RANDAO is not ideal in that minor manipulation (controlling a few bits) is easy, albeit costly. Even if it's rarely profitable to manipulate randomness for the block rewards, it could be helpful in setting up a single-shard attack. If nothing else, the potential for manipulation makes it harder to analyze the feasibility of attacks.

Threshold signatures make manipulation harder, but not quite infeasible. If the threshold is 51%, an attacker could conceivably DOS 50% of the network, get the others' signature shares, privately compute the group signature, and decide whether to broadcast their own shares based on the result.

How about the following scheme? We devise some time-hard function f

with an adjustable difficulty d

. Maybe an iterated hash function f(x) = \text{hash}^d(x)

, preferably with a memory-hard hash function so that ASICs won't have a major advantage. In any case, the computation should take everybody \mathcal{O}(d)

time since it's not parallelizable; any advantages would be limited to a constant factor.

We have each validator include a random number r

in their block header. After each epoch, let's say 100 blocks, we start computing f(r_1, ..., r_{100})

. The result becomes the random seed for some future epoch, far enough out that everybody can complete the computation before then.

Admittedly this would probably introduce more problems than it solves. It would waste some computational resources, and it would become more difficult for new nodes to catch up to the network. I just think it's academically interesting that a single honest validator per epoch can prevent manipulation.