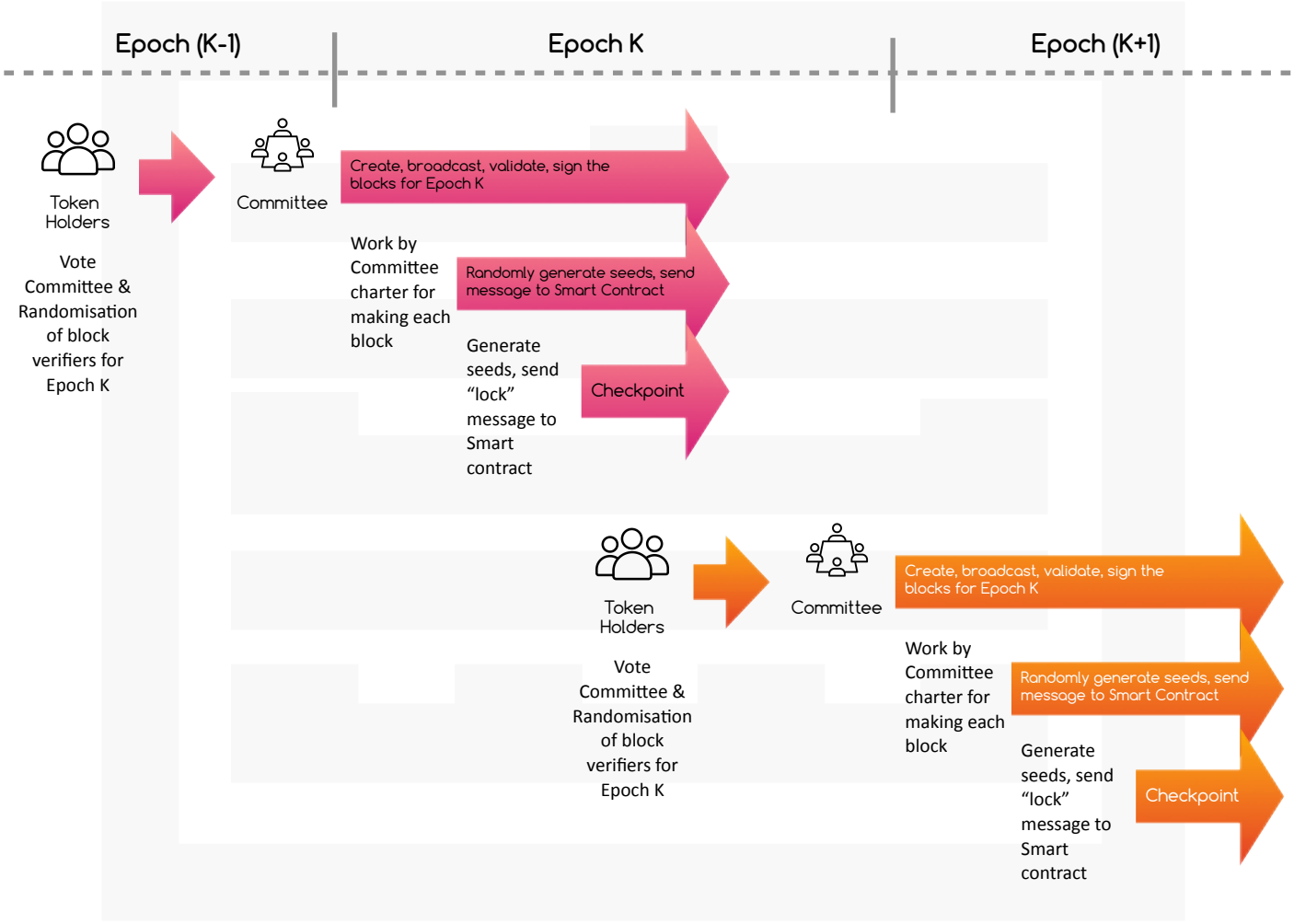

Fig. 5. Process of Voting Committee, Randomisation of Block Verifiers, Creating and Validating Blocks in Each Epoch

III. SECURITY ANALYSIS

Nothing-at-stake

Nothing-at-stake is a well-known problem in PoS-based blockchains, like 51% attacks in PoW algorithms. PoW-based miners require capital expenditures (capex) to purchase mining equipment such as ASICs and operational expenditures (opex) such as electricity to solve the mathematical puzzle of securing the network. This means that, regardless of its success, there is always an inherent cost to miners in mining. Therefore, in the case of a fork, miners always allocate their resources (equipment) to what they think is the correct chain to gain an incentive to compensate for the inherent cost of mining.

Conversely, in a PoS-based system without mining, in an ideal execution process, in order to create a fork-only cost, a masternode would incur practically no intrinsic cost, other than about some block verification and signing costs. As a result, masternodes have an inherent problem of having no downsides on both forks. So there are actually two problems with the original design of PoS. On the one hand, for any masternode, the best strategy is to validate every chain/fork so that the masternode is rewarded no matter which fork wins. On the other hand, for attackers/malicious masternodes, they can easily create a double-spend fork.

Let's review how PGChain handles these two problems. As a reminder, PGChain maintains a certain order of masternodes when creating and sealing blocks in each epoch. For the first problem, random/arbitrary forks almost never happen because the order in which the masternodes are created by the blocks of each epoch is predetermined. Furthermore, the double verification mechanism eliminates the second problem, because even if a malicious masternode has his turn to create two blocks, only one block can be verified by a second randomly chosen masternode.

Long-range attack

In PGChain, blocks are only valid when two-factor verification is collected and finalized after ¾ master node verification. Therefore, as long as the number of attackers or malicious nodes and/or fail-stop nodes is less than ¼ the number of masternodes, the number of masternodes signing the block is at least ¾ of the total number of masternodes, which makes the block complete. Therefore, a malicious masternode has no chance to create a longer valid chain because other masternodes will reject it.

Censorship Attack

If there are more than ¾ malicious masternodes in the PGChain, a censorship attack may occur. For example, these masternodes reject valid blocks or simply become inactive. In this case, the chain is stuck.

In fact, the correct job of masternodes is for them to put in the effort to proactively update the chain in a consistent manner. More importantly, being a masternode means locking up a certain amount of coins, specifically 30,000 PGC. Therefore, in order to control more than ¾ masternodes, the attacker must hold a considerable amount of PGC and gain huge support from the coin holders. Because of this, attackers have no incentive to do anything malicious to compromise the chain.

However, in the worst case, PGChain has to do a soft fork to reduce the number of masternodes to keep the chain running and figure out a slasher mechanism for these malicious masternodes.

Relay Attack

PGChain supports EIP155 (https://github.com/ethereum/EIPs/blob/master/EIPS/eip-155.md). Transactions in PGChain are included CHAIN ID specified for different public chains. Table I shows recognized CHAIN IDs.

TABLE I: CHAINS AND CHAIN ID

Chain ID	Chain
1	Ethereum mainnet
2	Morden (disused), Expanse mainnet
3	Ropsten
4	Rinkeby
30	Rootstock mainnet
31	Rockstock testnet
42	Kovan
61	Ethereum Classic mainnet
62	Ethereum Classic testnet
1337	Geth private chains (default)
77	Sokol, the public POA Network testnet
99	Core, the public POA Network main network
1688288	PGChain mainnet
1688388	PGChain testnet

Safety and liveness

Security means having an agreed-upon chain where there are no two or more competing chains with valid transactions. Consensus protocols may be secure when blocks have settlement finality or probabilistic finality. The last sentence shows that PGChain can provide security because of its settlement finality.

A consensus protocol is considered valid if it can eventually propagate and make valid transactions on the blockchain. Liveness failures occur when transaction omissions, information withholding, or message reordering are observed in many breaches. This type of failure is unlikely to happen in PGChain because the block creation masternode list for each epoch is ordered in a pre-determined way, so even if the attacking masternode omits some transactions, the latter will be replaced by the next Honest masternodes process and validate on the next block.

DDOS Attack

Masternodes are encouraged to run in well-known public cloud providers that offer multiple DDOS protection mechanisms, such as AWS, Google Cloud, or Microsoft Azure. Even in the event that some nodes are attacked or the failure stops, as long as the number of failed and/or attacked nodes is less than ¼ of the number of master nodes, the network can still work normally.

Spam Attack

PGChain maintains the same transaction fee mechanism as Ethereum, represented by gasPrice. However, PGChain supports a minimum transaction fee (1 wei), which somehow makes an attacker attempt to spam a large number of low-fee transactions into the system. However, PGChain masternodes always order transactions and only pull high-fee transactions into proposed blocks. Therefore, spammers have little chance of compromising the system.

IV. RELATED WORK

Consensus plays an important role in ensuring the success of distributed and decentralized systems. Bitcoin's core consensus protocol, commonly referred to as Nakamoto consensus, implements the "replicated state machine" abstraction, where nodes in a permissionless network agree on a set of submitted transactions and their ordering. However, known permissionless consensus protocols (such as Bitcoin's Nakamoto consensus) come at a cost. Bitcoin and Ethereum rely on PoW to roughly enforce the "hash one vote" philosophy and defend against Sybil attacks. Unfortunately, PoW-based Bitcoin and Ethereum are notoriously poor performers (Bitcoin's transaction processing performance peaks at around 7 transactions per second, as mentioned earlier). In addition, PoW has been criticized for consuming a lot of power.

To design an efficient and cost-effective consensus protocol in a permissionless model, PoS has been discussed extensively in Bitcoin and Ethereum forums. PoS blockchain can replace the expensive PoW in Nakamoto blockchain while still providing similar guarantees in terms of transaction processing in the presence of a dishonest minority of users, the "minority" here is in the context of stake rather than computing power Understood. The Ethereum Design Casper published by Buterin & Griffith provides as its initial version a hybrid PoW/PoS consensus protocol that may eventually switch to a pure PoS system. Like PGChain, Ethereum Casper requires validators (a term similar to block creators) to deposit a certain amount. In fact, some concepts used in PGChain, such as checkpoint blocks, are borrowed from Casper.

The Proof of Stake Voting (PoSV) consensus protocol we propose in this paper can be viewed as a hybrid model. In particular, first, we apply PoSV with voting and two-factor verification to create, verify, and vote blocks smoothly and efficiently. Whenever the possibility of a forking branch is detected, we adopt the idea in PoW to choose the longest and most voted branch, and discard the others. Through this hybrid approach, PoSV not only improves the performance and security of the blockchain, but also reduces forks in an efficient and practical way.

Furthermore, there has been some consensus protocol research work closely related to PGChain, such as EOS and Cardano's Ouroboros. Bitshares and EOS use a mechanism of voting for masternodes to reach consensus, and their consensus protocol is called Delegated Proof of Stake (DPoS). DPoS is similar to PGChain's Proof-of-Stake voting consensus, and the master node (block creator or witness in DPoS) is elected through the voting system. However, PGChain requires masternodes to deposit the required minimum number of PGCs to become candidate masternodes, which puts more pressure on masternodes to work honestly. In addition, as mentioned earlier, PGChain's two-factor verification mechanism reduces the probability of handshake attacks and invalid blocks. EOS also has a maximum of 21 block producers per epoch, which is less decentralized than PGChain, and elects a maximum of 21 masternodes (and this number of masternodes can be changed by voting according to decentralized governance).

The research-backed Cardano blockchain solution, Ouroboros, and the fully proof-of-stake-based ADA token, promises to offer strict security guarantees. Similar to PGChain, Ouroboros has a set of block producers to create blocks in each epoch, and each block producer candidate needs to deposit a minimum amount of stake (ADA amount). However, please note that Ouroboros only provides single verification, while PGChain's double verification provides several advantages over single verification, as mentioned earlier. In Ouroboros, the order of block producers selected among stakers is based on biased randomization, while PGChain's randomization of block validators may be uniform and based on smart contracts. Furthermore, the use of voting in PGChain and DPoS makes the incentives more equal among stakers: in Ouroboros, stakers with very few stakes are less likely to become block creators, while in PGChain these stakers are less likely to become block creators The best strategy can be chosen to vote for potential masternodes for rewards.

V. CONCLUSION AND PERSPECTIVES

In this paper, we propose PoSV, a blockchain protocol based on PoS voting with heuristic and fair voting mechanism, strict security guarantees, and fast finality. We also propose a novel reward mechanism and show that through this mechanism, the probability of blockchain forks is low, the confirmation time is fast, plus the contribution and benefit of the master node is fair, that is, the probability distribution function is ultimately average.

Perspectives

- **Future work:** The PGChain team is currently working on implementing a proof-of-stake vote, which will be released on schedule as described in our roadmap. Furthermore, in parallel with our novel consensus protocol, we will investigate sharding mechanisms to provide better transaction processing performance. We believe that sharding technology with a stable number of masternodes will provide better stability and efficiency for the blockchain. At the same time, we are committed to keeping EVM-compatible smart contracts in our masternode sharding framework.

- **Economic sustainability:** is also an important concept for blockchain-based decentralized networks. This means that maintaining the network in a sustainable state requires a balance where the cost of running the network infrastructure can be offset by the revenue generated. In this case, the cost of network infrastructure consists of two parts: the physical cost of owning the hardware, such as servers, memory required by the network technology; and the capital cost of locking the PGC in a smart contract. The revenue of Masternodes will mainly come from the emission of the reward engine, and then from service revenue such as token exchange fees provided by applications running on top of PGChain. We will publish PGChain economic analysis and recommendations separately from this technical paper at a later date.

REFERENCES

[1] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronics cash system. 2008.

[2] Ethereum Foundation. Ethereum's White Paper. http://github.com/ethereum/wiki/white-paper, 2014. Online available 25/05/2018.

[3] D. Larimer. Delegated Proof-of-Stake (DPOS). BitShare White Paper 2014.

[4] S. King and S. Nadal. PPCoin: Peer-to-peer crypto-currency with proof-of-stake. Self-Published, 2012.

[5] V. Buterin. On public and private blockchains. Ethereum Blog, 2015.

[6] A. Kiayias, A. Russell, B. David, and R. Oliynykov: Ouroboros: A Provably Secure Proof-of-Stake Blockchain Protocol. IACR-CRYPTO-2017.

[7] D. Mingxiao, et al. A Review on Consensus Algorithms of Blockchain. 2017 IEEE International Conference on Systems, Man, and
Cybernetics (SMC) Banff Center, Banff, Canada, October 5-8, 2017

[8] R.PassandE.Shi.RethinkingLarge-ScaleConsensus.IntheProceedingsofthelEEE30thComputerSecurityFoundationsSymposium, 2017.

[9] Thunder Token Foundation: Thunder Consensus White Paper, Janurary, 2018.

[10] R. Pass, L. Seeman, and A. Shelat. Analysis of the Blockchain Protocol in Asynchronous Networks. In EUROCRYPTO 2017.

[11] JuanA.Garay,A.Kiayias,andN.Leonardos.Thebitcoinbackboneprotocol:Analysisandapplications.InElisabethOswaldandMarc Fischlin, editors, Advances in Cryptology - EUROCRYPT 2015, Volume 9057 of Lecture Notes in Computer Science, pages 281-310.
Springer, 2015.

[12] TendermintTeam.UnderstandingtheBasicsofaProof-of-StakeSecurityModel.https://blog.cosmos.network/understanding-the-basics-
of-a-proof-of-stake-security-model-de3b3e160710. Online available 25/05/2018.

[13] V. Buterin. On Settlement Finality. https://blog.ethereum.org/2016/05/09/on-settlement-finality/. Online available 25/05/2018.

[14] EOSTeam.EOS.IOTechnicalWhitePaperv2.https://github.com/EOSIO/Documentation/blob/master/TechnicalWhitePaper.md.Online
available 25/05/2018.

[15] BitsharesTeam.DelegatedProof-of-StakeConsensus.https://bitshares.org/technology/delegated-proof-of-stake-consensus/.Onlineavail- able 25/05/2018.

[16] R. Pass, and E. Shi. (2017). Hybrid consensus: Efficient consensus in the permissionless model. In LIPIcs-Leibniz International Proceedings in Informatics (Vol. 91). Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

[17] V. Buterin, and V. Griffith. (2017). Casper the Friendly Finality Gadget. arXiv preprint arXiv:1710.09437.

[18] H. McCook. Under the Microscope: Economic and Environmental Costs of Bitcoin Mining. https://www.coindesk.com/microscope-economic-environmental-costs-bitcoin-mining/. Online available 25/05/2018.