Let us suppose that, for pure proof of stake, the main chain uses a simple RANDAO-based RNG. That is, the main chain has an internal state, R

, and every validator, upon depositing, commits to the innermost value in a hash onion, H(H(H(.....S.....)))

; the state stores the “current commitment value” C_{V_i}

for each validator V_i

. To create a block, a validator V_i

must include the preimage of the C_{V_i}

value (call it H^{-1}(C_{V_i})

), and the following state changes happen: (i) C_{V_i} := H^{-1}(C_{V_i})

, (ii) R := R\ xor\ H^{-1}(C_{V_i})

; that is, the current commitment value for that validator is set to the preimage, so next time the validator will have to unwrap another layer of the onion and reveal the preimage of that

, and the preimage is incorporated into R

.

This has the property that when some validator gets an opportunity to propose a block, they only have two choices: (i) make a block, where they can only provide one specific hash value, the preimage of the previous value they submitted, or (ii) not make a block. The validator knows what the new R

value would be if they make a block, but not the R

value if they don’t (as it’s based on the next validator’s preimage, which is private information); the validator can either choose the value that they know or a value that they don’t know, the latter choice coming at a cost of their block reward B

.

Now, suppose that this value is used as an RNG seed to sample a large set of N

validators (eg. 200 validators per shard, multiplied by 100 shards, would give N = 20000

). Suppose the reward for being selected is S

. Suppose that a given participant has portion p

of participation in both the main chain and the shards; hence, they have a probability p

of creating the next block, and they get an expected N * p

slots selected each round, with standard deviation \sqrt{N * p}

(it’s a Poisson distribution). The question is, when the validator gets selected, could they profit by skipping block production?

The validator knows ahead of time how many slots they would get selected for if they make a block; this value is sampled from the probability distribution with mean N * p * S

and standard deviation \sqrt{N * p} * S

. By not making a block, they sacrifice R

, but get to re-roll the dice on the other distribution and get an expected N * p * S

. By the 68 - 95 - 99.7 rule, the advantage from re-rolling will only exceed three standard deviations 0.15% of the time. If we are ok with exploitation being profitable at most 0.15% of the time, then we want R \ge 3 * \sqrt{N * p} * S

. If main-chain block rewards are comparable to total shard rewards, then R \approx N * S \gt\gt 3 * \sqrt{N * p} * S

even for p = 0.5

(above p = 0.5

, re-rolling the dice actually once again becomes less

profitable because there’s less room to get lucky in some sense). Hence, in a simple model there is little to worry about.

One further optimization (thanks @JustinDrake) is that if the validator sampling is split into shards, where in each shard the ability to validate is dependent on a proposal being available in that shard, then we can make the random source for that shard be R

xor'd with a preimage reveal from the proposer. This ensures that the validator making the main-chain block can only see the expected results for the portion of shards that they are a proposer of (in expectation, portion p

). This restricts the attacker's analysis to those shards, with standard deviation of the reward \sqrt{N * p * p} * S

= \sqrt{N} * p * S

; this makes lower main-chain rewards even safer.

We could look at more complex models, where we take into account the possibility of validators skipping one round because they see that they have both proposal slot 0 and slot 1, but if they wait until slot 1 then they also have slot 0 for the child of slot 1. These situations can be mitigated by penalizing absence of proposers; in any case, however, if total main-chain rewards are comparable to total shard-chain rewards, shard-chain rewards are a pretty negligible contribution to the attacker's profitability calculus.

In short, I really don't think we need fancy threshold signature randomness or low-influence functions or any other fancy math; a simple RANDAO mechanism as described above with total main chain rewards being comparable to total shard chain rewards is enough.