Cubic slashing

The slashing mechanism in Namada is designed to heavily disincentivize coordinated validator misbehavior, particularly among different operator accounts. It is known ascubic slashing because the slashed token amount scales with the cube of the infracting voting power or stake.

Typical examples of such misbehaviors or infractions include double signing a block or proposing an invalid block.

How slashes are processed

The Namada protocol receives data from CometBFT each block that might contain some piece of evidence of validator misbehavior. Namada ensures that the evidence is not outdated and then enqueues the slash for future processing at a parameterizable number of epochs after the infraction epoch.

Slashes are processed in the future to allow for sufficient time for the blockchain to detect infractions that might have occurred. During this period, the protocol ensures that no delegators to a slashed validator may unbond or redelegate.

Cubic slashing algorithm

When the slash is ready for processing, the slash rate for a given slash is computed as

min(param_min_slash_rate, cubic_slash_rate) whereparam_min_slash_rate is the minimum slash rate for a given slash type as set in the network's protocol parameters. These values can be queried from a network with the following (and example output):

> namadac query-protocol-parameters --node NODE | grep 'slash rate' Duplicate vote minimum slash rate: 0.001 Light client attack minimum slash rate: 0.001 Thecubic_slash_rate for a given slash or infraction is computed by considering other nearby slashes as well. Through this, repeated validator misbehavior or coordinated misbehavior by several malicious operators can be heavily penalized.

In particular, the cubic slash rate is determined by first computing the sum of the fractional voting power of each infraction relative to the total voting power of the consensus validator set, for all infractions in a window of epochs around the epoch of the current slash being processed.

Ifcurrent_epoch is the epoch of the current slash, then the window is[current_epoch - cubic_slashing_window_len, current_epoch + cubic_slashing_window_len] . The value ofcubic_slashing_window_len can also be obtained from the protocol parameters:

> namadac query-protocol-parameters --node NODE | grep 'Cubic slashing' Cubic slashing window length: 1 epoch With this summed fractional voting power computed, the cubic slash rate is then the square of this number scaled by9 . This is done such that one malicious validator with at least 1/3 of the consensus voting power will be slashed 100%.

The following is some Rust-like pseudocode that performs the cubic slashing computation:

/// Calculate the cubic slash rate for a slash in the current epoch fn

calculate_cubic_slash_rate (current_epoch :

Epoch , cubic_window_len :

u64 , slashes :

Map < Epoch , Vec < Slash

,) ->

Decimal { let start_epoch = current_epoch - cubic_window_len; let end_epoch = current_epoch + cubic_window_len;

let

mut sum_vp_frac =

Dec :: zero ();

for epoch in start_epoch ..= end_epoch { let cur_slashes = slashes . get (epoch); let total_consensus_voting_power =

get_consensus_voting_power (epoch);

let

mut vp_frac_this_epoch =

Dec :: zero (); for slash in cur_slashes { vp_frac_this_epoch += (slash . validator_voting_power / total_consensus_voting_power) }

vp_frac_sum += vp_frac_this_epoch; }

let rate = cmp :: min (Decimal :: ONE, cmp :: max (slash . param_min_slash_rate, 9

* vp_frac_sum * vp_frac_sum,),); rate }

/// Generic slash object with the misbehaving validator and infraction type struct

Slash { validator_voting_power :

Amount , param_min_slash_rate :

Dec , } The cubic slashing rate itself can be expressed in concise mathematical form as:

9 â (â e â i e v i e v tot e) 2 9*\left(\sum_e \sum_{i^e}\frac{\text{v}*i^e}}{\text{v}{\text{tot}}^e}\right)^2 9

â (e â â

i e â â

v tot e â

v i e â â) 2 wherev \text{v} v is voting power,e e e includes all epochs in the cubic slashing window, andi e i^e i e runs over all known infractions in the epoche e e .

After getting the slash rate

Once the slash rate is computed for a given slash, the appropriate number of tokens is deducted from the validator's stake and the slash data is stored on-chain.

The amount of tokens that is ultimately slashed is further determined such that a validator is slashed only considering stake that was used to commit the misbehavior. This means that if a given bond of staked tokens was never itself contributing to voting power when an infraction occurs, then those tokens can never be slashed.

Examples

Consider some examples to help demonstrate how cubic slashing works:

Single validator with a single slash

Assume that validatorA has10% of the consensus voting power and misbehaves once. The minimum slash rate is0.1% .

The slash rate ismin (0.001 , 9 * (0.1)^2) = 0.09 = 9%

Single validator with two nearby slashes

Assume that validatorA had10% of the consensus voting power in epoche and5% of the voting power in epoche+1 . The cubic slashing window length is1 , so both slashes are considered for the computation.

The cubic slash rate is computed as9 * (0.1 + 0.05)^2 = 9 * 0.15^2 = 0.2025 = 20.25% . Since there are no other slashes, this will be the slash rate for each of these slashes.

We then apply the slash rate of each slash to the voting power used to commit the infraction. So say that there are100 staked tokens in consensus, then the number of tokens slashed after both are processed:

slashed = (10)(0.2025) + (5)(0.2025) = ~ 3 tokens