One feature that would be nice to add to eth2 would be the ability for a validator to switch their staking key. This is nice because it would allow a form of de-facto "delegation" where validators give their staking keys to a pool, and if the pool misbehaves or starts to have a high risk of misbehaving they can quickly switch the key to a different pool or to a self-hosted key, without any

downtime. Currently, switching keys requires one withdraw → re-deposit cycle, incurring a minimum of 27 hours of downtime.

The reason why no functionality for delegation or switching keys is available in eth2 is not (as somehave guessed) out of a deliberate attempt to prevent these behaviors; rather, it was because it is difficult to do this while at the same time preserving accountability for slashing. Most protocols with easy delegation mechanics do not have an accountability mechanism where the user themselves gets slashed if the user they delegate to misbehaves. Here is a proposal for allow key switching in a safe way that preserves accountability.

Data structure changes

We replace the pubkey

variable in the Validator

object with three variables:

old_pubkey: BLSPubkey new_pubkey: BLSPubkey switch_epoch: Epoch

The pubkey of a validator in some epoch is just validator.new_pubkey if epoch >= validator.switch_epoch else validator.old_pubkey

. Attestations and slashings can be processed with the correct pubkey automatically.

Key switching

A validator can send a message KeySwitch = {new_pubkey: BLSPubkey, switch_epoch: Epoch, signature: BLSSignature}

which can get included on-chain. It checks that the switch_epoch

is in the future, and that the signature is valid against the current validator.new_pubkey

. Processing the key switch is simple:

validator.old_pubkey = validator.new_pubkey validator.switch_epoch = key_switch.switch_epoch validator.new_pubkey = key_switch.new_pubkey

To prevent the existing old_pubkey

from being overwritten too early and thus escaping slashings, we also require the validator to be well past

its key transition. That is, we require current_epoch >= validator.switch_epoch + {8 months}

(note: if a transition has not yet happened, the check passes because validator.switch_epoch = 0

, assuming the chain is at least 8 months old).

To allow validators to switch keys more frequently when this is safe, we add an additional mechanism: during the exit queue clearing process, if normal exit clearing fully clears the exit queue, the remaining exit slots for that epoch are instead used to take the validators whose switch_epoch

is furthest in the past but nonzero, and set it to zero (the old_pubkey

can be set to zero at the same time to cut down storage space). This should allow pubkey switches to complete very quickly under normal circumstances, and only actually take up to 8 months in exceptional periods of congestion.

Control of key switching rights

The last question is, which validator key has the right to control signing key switches? We don't want the signing key to control signing key switches, as that gives the signing key too much power and runs counter to the goal of safe delegation. The withdrawal credentials are not necessarily a key at all. My proposed solution is to expand to three

keys:

- Signing key
- Administration key

- Withdrawal credentials

The administration key has the right to control (i) key switches and (ii) when the validator withdraws. The signing key can also trigger a withdrawal, though not a key switch. The administration key and withdrawal credentials can be stored under a single hash to save state space.