Roll-DPoS🎲: A Randomized Delegated Proof of Stake Scheme for Scalable Blockchain-Based Internet of Things Systems

Short Paper

Xinxin Fan and Qi Chai
IoTeX
Menlo Park, CA, USA
{xinxin,raullen}@iotex.io

ABSTRACT

Delegated Proof-of-Stake (DPoS) is an efficient, decentralized, and flexible consensus framework available in the blockchain industry. However, applying DPoS to the decentralized Internet of Things (IoT) applications is quite challenging due to the nature of IoT systems such as large-scale deployments and huge amount of data. To address the unique challenge for IoT based blockchain applications, we present Roll-DPoS, a randomized delegated proof of stake algorithm. Roll-DPoS inherits all the advantages of the original DPoS consensus framework and further enhances its capability in terms of decentralization as well as extensibility to complex blockchain architectures. A number of modern cryptographic techniques have been utilized to optimize the consensus process with respect to the computational and communication overhead.

CCS CONCEPTS

• **Security and privacy → Distributed systems security**; • **Computer systems organization → Reliability**; **Availability**;

KEYWORDS

Delegated Proof-of-Stake, Randomization, Distributed Key Generation, Threshold Signature, Bilinear Pairing

1 INTRODUCTION

Delegated Proof of Stake (DPoS) is a robust and flexible blockchain consensus mechanism invented by Larimer [11] in 2014, which leverages the voting power of the stakeholder to resolve consensus issue in a fair and democratic manner. DPoS was first applied by Larimer to power the blockchain project **BitShares** [4] and further refined in his subsequent projects **Steem** [15] and **EOS** [8]. Other blockchain projects, such as **Ark** [1], **Lisk** [12], **Tezos** [16], etc., also adopted the similar DPoS framework with certain feature changes.

Unlike the traditional Proof of Stake (PoS) system that involves the participation of the entire network to validate a transaction, DPoS concentrates block production in the hands of a limited number of semi-trusted delegates. The DPoS consensus framework consists of the four major steps:

(1) **Block Producer Election**. The cryptocurrency token holders cast votes to elect an odd number of users to produce blocks. The number of votes are proportional to the voter's stake and a fixed number of top candidates that receive the most votes become the block producers.
(2) **Block Production**. The block production happens in rounds and is conducted in a round-robin manner among block producers. At the beginning of each round, the block producers are shuffled and assigned a time slot in which they should produce a block. In the case that a block producer fails to create a block during its time slot, the block is skipped and the transactions within are included in the next one.
(3) **Block Reward Distribution**. The block producer receives a block reward for successfully producing a block and spreads the (partial) block reward to its voters as dividends.
(4) **Block Producer Reelection**. A reelection process is performed at the end of each round and block producers can be voted in or out by the cryptocurrency token holders.

2 ROLL-DPOS: A RANDOMIZED DPOS CONSENSUS ALGORITHM

A high-level description of Roll-DPoS is illustrated in Figure 1. In a nutshell, Roll-DPoS initiates a candidate pool through a community voting process with the aid of the Ethereum blockchain and nodes that receive the most number of votes from the community become the potential block producers. The Roll-DPoS consensus algorithm runs in epochs and each epoch is further composed of multiple sub-epochs. At the beginning of the epoch, a fixed number of block producers are randomly chosen from the candidate pool using a cryptographic hash function as well as a random beacon that is generated from the previous epoch. The selected block producers then perform the Pedersen's DKG protocol [14] to generate short-lived, epoch-specific private key shares that will be used to sign the messages in the Practical Byzantine Fault Tolerance (PBFT) process [6] with the short-lived ECDSA and BLS threshold signature schemes. The five core components (CCs) of the Roll-DPoS design are detailed in the following subsections.

2.1 CC-I: Ethereum-Aided Bootstrapping

The Roll-DPoS protocol is bootstrapped with the aid of the Ethereum blockchain and composed of the three steps below.

MobiQuitous '18, November 5–7, 2018, New York, NY, USA

ACM ISBN 978-1-4503-6093-7/18/11…$15.00

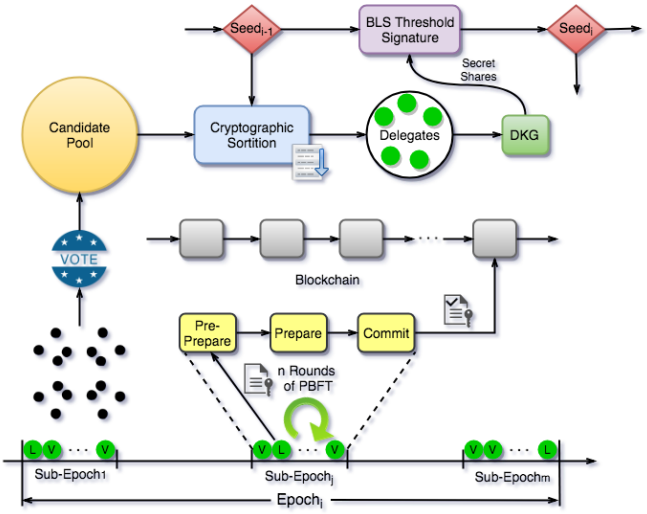

Figure 1: A High-Level Overview of the Roll-DPoS Consensus Algorithm

2.1.1 Step 1: Block Producer Self-Nomination. In Step 1, any miner can self-nominate in the community and register to become a potential block producer. During this phase, a miner usually sets up a campaign website (see [7] for an example) for attracting community members to vote for him/her, which specifies the software and hardware resources available for hosting block producer servers as well as the terms for block reward distribution, among many other things. The self-nomination process must be completed before all the community members are able to start the voting process.

2.1.2 Step 2: Block Producer Candidate Pool Formation. In Step 2, the community members vote for an initial block producer candidate pool of N nodes by sending special Ethereum voting transactions that contain '0' in the VALUE field and 'vote' in the DATA field. Each transaction indicates that a community member (i.e., an ERC-20 token wallet holder) has pledged backing towards a candidate and all the ERC-20 tokens in the wallet will be tallied at the end of a pre-defined cut-off time. At that moment, the Ethereum token contract freezes all the accounts and takes a snapshot of the beginning balances for all the ERC-20 token holders. Moreover, a vote counting process is executed and the highest-backed set of N candidates, denoted by $\{C_1^{(1)}, \ldots, C_N^{(1)}\}$, are selected to form the initial block producer candidate pool.

2.1.3 Step 3: Block Producer Selection and Ordering. In Step 3, n block producers, denoted by $\{BP_1^{(1)}, \ldots, BP_n^{(1)}\}$, are sequentially chosen from the candidate pool as the bootstrapping nodes for the first epoch using a deterministic random bit generator (DRBG) [3]. DRBG can be efficiently realized with a block cipher or cryptographic hash function (see [2] for an example) and a nothing-up-my-sleeve number, e.g., the hash of the string "IoTeX", is used as the initial seed s_0 for the DRBG. Let $\{V_1^{(1)}, \ldots, V_N^{(1)}\}$ be the number of votes received by the miners in the candidate pool $\{C_1^{(1)}, \ldots, C_N^{(1)}\}$, respectively. $V^{(1)} = \sum_{i=1}^{N} V_i^{(1)}$ denotes the initial total number of votes and L is its bit length. The n bootstrapping nodes are then selected in a distributed manner. More specifically, each candidate initializes the DRBG as $\mathbf{DRBG}(s_0, pk_1^{(1)}, \ldots, pk_N^{(1)}, 1)$, where $pk_i^{(1)}$ is the public key of the candidate $C_i^{(1)}$ and '1' is the epoch number of the first epoch. Each candidate continues generating L-bit random numbers R's until n block producers $\{BP_1^{(1)}, \ldots, BP_n^{(1)}\}$ are selected. The candidate $C_j^{(1)}$ is chosen as the block producer if and only if

$$R \bmod V^{(1)} \in \begin{cases} \left(0, V_1^{(1)}\right] & \text{if } j = 1 \\ \left(\sum_{i=1}^{j-1} V_i^{(1)}, \sum_{i=1}^{j} V_i^{(1)}\right] & \text{if } j > 1 \end{cases}.$$

Once n block producers $\{BP_1^{(1)}, \ldots, BP_n^{(1)}\}$ are selected, they are going to propose blocks sequentially in the first epoch.

2.2 CC-II: Blockchain-Aided Distributed Key Generation

The Roll-DPoS protocol takes advantage of two short-lived signature schemes[1] (i.e., ECDSA and BLS threshold signature) coupled with the Pedersen's DKG mechanism to accelerate the PBFT process and generate random beacons for each epoch, respectively. To utilize short-lived signatures in each epoch, n block producers jointly perform the Pedersen's DKG scheme to generate short-length (i.e., 158-bit) key shares of an unknown group private key, which results in two short-lived private/public key pairs on the MNT curve for each block producer. While the ECDSA key pair is used to sign/verify block proposals and other messages during the PBFT process (see Section 2.3), the BLS key pair is utilized to generate a random beacon for the next epoch using the threshold signature scheme (see Section 2.4). Moreover, to further amortize the cost of DKG, an epoch has been divided into m sub-epochs and the DKG scheme is only executed once per epoch.

The Pedersen's DKG scheme, while enabling network entities to generate a group key in a distributed manner, is quite involved. In particular, due to the potential Byzantine behaviors of network nodes, it is challenging for the block producers to achieve the consistent view regarding to the qualified entities in the DKG process. To address this issue, we slightly modify the DKG scheme by allowing the block producers in the previous epoch to assist those in the next epoch to complete DKG. We assume that a random beacon for the epoch $i+1$ has been generated at the $(j-1)$-th sub-epoch of the epoch i (see Section 2.4). Therefore, the block producers for the epoch $i+1$ are determined and the DKG process is then kicked off. After exchanging the partial secret shares and witnesses within the block producer group, each block producer of the epoch $i+1$ verifies the validity of the received shares and broadcasts a special transaction that contains a $(0,1)$-vector recording the status of each share. The block producers of the epoch i, upon receiving those special transactions, will add them to the blockchain in the subsequent sub-epochs. This process continues until the end of the epoch i and all the block producers of the epoch $i+1$ have the same view regarding to the status of all the partial secret shares.

[1] Both short-lived signature schemes are implemented using a Miyaji-Nakabayashi-Takano (MNT) curve [13] with embedding degree 6 which offers an approximate security level of 70-bit.

2.3 CC-III: PBFT Consensus with Short-Lived ECDSA Signatures

The Practical Byzantine Fault Tolerance (PBFT) algorithm, proposed by Castro and Liskov [6], is employed in Roll-DPoS as a core component to reach consensus and instant finality for block proposals in each epoch. PBFT runs in rounds and each round consists of three phases. In the *Pre-prepare Phase*, a leader initiates the consensus process by sending a signed PRE-PREPARE message that is a block proposal containing a certain number of transactions, which is followed by the *Prepare Phase* where each node in the consensus group checks the correctness and validity of the block and multicasts a signed PREPARE message (i.e., 'YES/NO') to all the other nodes. Finally, each node analyzes the received PREPARE messages and multicasts a signed COMMIT message (i.e., 'YES/NO') to the consensus group in the *Commit Phase*. The block proposal is committed to the blockchain only if a sufficient number of nodes in the consensus group agree on it.

As pointed out in [9], both the *Prepare* and *Commit* phases involve intensive interactions among block producers within the consensus group in PBFT. More specifically, given that a consensus group of n block producers can tolerate $f = \lfloor (n-1)/3 \rfloor$ Byzantine nodes in PBFT, each honest node needs to collect and verify $2f + 1$ digital signatures for both phases. Using short-lived ECDSA signatures with short-length cryptographic keys can not only accelerate the verification process of multiple signatures but also reduce the signature size simultaneously, thereby further improving the performance of PBFT. In Roll-DPoS, each block producer obtains a 158-bit ECDSA key pair on the MNT curve upon completion of the Pedersen's DKG scheme (see Section 2.2) at the beginning of each epoch. For the subsequent m sub-epochs, those short-length ECDSA keys are utilized to sign the block proposals as well as the PREPARE and COMMIT messages across the three phases of PBFT.

2.4 CC-IV: Random Beacon Generation with Short-Lived BLS Threshold Signature

In Roll-DPoS, a random beacon is generated using the short-lived, non-interactive BLS threshold signature scheme [5] in each epoch, which is motivated by the similar approach proposed in **DIFINITY** [10]. More specifically, we assume that a block producer i obtains a 158-bit BLS key share $sk_i^{(j)}$ on the MNT curve through the Pedersen's DKG scheme at the beginning of the j^{th} epoch. Besides proposing blocks in the j^{th} epoch, the block producer i also uses the **SignShareGen**$(\cdot)$ algorithm to generate a random beacon share $s_j^{(i)} = \textbf{SignShareGen}(sk_i^{(j)}, s_{j-1} \| j)$, where s_{j-1} is the random beacon of the previous epoch. $s_j^{(i)}$ is then included in the 'extra data' field of the block proposal by the block producer. If the block proposal passes the PBFT consensus successfully, $s_j^{(i)}$ will be available on the blockchain. Otherwise, $s_j^{(i)}$ is going to be added into a new block proposal in the next sub-epoch by the block producer i.

Once the number of random beacon shares exceeds the predefined threshold, every node in the candidate pool is able to compute and verify s_j using the **SignShareCombine**$(\cdot)$ and **Verify**$(\cdot)$ algorithms, respectively. The random beacon is then used as the seed to select block producers for the next epoch with the same process as detailed in Section 2.1.3. In the case that the number of random beacon shares contained in the blockchain is less than the threshold at the end of the epoch, the random beacon will be updated pseudorandomly, i.e., $s_j = H(s_{j-1} \| j)$, where $H(\cdot)$ is a cryptographic hash function.

2.5 CC-V: Auto-Scaling Candidate Pool for Complex Blockchain Architectures

For supporting complex blockchain architectures (e.g., root chain and sidechians) as well as large-scale IoT DApps, we propose to use an auto-scaling candidate pool for powering consensus process for sidechains in Roll-DPoS. The basic idea is to dynamically adjust the size of the candidate pool based on the number of sidechains running simultaneously in the system. In our design, the blockchain nodes in the candidate pool has an option to become the block producers for subchains if they are not being selected as the block producers for the root chain at the current epoch. As a result, the candidate pool is divided into multiple subgroups, one of which acts as block producers for the root chain and others for different subchains. Whenever the number of idle nodes (i.e., those nodes are not block producers for either root chain or sidechains) is larger than (resp. less than) a predefined threshold, a certain number of blockchain nodes will be evicted from (resp. filled into) the candidate pool. The auto-scaling candidate pool is able to power large-scale blockchain-based IoT systems by enabling a significant number of nodes participating in the consensus process for different applications. Our novel design not only solves the scalability issues of sidechains, but also enables more blockchain nodes to become block producers and receive rewards.

REFERENCES

[1] ARK. 2018. All-in-One Blockchain Solutions. https://ark.io/
[2] J-P. Aumasson, S. Neves, Wilcox-O'Hearn, and C. Winnerlein. 2016. BLAKE2X. https://blake2.net/blake2x.pdf
[3] E. Barker and J. Kelsey. 2016. *Recommendation for Random Number Generation Using Deterministic Random Bit Generators.* National Institute of Standards and Technology.
[4] BitShares. 2018. https://bitshares.org/
[5] Dan Boneh, Craig Gentry, Ben Lynn, and Hovav Shacham. 2003. Aggregate and Verifiably Encrypted Signatures from Bilinear Maps. In *Proceedings of the 22Nd International Conference on Theory and Applications of Cryptographic Techniques (EUROCRYPT'03)*. Springer-Verlag, Berlin, Heidelberg, 416–432.
[6] M. Castro and B. Liskov. 2002. Practical Byzantine Fault Tolerance and Proactive Recovery. *ACM Trans. Comput. Syst.* 20, 4 (Nov. 2002), 398–461.
[7] cc001. 2018. Professional Lisk Delegate. http://www.liskdelegate.io/
[8] Eos. 2018. https://eos.io/
[9] X. Fan. 2018. Scalable Practical Byzantine Fault Tolerance with Short-Lived Signature Schemes. In *Proceedings of the 28th Annual International Conference on Computer Science and Software Engineering (CASCON'18)*. IBM Corp.
[10] T. Hanke, M. Movahedi, and D. Williams. 2018. DIFINITY Technology Overview Series – Consensus System (Rev. 1). https://dfinity.org/pdf-viewer/pdfs/viewer?file=../library/dfinity-consensus.pdf
[11] D. Larimer. 2017. DPOS Consensus Algorithm – The Missing White Paper. https://steemit.com/dpos/@dantheman/dpos-consensus-algorithm-this-missing-white-paper
[12] Lisk. 2018. Access the Power of Blockchain. https://lisk.io/
[13] A. Miyaji, M. Nakabayashi, and S. Takano. 2011. New Explicit Conditions of Elliptic Curve Traces for FR-Reduction. *IEICE Trans. Fundamentals* E84-A, 5 (May 2011), 1234–1243.
[14] T. Pedersen. 1991. A Threshold Cryptosystem Without a Trusted Party. In *Proceedings of the 10th Annual International Conference on Theory and Application of Cryptographic Techniques (EUROCRYPT'91)*. Springer-Verlag, Berlin, Heidelberg, 522–526. http://dl.acm.org/citation.cfm?id=1754868.1754929
[15] Steem. 2018. https://steem.io/
[16] Tezos. 2018. https://tezos.com/