A major difference between Ethereum and most other (finality-bearing) proof of stake systems is that Ethereum tries to support a very high number of validators: we currently have 895,000 validator objects and a naive Zipf's law analysis implies that this corresponds to tens of thousands of unique invididuals/entities. The purpose of this is to support decentralization, allowing even regular individuals to participate in staking, without requiring everyone to give up their agency and cede control to one of a small number of staking pools.

However, this approach requires the Ethereum chain to process a huge number of signatures (~28,000 today; 1,790,000 post-SSF) per slot, which is a very high load. Supporting this load entails a lot of technical sacrifices:

- It requires a complicated attestation propagation

mechanism involving attestations being split between multiple subnets, needing tohyper-optimize BLS signature operations to verify these signatures, etc.

- We don't have a clear drop-in quantum-resistant alternative

that is anywhere near efficient enough.

- Fork choice fixes like view mergebecome much more complicated

because of the inability to extract individual signatures.

- SNARKing the signatures is hard

, because there's so many of them.Helios needs to operate over a specialized extra signature, called thesync committee signature

- It increases safe minimum slot times

by requiring three sub-slots in a slot instead of two.

The signature aggregation system feels

reasonable at first glance, but in reality it createssystemic complexity that bleeds out all over the place.

Furthermore, it does not even achieve its goal. The minimum requirement to stake is still 32 ETH, which is out of reach for many people. And just from a logical analysis, it seems infeasible to make an everyone-signs-in-every-slot system truly

enable staking for the average person in the long term

: if Ethereum has 500 million users, and 10% of them stake, then that implies 100 million signatures per slot. Information-theoretically, processing slashing in this design requires at least 12.5 MB per slot of data availability space, roughly as much as the target for full daksharding (!!!). Perhaps doable, but requiring staking itself to be dependent on data availability sampling is a large complexity gain - and even that is only ~0.6% of the world population staking, and does not begin to get into the computational

issue of verifying that many signatures.

Therefore, instead of relying on cryptographers to create magic bullets (ormagic bulletproofs) to make an ever-increasing number of signatures per slot possible, I propose that we make a philosophical pivot: move away from having such an expectation in the first place

. This will greatly expand the PoS design space, and will allow for a large amount of technical simplification, make Helios more secure by allowing it to SNARK over Ethereum consensus directly, and solve the quantum resistance issue by making even boring long-existing signature schemes like Winternitz viable.

Why not "just do committees"

Many non-Ethereum blockchains, faced with this exact problem, use a committee-based approach to security. During each slot, they randomly choose N validators (eg. N ~= 1000), and those are the validators responsible for finalizing that slot. It is worth reminding why this approach is insufficient: it does not provide accountability

.

To see why, suppose that a 51% attack does happen. This could be a finality-reversion attack or a censorship attack. For the attack to take place, you do still need economic actors controlling a large fraction of the stake to agree to

in the attack, in the sense of running software that participates in the attack with all validators that end up being selected for the committee. The math of random sampling ensures this. However, the penalty that they incur for such an attack is tiny

, because most of the validators that agreed to the attack end up not being seen because they are not chosen for the committee.

Ethereum, at present, takes the opposite extreme. If a 51% attack takes place, a large fraction of the entire

attacking validator set has their deposits slashed. The cost of an attack is at present around 9 million ETH (~$20 billion), and that assumes network synchrony breaks in a way that maximally favors the attacker

.

I argue that this is a high cost, but it is too high

a cost, and we can afford to make some sacrifices in the matter. Even a cost of attack of 1-2 million ETH should be totally sufficient. Furthermore, the main centralization risks of Ethereum that are present today are in a totally different place: large-scale staking pools would not be that

much less powerful if the minimum deposit size were reduced to near-zero.

This is why I advocate a moderate solution: one that makes some sacrifices on validator accountability, but still keeps the amount of total slashable ETH quite high, but in exchange gets us most of the benefits of a smaller validator set

.

What would 8192 signatures per slot under SSF look like?

Assuming a traditional two-round consensus protocol (like what Tendermint uses, and like what SSF would inevitably use), you need two signatures per slot per participating validator. We need to work around this reality. I see three major approaches for how we could do this.

Approach 1: go all-in on decentralized staking pools

The zen of Python contains a really key line:

There should be one-- and preferably only one --obvious way to do it.

For the problem of making staking egalitarian, Ethereum currently violates this rule, because we are simultaneously executing on two

distinct strategies toward this goal: (i) small-scale solo staking

, and (ii) decentralized stake pools usingdistributed validator technology (DVT)

. For the reasons described above, (i) is only able to support some

individual stakers; there will always be very many people for whom the minimum deposit size is too large. However, Ethereum is paying the very high technical burden costs of supporting (i).

A possible solution is to give up on (i) and go all-in on (ii). We could raise the min deposit size to 4096 ETH and make a total cap of 4096 validators (~16.7 million ETH). Small-scale stakers would be expected to join a DVT pool: either by providing capital or by being a node operator. To prevent abuse by attackers, the node operator role would need to be reputation-gated somehow, and pools would compete by providing different options in this regard. Capital provision would be permissionless.

We can make pooled staking in this model more “forgiving” by capping penalties, eg. to 1/8 of total provided stake. This would allow reducing trust in node operators, though it’s worth approaching this carefully due to the issues outlined here.

Approach 2: two-tiered staking

We create two layers of stakers: a “heavy” layer with a 4096 ETH requirement that participates in finalization, and a “light” layer with no

minimum (also no deposit and withdrawal delay, and no slashing vulnerability) that adds a second layer of security. For a block to finalize, both

the heavy layer needs to finalize it and

the light layer needs to have >= 50% of online light validators attest to it.

This heterogeneity is good for censorship and attack resistance, because one would need to corrupt both

the heavy layer and the light layer for an attack to succeed. If one layer is corrupted and not the other, the chain halts; if it’s the heavy layer that was corrupted, the heavy layer can be penalized.

Another benefit of this is that the light layer can include ETH that is simultaneously used as collateral inside applications. The main downside is that it makes staking less egalitarian by enshrining a divide between small-scale stakers and larger stakers.

Approach 3: rotating participation (ie. committees but accountable)

We take an approach in a similar spirit to thesuper-commtittee design proposed here: for each slot, we choose 4096 currently active validators, and we carefully adjust that set during each slot in such a way that we still have safety.

However, we make some different parameter choices to get the “maximum bang for our buck” within this framework. Particularly, we allow validators to participate with arbitrarily high balances, and if a validator has more than some quantity M

of ETH (which would have to be floating) then they participate in the committee during every slot. If a validator has $N < M$

ETH, then they have a $\frac{N}{M}$

probability of being in the committee during any given slot.

One interesting lever that we have here is decoupling “weight” for incentive purposes, vs “weight” for consensus purposes

: each validator’s reward within the committee should be the same (at least for validators with $\le M$

ETH), to keep average rewards proportional to balance, but we can still make consensus

count validators in the committee weighted by ETH. This ensures that breaking finality requires an amount of ETH equal to $> \frac{1}{3}$

of the total ETH in the committee.

A back-of-the-nakpin Zipf’s law analysis would compute that amount of ETH as follows:

- At each power-of-two level of total balance, there would be a number of validators inversely proportional to that balance level, and the total balance of those validators would be the same.
- Hence, the committee would have an equal amount of ETH participating from each balance level, except the levels above the barrier M

, where the validator is in the committee always.

- Hence we have k

validators at each of $log_2(M)$

levels, and $k + \frac{k}{2} + ... = 2k$

validators at the levels above. So $k = \frac{4096}{log_2(M) + 2}$

.

- The largest validator would have $M * k$

ETH. We can work backwards: if the largest validator has $2^{18} = 262144$

ETH, this would imply (roughly) $M = 1024$ and $k = 256$.

- The total ETH staked would be:
- The full stake of the top 512 validators $(2^{18} * 1 + 2^{17} * 2 + ... + 2^{10} * 2^{8} = 2359296$

)

- Plus the randomly sampled smaller stakes $(2^8 * (2^9 + 2^8 + 2^ 7 ...) \approx 2^8 * 2^{10} = 2^{18}$

)

- In total we get 2621440

ETH staked, or a cost of attack of ~900k ETH.

- The full stake of the top 512 validators $(2^{18} * 1 + 2^{17} * 2 + ... + 2^{10} * 2^{8} = 2359296$

)

- Plus the randomly sampled smaller stakes $(2^8 * (2^9 + 2^8 + 2^ 7 ...) \approx 2^8 * 2^{10} = 2^{18}$

)

- In total we get 2621440

ETH staked, or a cost of attack of ~900k ETH.

The main disadvantage of this approach is somewhat more in-protocol complexity to randomly choose validators in such a way that we get consensus safety even across committee changes.

The main advantages are that it preserves solo staking in a recognizable form, preserves a one-class system, and even allows for the minimum deposit size to be reduced to a very low level (eg. 1 ETH).

Conclusions

If we establish that in a post-SSF protocol, we want to stick to 8192 signatures, this makes the job much easier for technical implementers, as well as builders of side infrastructure like light clients. It becomes much easier for anyone to run a consensus client, and users, staking enthusiasts and others would be able to immediately work off of that assumption. The future load of the Ethereum protocol becomes no longer an unknown: it can be raised in the future through hard forks, but only when developers are confident that technology has improved enough to be able to handle a larger number of signatures-per-slot with the same level of ease.

The remaining job would be to decide which of the above three approaches, or perhaps something else entirely, we want to go with. This would be a question of which tradeoffs we are comfortable with, and particularly how we navigate related issues such as liquid staking, which could be resolved separately from the now-much-easier technical questions.