See previous work: Analysis on "Correlated Attestation Penalties" and Supporting decentralized staking through more anti-correlation incentives

This post introduces a concrete proposal for how correlated penalties could be done, in a way that maximizes (i) simplicity, and (ii) consistency with valuable invariants that exist today.

## Goals

1. Replicate the spirit of the basic design proposed here.
2. Maximum simplicity (the same type of simplicity as we can see in eg. the EIP-4844 blob gas market design)
3. Same average validator revenue as today, at all levels of “percent attesting correctly”. This should hold as a hard invariant, even against attackers with a large percent of stake trying to break it
4. Same penalty as today from failing to make one attestation, on average
5. Validators should only be rewarded for sending an attestation, never passively

## Mechanism

- We set two constants: PENALTY_ADJUSTMENT_FACTOR = 2**12

, MAX_PENALTY_FACTOR = 4

- We add a counter to the state, NET_EXCESS_PENALTIES
- During a slot, let non_attesting_balance

be the total balance that is not

correctly attesting in that slot

- Let: penalty_factor = min((non_attesting_balance * PENALTY_ADJUSTMENT_FACTOR) // (NET_EXCESS_PENALTIES * total_active_balance + 1), MAX_PENALTY_FACTOR)
- Let R

be the current reward for attesting correctly (computed based on base_reward

and adjusted based on the fraction allocated to the job in question). This stays the same.

- If a validator fails

to attest correctly, they get penalized penalty_factor * R

(as opposed to R

as today)

- At the end of a slot, set: NET_EXCESS_PENALTIES = max(1, NET_EXCESS_PENALTIES + penalty_factor) - 1

## Rationale

It should be easy to see that NET_EXCESS_PENALTIES

tracks sum(penalty_factor[slot] for slot in slots) - len(slots)

. Hence, if penalty_factor

on average exceeds 1 for a sustained period of time, NET_EXCESS_PENALTIES

will keep rising until that’s no longer the case. NET_EXCESS_PENALTIES

is part of the denominator in the calculation of penalty_factor

, and so NET_EXCESS_PENALTIES

rising will push the average penalty_factor

values down until the average is below 1 (and likewise in reverse, if it decreases).

penalty_factor

is proportional to the total non_attesting_balance

of the current slot, and so for it to average 1, it must roughly equal the non_attesting_balance

of the current slot divided by the long term average - exactly the design proposedhere.

Because penalty_factor

averages 1, average non-participation penalties are equal to R

, as today. And so average rewards for a validator are the same as today, for any correct attestation rate, assuming that their incorrect attestations are uncorrelated with those of other validators.

PENALTY_ADJUSTMENT_FACTOR

affects how quickly penalties can adjust.

## Possible extensions

- Make penalty_factor

more “continuous”, eg. by putting the base_reward

into the numerator that computes the penalty_factor

(and into the maximum, and into the per-slot decrement) and then using it to compute penalties directly.

- Explore smarter ways to apply this mechanism across multiple jobs (correct head attestation, target attestation…). The naive approach is to just apply it sequentially for each job, but there may be a smarter approach.