A key sharding security assumption is that validators perform “windback”. That is, before proposing a collation header to the VMC the corresponding validator first checks the availability and validity of the immediate n

predecessors (e.g. n = 25

). This windback parameter is a local policy for each validator and there is the risk that “lazy” validators using too low a windback parameter weaken overall security.

In this post we suggest a natural way to enshrine a global windback parameter in the VMC, increasing the incentives for validators to make their local windback parameter at least as large as the global windback parameter. (For a much more high-tech approach to enforcing windback see this post.)

Construction

Modify the addHeader

method to return False

(and revert) on a collation header h

if the corresponding validator has previously called addHeader

on a header h'

where:

1. h'

and h

conflict, i.e. are on different forks

1. The parent of h'

is one of the immediate GLOBAL_WINDBACK

predecessors of h

Discussion

With this GLOBAL_WINDBACK

rule, calling addHeader

is:

1. Making a claim about the availability and validity of immediate predecessors
2. Accepting that future addHeader

calls be consistent with that claim

Put differently, adding a collation header inconsistent with one of your previously added collations for reasons other than availability and validity is considered adversarial and prevented by the VMC.

Notice that this rule holds large validators to a stricter standard than small validators, as larger validators propose collation headers more frequently and are more likely to “get caught in their own net”. This feels like a good decentralising counterbalance against the economies of scale validator pooling may provide.