We can use Kate commitments to create a very clean and simple 2D data availability scheme with collaborative encoding (ie. O(N^2)

data can be encoded by a very loose collection of N

participants with each participant only doing O(N*log(N))

work).

The scheme works as follows. Suppose that you have N

blobs, (D_1 ... D_n)

each containing M

chunks. The proposer of the blob publishes a Kate commitment to that blob. The proposer of the block

(eg. the beacon chain block in eth2) runs a data availability check on each commitment and includes the Kate commitments to the blobs (C_1 ... C_n)

in the block. However, they also

use Lagrange interpolation to compute a low-degree extension of the Kate commitments; that is, they treat [C_1 ... C_n]

as the evaluations of a polynomial at coordinates 1...n

, and compute the evaluations at coordinates n+1....2n

.

Call C_{n+1} ... C_{2n}

these extended commitments, and call D_{n+1} ... D_{2n}

the data blobs generated by low-degree extending the data blobs, ie. D_{n+1}[k]

would be the low degree extension of [D_1[k], D_2[k] ... D_n[k]]

.

Because converting between evaluations and coefficients and in-place multi-point evaluation are linear operations, you can compute C_{n+1} ... C_{2n}

despite the fact that the C_i

's are elliptic curve points. Furthermore, because C_i

is linear

in D_i

(ie. comm(D_1 + D_2) = comm(D_1) + comm(D_2)

and comm(D * k) = comm(D) * k

where the +

and *

on the right hand sizes are elliptic curve addition and multiplication), we get two nice properties:

- C_{n+1}

is the correct commitment to D_{n+1}

(and so on for each index up to 2n

)

- If W_{i, j} = G * (as_polynomial(D_i) // (X - \omega^j))

is the Kate witness verifying that D_i[j]

actually is in that position in D_i

, then you can take $[W_{1, j} \ldots W_{n, j}]$

and also do a low-degree extension to generate $[W_{n+1, j} \ldots W_{2n, j}]$

To recap:

- The block producer can generate the commitments $C_{n+1} \ldots. C_{2n}$

to the “redundancy batches” $D_{n+1} \ldots D_{2n}$

without knowing anything except the commitments

- If you know $D_1[j] \ldots D_n[j]$

for some column j

, you can reconstruct the values in that position in all of the redundancy batches, and those values are immediately verifiable

This gives us the conditions needed to have a very efficient protocol for computing and publishing the entire extended data. The design would work as follows:

- When doing the initial round of data availability sampling, each node would use the same indices for each batch. They would as a result learn $D_1[j] \ldots D_n[j]$

and $W_{1,j} \ldots W_{n,j}$

for some random j

- They can then reconstruct $D_{n+1}[j] \ldots. D_{2n}[j]$

and $W_{n+1,j} \ldots W_{2n,j}$

- If there are $\ge O(M)$

nodes (reminder: M

is the number of chunks), and the data is available, then with high probability for every row $i \in [n+1, 2n]$

there will be a node that learns $D_i[j]$

for enough positions j

that if they republish that data, all of D_i

can be reconstructed

This gives us an end state similar to the 2D encodinghere, except (i) we avoid any fraud proofs, and (ii) we avoid the need for one node to serve as the bottleneck that aggregates all the data to generate the extended Merkle root.