

Contents

| | | |
|----------|---|----------|
| 1 | Dynamic Proof of Stake | 1 |
| 1.1 | Blockchain | 1 |
| 1.2 | Epoch | 1 |
| 1.3 | Genesis block | 1 |
| 1.4 | Block | 1 |
| 1.5 | Leader selection | 2 |
| 1.5.1 | absolute stake aggregation dependent leader selection functions | 2 |
| 1.6 | Leaky non-resettable beacon | 3 |
| 2 | Protocol | 3 |
| 3 | Appendix | 3 |
| 3.1 | Blockchain | 3 |
| 3.2 | Block | 4 |
| 3.3 | Metadata | 4 |
| 3.4 | Ouroboros Metadata | 4 |
| 3.5 | Streamlet Metadata | 4 |

1 Dynamic Proof of Stake

1.1 Blockchain

Blockchain \mathbb{C} is a series of epochs: it's a tree of chains, C_1, C_2, \dots, C_n , the chain of the max length in \mathbb{C} is the driving chain C .

1.2 Epoch

An epoch is a multiple of blocks. Some of those blocks might be empty due to the nature of the leader selection with VRF.

1.3 Genesis block

The first block in the epoch updates the stake for stakeholders, which influences the weighted random leader selection algorithm. For epoch j , the pair (S_j, η_j) is the genesis block's data for n stakeholders of the blockchain:

$$S_j = ((U_1, v_1^{vrf}, v_1^{kes}, v_1^{dsig}, s_1), \dots, (U_n, v_n^{vrf}, v_n^{kes}, v_n^{dsig}, s_n))$$

$$\eta_j \leftarrow \{0, 1\}^\lambda$$

Note that new stakeholders need to wait for the next epoch to be added to the genesis block

1.4 Block

A block \mathbf{B} is the building block of the blockchain.

Block $B_i = (st, d, sl, B_\pi, \rho, \sigma_s)$ created for slot i by a stakeholder, and slot i leader U_s :

st : state of the prebvious block, $\text{Hash}(\text{head}(\mathbb{C}))$

d : data held by the block

sl : slot id generated by the beacon

B_π : proof the stakeholder U_s is the owner, $B_\pi = (U_s, y, \pi)$, y, π are the output of the VRF

ρ : random seed for vrf, $\rho = (\rho_y, \rho_\pi)$

σ_s : owner signature on the block

1.5 Leader selection

At the onset of each slot each a stakeholder needs to verify if it's the weighted random leader for this slot.

$$y < T_i$$

check if VRF output is less than some threshold

This statement might hold true for zero or more stakeholders, thus we might end up with multiple leaders for a slot, and other times no leader. Also note that no one would know who the leader is, how many leaders are there for the slot, until you receive a signed block with a proof claiming to be a leader.

$$y = VRF(\eta || sid)$$

η is random nonce generated from the blockchain, **sid** is block id

$$\begin{aligned}\phi_f &= 1 - (1 - f)^{\alpha_i} \\ T_i &= 2^{l_{VRF}} \phi_f(\alpha_i^j)\end{aligned}$$

Note that $\phi_f(1) = f$, **f**: the active slot coefficient is the probability that a party holding all the stake will be selected to be a leader. Stakeholder is selected as leader for slot j with probability $\phi_f(\alpha_i)$, α_i is U_i stake.

1.5.1 absolute stake aggregation dependent leader selection functions

1.5.1.1 linear functions in the previous leader selection function, it has the unique property of independent aggregation of the stakes, meaning the property of a leader winning leadership with stakes σ is independent of whether the stakeholder would act as a pool of stakes, or distributed stakes on competing coins. “one minus the probability” of winning leadership with aggregated stakes is $1 - \phi(\sum_i \sigma_i) = 1 - (1 - (1 - f)^{\sum_i \sigma_i}) = (1 - f)^{\sum_i \sigma_i}$, the joint “one minus probability” of all the stakes (each with probability $\phi(\sigma_i)$) winning aggregated winning the leadership $\prod_i^n (1 - \phi(\sigma_i)) = (1 - f)^{\sum_i \sigma_i}$ thus:

$$1 - \phi(\sum_i \sigma_i) = \prod_i^n (1 - \phi(\sigma_i))$$

1.5.1.1.1 linear leader selection

$$\begin{aligned}y &< T \\ y &= 2^l k \mid 0 \leq k \leq 1 \\ T &= 2^l \phi(v) \\ \phi(v) &= \frac{1}{v_{max}} v\end{aligned}$$

1.5.1.1.2 dependent aggregation linear leader selection has the dependent aggregation property, meaning it's favorable to compete in pools with sum of the stakes over aggregated stakes of distributed stakes:

$$\begin{aligned}\phi(\sum_i \sigma_i) &> \prod_i^n \phi(\sigma_i) \\ \sum_i \sigma_i &> (\frac{1}{v_{max}})^{n-1} v_1 v_2 \dots v_n\end{aligned}$$

let's assume the stakes are divided to stakes of value $\sigma_i = 1$ for $\Sigma > 1 \in \mathbb{Z}$, $\sum_i \sigma_i = V$

$$V > (\frac{1}{v_{max}})^{n-1}$$

note that $(\frac{1}{v_{max}})^{n-1} < 1, V > 1$, thus competing with single coin of the sum of stakes held by the stakeholder is favourable.

1.5.1.1.3 scalar linear aggregation dependent leader selection a target function T with scalar coefficients can be formalized as

$$T = 2^l k\phi(\Sigma) = 2^l \left(\frac{1}{v_{max}}\right) \Sigma$$

let's assume $v_{max} = 2^v$, then:

$$T = 2^l k\phi(\Sigma) = 2^{l-v} \Sigma$$

then the lead statement is

$$y < 2^{l-v} \Sigma$$

for example for a group order or $l = 24$ bits, and maximum value of $v_{max} = 2^{10}$, then lead statement:

$$y < 2^{14} \Sigma$$

1.5.1.1.4 competing max value coins for a stakeholder with nv_{max} absolute stake, $|n \in \mathbb{Z}|$ it's advantageous for the stakeholder to distribute stakes on n competing coins.

1.5.1.2 inverse functions inverse lead selection functions doesn't require maximum stake, most suitable for absolute stake, it has the disadvantage that it's inflating with increasing rate as time goes on, but it can be function of the inverse of the slot to control the increasing frequency of winning leadership.

1.5.1.2.1 leader selection without maximum stake upper limit the inverse leader selection without maximum stake value can be $\phi(v) = \frac{v}{v+c}$ where c is $\$ > 1\$$ and inversely proportional with probability of winning leadership, let it be called leadership coefficient.

1.5.1.2.2 decaying linear leader selection as the time goes on, and stakes increase, this means the combined stakes of all stakeholders increases the probability of winning leadership in next slots leading to more leaders at a single slot, to maintain, or to be more general to control this frequency of leaders per slot, c (the leadership coefficient) need to be function of the slot sl , i.e $c(sl) = \frac{sl}{R}$ where R is epoch size (number of slots in epoch).

1.5.1.2.3 pairing leader selection independent aggregation function the only family of functions that are isomorphic to summation on multiplication (having the independent aggregation property) is the exponential function, and since it's impossible to implement in plonk,

TODO (proof)

a re-formalization of the lead statement using pairing that is isomorphic to summation on multiplication is also an options.

1.6 Leaky non-resettable beacon

Built on top of globally synchronized clock, that leaks the nonce η of the next epoch a head of time (thus called leaky), non-resettable in the sense that the random nonce is deterministic at slot s , while assuring security against adversary controlling some stakeholders.

For an epoch j , the nonce η_j is calculated by hash function H , as:

$$\eta_j = H(\eta_{j-1} || j || v)$$

v is the concatenation of the value ρ in all blocks from the beginning of epoch e_{i-1} to the slot with timestamp up to $(j-2)R + \frac{16k}{1+\epsilon}$, note that k is a persistence security parameter, R is the epoch length in terms of slots.

2 Protocol

3 Appendix

This section gives further details about the structures that will be used by the protocol. Since Streamlet consensus protocol will be used at early stages of the Blockchain development, we created hybrid structures, to enable seamless transition from one protocol to the other, without the need of a blockchain forking.

3.1 Blockchain

| Field | Type | Description |
|---------------------|-------------------------------|--|
| <code>blocks</code> | <code>Vec<Block></code> | Series of blocks consisting the Blockchain |

3.2 Block

| Field | Type | Description |
|-----------------------|-------------------------------------|-----------------------------------|
| <code>st</code> | <code>String</code> | Previous block hash |
| <code>sl</code> | <code>u64</code> | Slot UID, generated by the beacon |
| <code>txs</code> | <code>Vec<Transaction></code> | Transactions payload |
| <code>metadata</code> | <code>Metadata</code> | Additional block information |

3.3 Metadata

| Field | Type | Description |
|------------------------|--------------------------------|---|
| <code>om</code> | <code>OuroborosMetadata</code> | Block information used by Ouroboros consensus |
| <code>sm</code> | <code>StreamletMetadata</code> | Block information used by Streamlet consensus |
| <code>timestamp</code> | <code>Timestamp</code> | Block creation timestamp |

3.4 Ouroboros Metadata

| Field | Type | Description |
|--------------------|------------------------|--|
| <code>proof</code> | <code>VRFOutput</code> | Proof the stakeholder is the block owner |
| <code>r</code> | <code>Seed</code> | Random seed for the VRF |
| <code>s</code> | <code>Signature</code> | Block owner signature |

3.5 Streamlet Metadata

| Field | Type | Description |
|------------------------|------------------------------|---------------------------|
| <code>votes</code> | <code>Vec<Vote></code> | Epoch votes for the block |
| <code>notarized</code> | <code>bool</code> | Block notarization flag |
| <code>finalized</code> | <code>bool</code> | Block finalization flag |