For context see: [Open problem: ideal vector commitment]

Compressed introduction / recap

Kate commitments

A Kate commitment is a technique for committing to a polynomial P

, where one starts with a trusted setup string consisting of elliptic curve points G

, $G * s$

, $G * s^2$

… $G * s^{n-1}$

for some never-published secret s

, and commits to $P(x) = \sum_{i=0}^{n-1} c_ix^i$

by computing $G * P(s) = \sum_{i=0}^{n-1} c_i * (G * s^i)$

, a linear combination of elements of the trusted setup string. We'll use [P]

as shorthand for $G * P(s)$

, including eg. $G = [1]$

, $G * s^i = [x^i]$

.

Witnesses: Q = P // (X - w^i)

Kate commitments can be used as a replacement for Merkle trees: to commit to a piece of data D = {D[0] ... D[n-1]}

, you interpolate the polynomial $P(x)$

that satisfies $P(1) = D[0]$

, $P(\omega) = D[1]$

… $P(\omega^{n-1}) = D[n-1]$

and more generally $P(\omega^i) = D[i]$

, where ω

is an "order-n root of unity", that is $\omega^n = 1$

. The "Merkle root" is just the commitment [P]

. A Merkle branch

is a Kate commitment of the quotient

$Q_i = P // (x - \omega^i)$

, where //

is the "rounded division" operator, eg. (3x + 4) // x = 3

, c // x = 0

for any constant c

, x^2 // (x - k) = (x + k)

(see if you can figure out why)… Note that unlike with integers, polynomial rounded division is a linear operator, that is (A + B) // C = A // C + B // C

and (A * z) // C = (A // C) * z

for any constant z

.

Checking witnesses: pairing check e(Q, X - w^i) ?= e(P-z, 1)

To use a witness Q_i

to prove that $P(\omega^i) = z$

, compute the pairing check: $e(Q_i, [X - \omega^i]) \overset{?}{=} e(P - z, [1])$

. If the check passes, then $Q_i * (X - \omega^i)$

actually equals $P - z$

, implying that $P - z$

is zero at ω^i

(as otherwise it could not be expressed as a product of $X - \omega^i$

and something else), implying that $P(\omega^i) = z$

.

Properties

Kate commitments have some powerful advantages:

- A witness is $O(1)$ sized, as opposed to Merkle witnesses, which are $O(\log(n))$

sized, where n

is the number of items in the tree

- You can combine many witnesses together: given Q_{i_1}

… Q_{i_k}

, you can take a linear combination of these values to generate $Q_{{i_1 ... i_k}} = P // ((X - i_1) * ... * (X - i_k))$

, which you can then verify as a single fixed-size witness for all

the values $z_1 ... z_k$

at those points.

So you can prove any number of values in a tree with a single point. However, compared to Merkle trees they have a big disadvantage: updating Kate witnesses is expensive

. With a Merkle tree, if in every block you need to read 1000 accounts and update 1000 accounts, you only need to update $\approx 1000 * \log(n)$

hashes in the tree to generate the Merkle branches for the 1000 accounts in the next block. With a Kate commitment, you would need to fully recompute every witness (or if you don’t recompute, computing a witness from scratch takes $O(n)$

time, where n

is once again the total size of the state, eg. $\approx 2^{29}$

for current ethereum).

This post proposes a technique that improves on this.

Our new technique

Client-side split: P’ = P + D, compute witnesses separately and recombine at the end

Let P

be the current state; assume we have already precomputed witnesses $[P // (X - \omega^i)]$

for all $i \in \{0 ... n-1\}$

. Now, suppose in a block we get k

writes to the state, so there is a new state P'

with k

modifications (ie. $P'(\omega^i) \ne P(\omega^i)$

at k

positions). We do the following. Client-side, represent $P' = P + D$

, where $|D| = k$

(we'll use $|D|$

to mean "the number of ω^i

coordinates where D

is nonzero").

Let $Z_{i_1 ... i_k}$

represent the degree-k polynomial that's zero at $\{\omega^{i_1} ... \omega^{i_k}\}$

, ie. $(X - \omega^{i_1}) * ... * (X - \omega^{i_k})$

. As mentioned above, we can make a "multi-witness" that proves $P(\omega^{i_j}) = z_j$

for all pairs in some list $\{(i_1, z_1) ... (i_k, z_k)\}$

simultaneously, and that witness just is $[P // Z_{\{i_1 ... i_k\}}]$

.

How to compute witness for D in O(|D|) time

Rounded polynomial division is linear, so a witness $[P' // Z_{\{i_1 ... i_k\}}]$

equals $[P // Z_{\{i_1 ... i_k\}}] + [D // Z_{\{i_1 ... i_k\}}]$

. We already have all $[P // (X - \omega^{i_j})]$

values and as mentioned above we can use a linear combination of those to construct $[P // Z_{\{i_1 ... i_k\}}]$

, so we just need to worry about $D // Z_{\{i_1 ... i_k\}}$

. If we look at $D // Z_{\{i_1 ... i_k\}}$

in evaluation form

(that is, in terms of its evaluations at ω^i

positions), we can see that it can only be nonzero at (i) positions where D

itself is nonzero, and (ii) positions in $\{i_1 ... i_k\}$

. Hence, we can compute it as a $|D| + k$

sized linear combination of $[L_i]$

elements (where $L_i = \frac{X^n - 1}{(X - \omega^i) * \prod_{j \ne i}(\omega^j - \omega^ii)}$

) equals 1 at ω^i

and zero at other powers of ω^i

); to determine the coefficients of this linear combination we need only solve some linear systems of equations (see https://notes.ethereum.org/AALpIfEzRWWExA5EVzLjOA for more efficient ways of doing this including removing the need for even superlinear field arithmetic).

Putting it all together

Let us summarize so far. Suppose in each block, there are k state modifications. After d blocks, $P' = P + D$, where $|D| = k * d$. The time needed to generate a witness for the next block is $|D| = k * d$ (note: this is one witness for all accesses in that block).

At any point, we can “refresh” the scheme by setting $P \leftarrow P'$ and $D \leftarrow 0$ and recomputing all $P // (X - \omega^i)$ values; this takes $O(n \log n)$ effort (a total of $3 n \log n$ EC operations using this technique: https://github.com/khovratovich/Kate/blob/master/Kate_amortized.pdf). If we recompute P every t blocks, then the total cost over those t blocks will be $3n \log n + \sum_{d=1}^t k * d \approx 3n \log n+ k * \frac{t^2}{2}$. The amortized per-block cost is $c = \frac{3n \log n}{t} + \frac{k * t}{2}$, minimized at $t = \sqrt{\frac{6*n \log n}{k}}$, $c = \sqrt{6k n \log n}$ (ie. $\sqrt{6k n \log n}$ elliptic curve operations per block).

Realistically, $n \approx 2^{30}$ and $k \approx 2^{10}$ so this would still require 2^{24} elliptic curve operations per block, slightly outside the range of feasibility, but nevertheless this is a considerable step forward.