

The Generic PoS algorithm

Diego Olivier Fernandez Pons

diego.olivier.fernandez.pons@gmail.com

TZ Ventures - Tezos Combinator

October 12, 2020



The generic PoS algorithm

Generic PoS

The generic PoS algorithm is a **template algorithm** similar to Paxos in the sense it can be instantiated in many ways to obtain a large variety of PoS algorithms

It **simulates Nakamoto consensus** (PoW) without resorting to hash reversal or any similar CPU intensive mechanism

PoS algorithms like Algorand, Tendermint Core, Ouroboros, Emmy, Emmy+ are instantiations of the generic PoS algorithm

Generic PoS - Properties

In the presence of **economically rational actors**, Nakamoto consensus is a **voting system**

- ▶ Each round represents a vote
- ▶ What is being chosen is a chain
- ▶ Votes are materialized by adding a block to an existing chain
- ▶ Participants have a number of votes that is proportional to their number of coins (if the distribution is considered fixed)

The generic PoS algorithms tries to achieve the same properties

Generic PoS - Origins

The generic PoS is the combination of the works of

- ▶ Peercoin (King et Nadal 2014)
- ▶ Chains of Activity (Bentov et al. 2014)
- ▶ Slasher (Buterin 2014)

To our best knowledge, there are no publication that clearly explain the generic PoS algorithm. The closest one being the Tezos white paper (Goodman 2014). The algorithm can also be inferred from the description of Tendermint (Buchman 2016).

Generic PoS - Step 1 Sortition

The generic Pos works by rounds

Sortition : Randomly select for each round

- ▶ A leader
- ▶ A list of backups
- ▶ A fixed cardinality committee

in a way that is proportional to the coins they own

This can be done on-the-fly or in advance, in a centralized or decentralized way.
Most of the time the process revolves around using a pseudo-random function
with a seed that is revealed to the network.

Generic PoS - Step 2 Committee vote

There may be more than one block circulating as the (tentative) block for round n

- ▶ To guarantee liveness, the node producer has backups
- ▶ As it is not possible to know if the node producer has crashed or is experiencing network delays, there will be situations where both the node producer and the backup have created blocks

Committee vote : The committee members vote for their choice of block for round n

The vote can be blocking or not, simple or using a BFT algorithm, etc

Generic PoS - Step 3 Chain selection

Consensus : Nodes are provided with a recommendation on which chain to chose for each round. While not required to follow the recommendation, nodes have to select one chain for each round.

This property is only *observable* when a node creates a new block, therefore the rule can be restated as "a block producer shall not create a blocks on multiple chains"

Generic PoS - Step 4 Slashing

Nodes may attempt to violate the rules in any of the previous steps

- ▶ Bias the randomness of the sortition
- ▶ Vote multiple times in a committee
- ▶ Create blocks on multiple chain

Slashing : Any violation is punished by destruction of coins

Generic PoS - Step 5 Finalization

Blocks need to be eventually finalized, meaning after a certain amount of time, it is not possible to vote for a chain originating in them anymore.

Finalization : There needs to be a finalization rule

The finalization rule can be arbitrary like "after N blocks", can be based on the committee vote or any other means

Generic PoS - Classic attacks

The generic PoS prevents the major classic attacks

Nothing-at-stake : creating blocks on two chains

- ▶ In PoW creating 2 blocks for the same round costs twice money and only one will be reimbursed, therefore it makes no economic sense
- ▶ In PoS the slashing rule economically penalizes the creation of 2 blocks in the same round

Generic PoS - Classic attacks

Posterior corruption : buying for cheap "coins in the past" of someone that already sold them (= buying their private key) and try to build a new chain from a past block with a large number of coins acquired that way

- ▶ In PoW it is not possible to buy "computer power in the past"
- ▶ In PoS the termination rule prevents building chains too far in the past

Generic PoS - Classic attacks

Long range attack : creating a chain that starts in the past with only the malicious actors participating (slots of other actors are ignored / deferred to malicious backups) and revealing it when its score becomes higher than the honest chain

- ▶ In PoW creating a block with full computing capacity takes 10 min. It takes longer if only a subset of the miners participate, therefore the chain is protected by the fact that the minority chain will never reach the length of the majority chain
- ▶ In PoS the termination rule partially protects against these type of attacks. Emmy+ has a delay mechanism that emulates the PoW time protection

Generic PoS - Classic attacks

Grinding : tampering with the randomness of sortition

- ▶ In PoW sortition is achieved by a race to reverse a hash, which is complex to tamper with (weaknesses of the hash function, dedicated hardware)
- ▶ In PoS the slashing rule economically penalizes the most obvious attempts at tampering with sortition (e.g. not revealing the seed) but some minor attacks are still possible in some implementations (e.g. trying many seeds for the pseudo-random function and choosing the one that gives you more slots)

Generic PoS - Other attacks

The Ouroboros paper claims the Ouroboros (an instance of the basic PoS algorithm)

Is immune to

- ▶ Double spending
- ▶ Grinding
- ▶ Transaction censorship
- ▶ Long-range attack
- ▶ Nothing at stake
- ▶ Past majority attack
- ▶ Selfish-mining

Is not immune to

- ▶ Desynchronization attack
- ▶ Eclipse attack
- ▶ 51% attack
- ▶ Bribery attack

Instances of the generic PoS algorithm

Emmy

Emmy is a very straightforward instance of the generic PoS. It was described in the Tezos white paper (Goodman 2014)

Miscellaneous

- ▶ Rounds are separated by (at least) 1 minute
- ▶ Rounds are grouped in cycles of 4096
- ▶ There is the *implicit* assumption that the network has a bounded delay $\Delta = 1 \text{ min}$ (messages will arrive in less than 1 minute)

Emmy - Sortition

Sortition is precomputed in a decentralized way using the *follow-the-satoshi* algorithm

- ▶ Snapshots of the coin distribution are taken on a regular basis
- ▶ A node takes randomly a snapshot and computes for each slot
 - ▶ A list of priorities for active nodes
(0 = block producers, others = backup order)
 - ▶ A committee of 32 members "endorsers" a snapshot and computes
- ▶ The node reveals the seed used to feed the pseudo-random function

Emmy - Block production

The block producer of block n has 1 minute to

- ▶ Select the transactions it wants to add in the block
- ▶ Add the endorsements it has received for block $n - 1$
- ▶ Execute the transactions
- ▶ Compute the hash of the final state
- ▶ Broadcast the new block

If no block from the block producer appears after 1 minute, the backup of priority 1 is allowed to produce the block, and so on

$$\text{delay} = 60 + 75 * \text{priority}$$

Emmy - Committee vote (endorsements)

Each member of the committee for level n votes for the block they endorse by broadcasting their endorsement in the network

Emmy - Chain selection

Nodes are invited to select as head the chain / node with the highest fitness

$$F_n = F_{n-1} + \text{nb_endorsements} + 1$$

Where $n - 1$ is the block preceding the n block (according to n which can choose any block as predecessor)

In other words the recommended chain is the one that maximizes length + endorsements

Emmy - Slashing

There are 3 types of offenses that result in destruction of coins

- ▶ Not revealing seed for sortition
- ▶ Voting twice when committee members
- ▶ Creating two blocks of same level when block producer

Coins required as collateral for slashing are frozen at the beginning of the cycle and unfrozen 5 cycles later. Any node that reports a violation receives 50% of the coins slashed (the other 50% are destroyed to avoid collusions)

Emmy - Finalization

Like in Nakamoto consensus, Emmy has probabilistic finality.

Blocks are also deterministically finalized after 5 cycles (chains rooted in a block older than 5 cycles are invalid)

Emmy - Limits / issues

Emmy has two issues

- ▶ Block producers have to decide between sending their block as soon as possible (to avoid their block to compete with the backup producer one), or waiting for more endorsements for the previous block (to increase their block score)
- ▶ The protection against long-range attacks given by static finalization is weak

Emmy+

Emmy+ (Tezos 2019) was designed to solve the limitations of Emmy and simplify its formal analysis

Emmy+

The idea behind Emmy+ design is to emulate the property of Nakamoto consensus where minority chains advance slower than majority ones, which protects Bitcoin from long-range attacks

- ▶ The fitness function is now time (longest chain) $F_n = n$
- ▶ The endorsements by the committee are not used to increase the fitness directly, but rather to delay the blocks endorsed by a minority from reaching other nodes

$$\text{delay} = 60 + 40 * \text{priority} + \max \{0, (\theta - e) * T_0\}$$

where e the number of endorsements, θ a threshold (24), and T_0 a time (8)

- ▶ In Emmy+ the rewards for producing and endorsing blocks are set in such a way that there is no economic benefit in not holding the production of a block to wait for more endorsements for block $n - 1$
- ▶ The delays create a much stronger protection against long-range attacks in that they make a chain that is supported only by a minority of blocks slower to build than the majority chain, like in Nakamoto consensus

Tenderbake

Tenderbake (Tezos 2020) is a variant of Tendermint-core that improves some aspects of Tendermint.

Tenderbake hasn't been implemented (and probably won't be) as it is considered inferior to Emmy+ even if it provides deterministic finality. Instead Nomadic Labs is investigating a "finality gadget" to be added to Emmy+

Ouroboros

Ouroboros is a family of PoS algorithm variants

- ▶ Ouroboros (2016) : similar to Emmy
- ▶ Ouroboros Praos (2017)
- ▶ Ouroboros Genesis (2018)
- ▶ Ouroboros BFT (2018)
- ▶ Ouroboros Chronos (2019)

TODO

TODO

Tendermint

TODO