---

title: Secret leader election description: Explanation of how secret leader election can help protect validators from attacks lang: en summaryPoints: - The IP address of block proposers can be known in advance, making them vulnerable to attacks - Secret leader election hides the identity of validators so that they are not knowable in advance - An extension of this idea is to make validator selection random in each slot.

Secret leader election {#single-secret-leader-election}

In today's proof-of-stake based consensus mechanism, the list of upcoming block proposers is public and it is possible to map their IP addresses. This means that attackers could identify which validators are due to propose a block and target them with a denial-of-service (DOS) attack that leaves them unable to propose their block in time.

This could create opportunities for an attacker to profit. For example a block proposer selected for slot $n+1$ could DOS the proposer in slot n so that they miss their opportunity to propose a block. This would allow the attacking block proposer to extract the MEV of both slots, or grab all the transactions that should have been split across two blocks and instead include them all in one, gaining all the associated fees. This is likely to affect home validators more than sophisticated institutional validators who can use more advanced methods to protect themselves from DOS attacks, and could therefore be a centralizing force.

There are several solutions to this problem. One is Distributed Validator Technology which aims to spread the various tasks related to running a validator across multiple machines, with redundancy, so that it is much harder for an attacker to prevent a block from being proposed in a particular slot. However, the most robust solution is **Single Secret Leader Election (SSLE)**.

Secret single leader election {#secret-leader-election}

In SSLE, clever cryptography is used to ensure that only the selected validator knows they have been selected. This works by having each validator submit a commitment to a secret they all share. The commitments are shuffled and reconfigured so that no-one can map commitments to validators but each validator knows which commitment belongs to them. Then, one commitment is chosen at random. If a validator detects that their commitment was chosen, they know it is their turn to propose a block.

The leading implementation of this idea is called Whisk. Which works as follows:

1. Validators commit to a shared secret. The commitment scheme is designed such that it can be bound to a validator identity but also randomized so that no third party can reverse engineer the binding and link a specific commitment to a specific validator.
2. At the start of an epoch, a random set of validators is chosen to sample commitments from 16,384 validators, using RANDAO.
3. For the next 8182 slots (1 day), block proposers shuffle and randomize a subset of the commitments using their own private entropy.
4. After the shuffling is finished, RANDAO is used to create an ordered list of the commitments. This list is mapped onto Ethereum slots.
5. Validators see that their commitment is attached to a specific slot, and when that slot arrives they propose a block.
6. Repeat these steps so that the assignment of commitments to slots is always far ahead of the current slot.

This prevents attackers from knowing in advance which specific validator will propose the next block, preventing the ability for DOS attacks.

Secret non-single leader election (SnSLE) {#secret-non-single-leader-election}

There is also a separate proposal that aims to create a scenario where validators each have a random chance of proposing a block in each slot, similarly to how block proposal was decided under proof-of-work, known as **secret non-single leader**

election (SnSLE). One simple way to do this is to make use of the RANDAO function used to randomly select validators in today's protocol. The idea with RANDAO is that a sufficiently random number is generated by mixing hashes submitted by many independent validators. In SnSLE, these hashes could be used to choose the next block proposer, for example by choosing the lowest-value hash. The range of valid hashes could be constrained to tune the likelihood of individual validators being selected in each slot. By asserting that the hash must be less than `2^256 * 5 / N` where `N` = number of active validators, the chance of any individual validator being selected in each slot would be `5/N`. In this example, there would be a 99.3% chance of at least one proposer generating a valid hash in each slot.

Current progress {#current-progress}

SSLE and SnSLE are both in the research phase. There is no finalized specification for either idea yet. SSLE and SnSLE are competing proposals that couldn't both be implemented. Before shipping they need more research and development, prototyping, and implementing on public testnets.

Further reading {#further-reading}

- SnSLE