VIP-193: COMMITTEE-BASED POA

Ziheng (Peter) Zhou
VeChain Foundation
peter.zhou@vechain.com

Zhijie Ren
VeChain Foundation
zhijie.ren@vechain.com

March 3, 2020

1 Overview

The Proof-of-Authority consensus algorithm [1], or PoA in short, is efficient of using network bandwidth. It divides time into rounds with a fixed length and assumes that the majority of its (authorized) nodes perform consensus in the same round. In each round, nodes select a leader (the node responsible for generating a new block) based on the round number, block height and their local views of the active nodes. Therefore, the procedure can be considered instant, which allows more time for transmitting transaction (TX) data in each consensus round.

However, PoA cannot prevent a malicious leader from causing temporary inconsistency by producing multiple blocks. To improve the security of PoA, we propose to introduce a committee (a group of selected nodes) to endorse the new block generated in each consensus round. The verifiable random function (VRF) [2] is used for committee selection in each round. With the committee mechanism, a malicious leader would have to collude with committee members to cause inconsistency. However, the property of VRF guarantees that the committee is selected randomly. Therefore, it makes it much more difficult for adversaries to launch such attacks.

It is assumed that the network is synchronous and the messages are transmitted through gossip communication, i.e., if an honest node sends a message or an honest node receives a message, all other honest nodes will receive same message within at most a delay of τ, which is determined by the network configuration and is known in advance.

2 Specifications

2.1 Notations

Symbol	Description
N	Total number of nodes
$u = 1, 2, \ldots, N$	node index
$H(\cdot)$	Cryptographic hash function
$\texttt{SIG}_u(\cdot)$	Signature of u
$\rho\left(\cdot\right)$	Merkle root of the input set
r	Consensus round number
$l_r = 1, 2, ..., N$	Leader in round r
Σ_r	Transaction set prepared by l_r

s_r	Block summary produced by l_r
x_r	Tx-set package produced by l_r
$e_{u,r}$	Endorsement produced by u
d	Number of endorsements required to produce a valid block
$\pi_{u,r}$	VRF proof produced by u
h_r	Block header produced by l_r

2.2 Producing a New Block

Algorithm 1 describes the procedure for leader l_r to publish a new block in round r. During the process, nodes needs to deal with four types of packages:

- Block summary s_r
- Tx-set package x_r
- Endorsement $e_{u,r}$
- Block header h_r

They will be described in detail in later sections.

Algorithm 1 Procedure for l_r to publish a new block.

```
Prepare and broadcast $s_r$
Prepare and broadcast $x_r$ if there is any tx included in the new block
Wait for $d$ valid endorsements
Prepare and broadcast $h_r$
```

2.3 Block Summary

At the beginning of round r, leader l_r computes a summary of the proposed new block, s_r, and broadcast it for the committee members to endorse. A block summary must include the following information:

- Parent header reference
- Merkle root of the transactions included in the new block, $\rho(\Sigma_r)$
- Round number, r
- Leader's signature, $\texttt{SIG}_{l_r}(\bar{s}_r)$ where $\bar{s}_r$ represents the block summary without a signature

Algorithm 2 shows how to validate an incoming block summary.

Algorithm 2 Procedure for u to validate s_r.

```
Verify $\texttt{Sig}_{l_r}(\bar{s}_r)$
Verify that $r$ is the current round
Verify that $l_r$ is the round leader
Verify that the parent block referred by $s_r$ is consistent with the latest block on the canonical chain
```

2.4 Tx-Set Package

After broadcasting s_r, leader l_r needs to prepare and broadcast the tx-set package x_r. A tx-set package must include the following information:

- Round number, r
- Set of txs included in the new block, Σ_r
- Leader's signature, $\texttt{SIG}_{l_r}(\bar{x}_r)$ where $\bar{x}_r$ represents the tx-set package without a signature

Algorithm 3 shows how to validate an incoming tx-set package.

Algorithm 3 Procedure for validating x_r.

```
Verify $\texttt{SIG}_{l_r}(\bar{x}_r)$
Verify that $r$ is the current round
Verify that $l_r$ is the round leader
```

2.5 Endorsement

After receiving and verifying s_r, nodes u that are selected as the committee members need to prepare and broadcast endorsements $e_{u,r}$. An endorsement $e_{u,r}$ must include the following information:

- Received and verified block summary, b_r
- VRF proof of u's committee membership, $\pi_{u,r}$
- Signature, $\texttt{SIG}_{l_r}(\bar{e}_{u,r})$ where $\bar{e}_{u,r}$ represents the endorsement without a signature

Note that it is assumed that the private key used by u to generate $\pi_{u,r}$ is different from the one that generates u's address. It is also assumed that the corresponding public key that will be used by other nodes to verify VRF proof can be assessed so that it does not needs to be transmitted with $e_{u,r}$.

Algorithm 4 shows how to validate an incoming endorsement by u.

Algorithm 4 Procedure for validating u's endorsement.

```
Verify $\texttt{SIG}_u(\tau_r)$
Verify $s_r$ by Algorithm 2
Verify $u$'s committee membership by Algorithm 8
```

2.6 Block Header

After receiving d valid endorsements, leader l_r needs to prepare and broadcast the header of the new block. A block header h_r must contain the following information:

- Parent header reference
- Round number, r
- Merkle tree root of the tx set, $\rho(\Sigma_r)$
- Leader's signature on the block summary, $\texttt{Sig}_{l_r}(s_r)$
- VRF proofs of the committee membership from d nodes, $\{\pi_{u_i,r}\}_{i=1}^{d}$
- Endorsement signatures from the same d nodes, $\{\texttt{SIG}_{u_i}(\bar{e}_{u,r})\}_{i=1}^{d}$
- $\forall\, 1 \leq i < j \leq N$, $\pi_{u_i,r} < \pi_{u,j}$. It is made to reduce the number of combinations l_r can try when arranging the vrf proofs.
- Leader's signature, $\texttt{Sig}_{l_r}(\bar{h}_r)$ where $\bar{h}_r$ represents the header without a signature

Algorithm 5 describes how to validate a block header generated in round r.

Algorithm 5 Procedure for validating the header of the new block generated in round r.

```
Verify $\texttt{Sig}_{l_r}(\bar{h}_r)$
Verify that $r$ is the current round
Verify that $l_r$ is the round leader
Reconstruct and verify $s_r$ by Algorithm 2
for $i = 1, 2, \ldots, d$ do
    if $i > 1$ then
        Verify that $\pi_{u,i-1} < \pi_{u,i}$
    end if
    Reconstruct and Verify $e_{u_i,r}$ by Algorithm 4
end for
```

2.7 Block Assembling

To use the network bandwidth more efficient, the round leader does not broadcast the whole block, but broadcast the tx-set x_r and block header h_r asynchronously. Other nodes need to assemble the new block locally following Algorithm 6.

Algorithm 6 Procedure for verify and assemble the new block generated in round r.

```
Verify $h_r$ by Algorithm 5
Verify $x_r$ by Algorithm 3
Verify that $\rho(\Sigma_r)$ in $h_r$ is consistent with $x_r$
Verify txs included in $x_r$
Assemble the new block
```

2.8 Committee Membership

The verifiable random function can be considered a public-key version of keyed cryptographic hash. The hash of a given message can only be computed by the owner of the private key $\texttt{SK}$ and can be verified by anyone given the public key $\texttt{SK}$ and message $\texttt{M}$. The scheme defines the following functions:

- $\pi \leftarrow \Pi_{vrf}(\texttt{SK}, \texttt{M})$ - function that generates VRF proof;
- $\{\texttt{TRUE}, \texttt{FALSE}\} \leftarrow V_{vrf}(\texttt{SK}, \texttt{M}, \pi)$ - function that verify VRF proof;
- $h \leftarrow H_{vrf}(\texttt{SK}, \texttt{M}) = H(\pi)$ - function that computes the hash.

Node u determines his committee membership using his private key $\texttt{SK}_u$ by Algorithm 7. Here $\texttt{M}_r$ can be locally computed by each node in a deterministic way and ϵ a predefined threshold. See Section 2.9 for details of how to compute $\texttt{M}_r$. Given the VRF proof $\pi_{u,r}$ and the corresponding public key $\texttt{PK}_u$, anyone can verify the membership claimed by u by Algorithm 8 below.

Algorithm 7 Procedure for u to determine his committee membership.

```
Compute $\texttt{M}_r$
Compute $\pi_{u,r} = \Pi_{vrf}(\texttt{SK}_u, \texttt{M}_r)$ and $\lambda = H(\pi_{u,r})$
if $\lambda \leq \epsilon$ then
    Return $(\texttt{TRUE}, \pi_{u,r})$
else
    Return $\texttt{FALSE}$
end if
```

Algorithm 8 Procedure for anyone to validate u's claimed committee membership.

```
Compute $\mathtt{M}_r$
**if** $V_{vrf}(\mathtt{PK}_u, \pi_{u,r}) = \mathtt{FALSE}$ **then**
    Return $\mathtt{FALSE}$
**end if**
$\lambda \leftarrow H(\pi_{u,r})$
**if** $\lambda \leq \epsilon$ **then**
    Return $\mathtt{TRUE}$
**else**
    Return $\mathtt{FALSE}$
**end if**
```

2.9 Random Beacon

Nodes need to compute common message $\mathtt{M}_r$ to determine and validate committee memberships. It is desirable to add certain randomness in the computation such that adversaries only know their committee memberships in a limited number of rounds in future.

To do that, we divide the consensus process into epochs each of which lasts for L rounds where L is a predetermined fixed number. Let $\hat{B}_{u,r}^m$ be the last block on the canonical chain observed by u in the m'th epoch. Node u can compute a random beacon b_u^m from $\hat{B}_{u,r}^m$ as:

$$b_u^m \leftarrow H(\mathtt{Sig}_{l_r}(\pi_{u_1,r} \,\|\, \pi_{u_1,r} \,\|\, \cdots \,\|\, \pi_{u_d,r})). \tag{1}$$

Here we define $b_u^0 \leftarrow H(B_{\text{genesis}})$. Beacon b_u^m is then used to calculate VRF messages for the next epoch. In particular, we compute message $\mathtt{M}_{r'}$ in round r' as:

$$\mathtt{M}_{r'} \leftarrow H(b_u^m \,\|\, r') \tag{2}$$

where $mL + 1 \leq r' \leq m(L + 1)$.

Now let us consider the case when the leader who generates $\hat{B}_{u,r}^m$ is malicious. Since signatures from the committee members are deterministic, the best he can do is to go through every combination of d signatures from all the signatures he has received from committee members and pick the one that maximizes his interest. The number of possible trials is fairly limited and therefore, we significantly limit the leader's influence on the beacon.

2.10 Double-Spending Attack

The main goal of introducing the committee mechanism is to make it more difficult for adversaries to cause inconsistency. Perhaps the most damaging of such attacks is the double-spending attacks (DSAs) [3] where adversaries are allowed to take control of a few consecutive consensus rounds such that they can produce two parallel branches to launch a DSA. Here we are going to infer a upper bound for the probability of launching a k-block DSA.

Let p_ϵ be the probability of a node being selected as a committee member. This probability is directly related to threshold ϵ and is equal to every node thanks to VRF. Let us assume that there are f malicious nodes that can behave arbitrarily. To produce two valid, but contradicting blocks in the same round, adversaries must control both the leader and d committee members. The probability of existence of d committee members being malicious can be computed as:

$$F(p_\epsilon, d, f) = \sum_{i=d}^{f} \binom{f}{i} {p_\epsilon}^i \, (1 - p_\epsilon)^{f-i} \tag{3}$$

Therefore, the probability of adversaries controlling the committee for k consecutive rounds is $F(p_\epsilon, d, f)^k$. Table 2 lists the values of this probability with different inputs. We set f according to the Byzantine Fault Tolerance (BFT) assumption. It is a security assumption that is considered reasonable and widely used in practice.

Now let us consider the worse-case scenario when a malicious leader is selected to generate the block in the last epoch from which the random beacon for the current epoch is computed. The best he can do is to try all the combinations of d valid signatures he receives from the committee members to search for a combination that will result in k consecutive rounds in which leaders are all malicious in the next epoch. Therefore, we can compute a upper bound for the probability of launching a k-block DSA as:

$$F^*(k, p_\epsilon, d, f) = \binom{c}{d} \cdot \left(\frac{f}{N}\right)^k \cdot F(p_\epsilon, d, f)^k. \tag{4}$$

where c is the total number of committee-member signatures the leader receives. Table 2 below provides some examples of the probabilities of launching a DSA.

$N = 101, f = 33$	$k = 5$	$k = 10$
$d = 5$	1.86×10^{-7}	1.65×10^{-15}
$d = 10$	2.78×10^{-8}	1.17×10^{-19}

Table 2: Examples of probabilities $F(p_\epsilon, d, f)^k$. Here we compute $p_\epsilon = 1.5 \cdot d/N$ and $c = \lfloor 1.5 \cdot d \rfloor$.

References

[1] Vechain development plan and whitepaper. `https://cdn.vechain.com/vechainthor_development_plan_and_whitepaper_en_v1.0.pdf`, 2018.

[2] D. Papadopoulos, D. Wessels, S. Huque, M. Naor, J. Vcelak, L. Reyzin, and S. Goldberg. Making NSEC5 practical for DNSSEC. *IACR ePrint*, 1999. `https://eprint.iacr.org/2017/099.pdf`.

[3] N. Satoshi. Bitcoin: A peer-to-peer electronic cash system. `https://bitcoin.org/bitcoin.pdf`, 2008.