Proof of Reserves

Emil Bay

@emil bayes

Hyperdivision

HYPERDIVISION

SCALING, DISTRIBUTED SYSTEMS AND CRYPTO

https://hyperdivision.dk

@hyperdivisiondk

INQUIRE TO HELLO@HYPERDIVISION.DK

$$xG$$

$$g^x$$

$$[x]$$

$$[x]G$$

Pedersen Commitments over EC

G, H are generators

Homomorphic

$$C(x_1, r_1) + C(x_2, r_2)$$

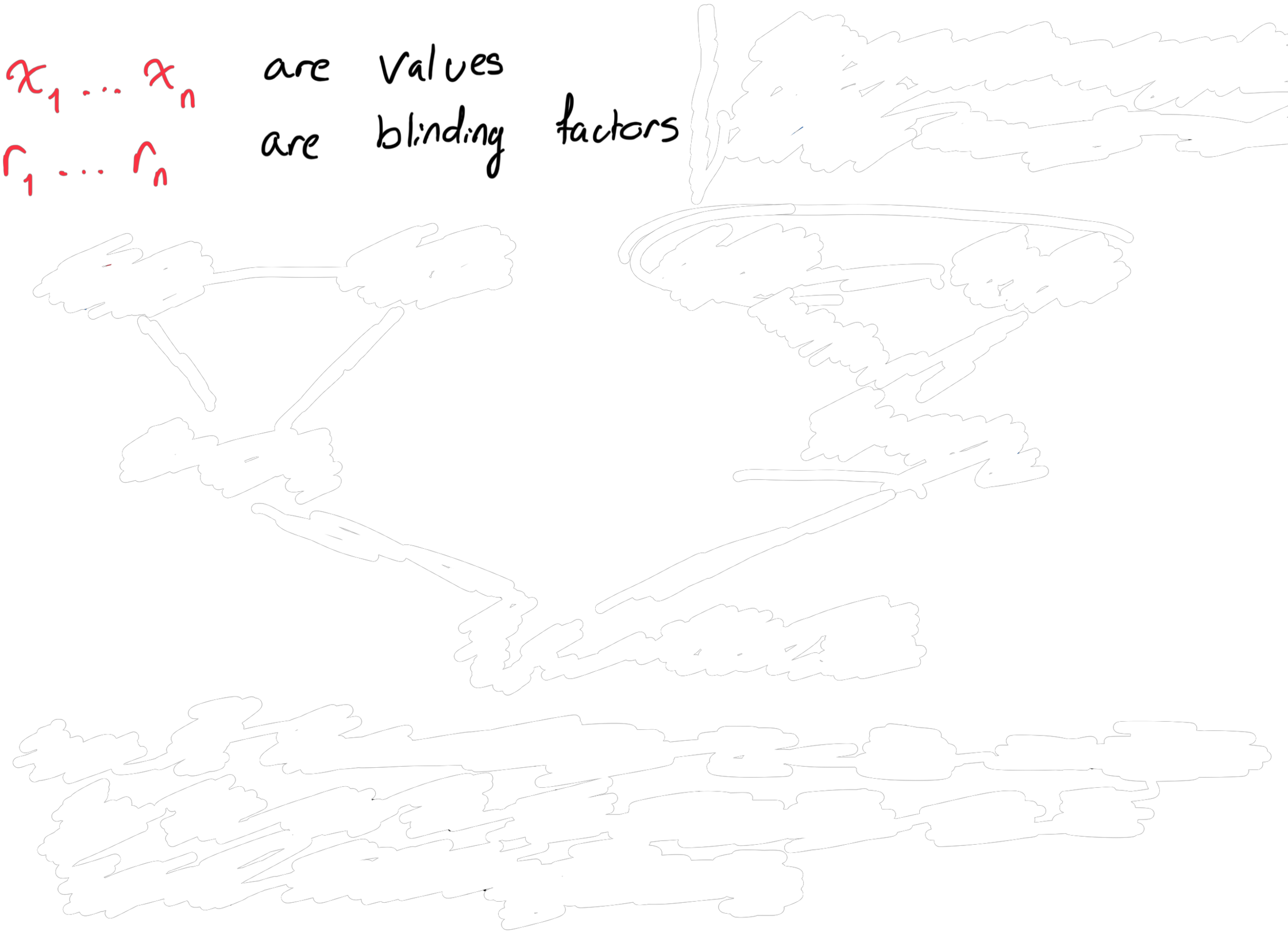
$$\| $$

$$C(x_1 + x_2, r_1 + r_2)$$

Pedersen Trees

$x_1 \dots x_n$ are values

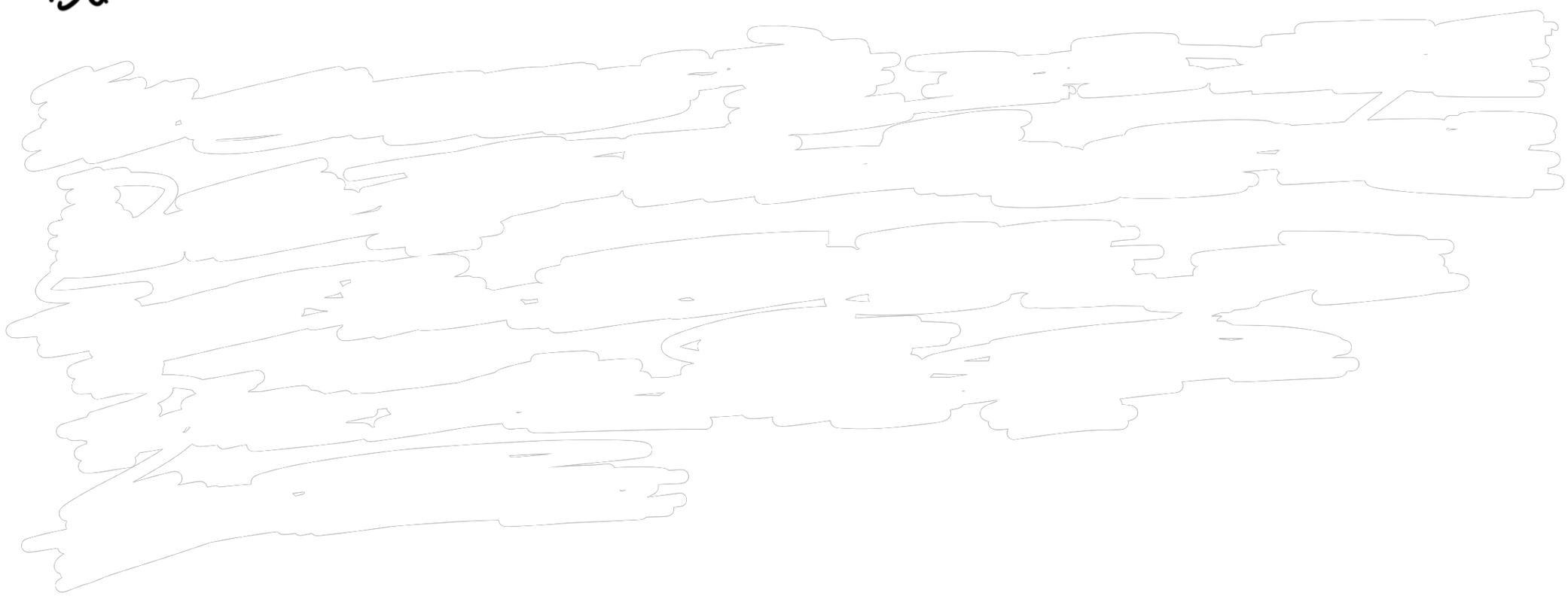
$r_1 \dots r_n$ are blinding factors

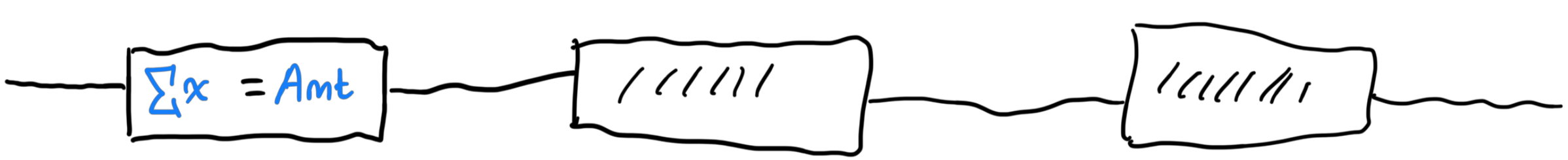


Use-Case: Proof-of-Reserves

If values are customer account balances, the root is the sum of all balances.

Each customer know their own x_i, receive their c_i and r_i and can verify their balance is accounted for.




∑x = Amt

But there's an issue

A malicious dealer could commit to $C(1, r_1)$ and $C(-1, r_2)$ and they would cancel out in $\sum x$ without verifiers knowing

Zero Knowledge Proof of Knowledge

Prove that:

$$C_i = C(x_i, r_i), \quad 0 \leq x_i \leq 2^n$$

$$\sum_i x_i = \sum X$$

Not revealing x_i, r_i, $\| x_i \|$

Why this works

1. The opening of the commitment inside the SNARK binds the public knowledge $c_1 \dots c_n$ to the amounts x_i
2. The bit decomposition of the amount binds the range of the amounts x_i, ie. prevents wrap around, such that two amount may cancel out due to the Pederson Commitment field size

$$x_i = 2^{b_1} + 2^{2b_2} + 2^{3b_3} + \dots + 2^{nb_n}$$

ie $x_i \leq 2^{33}$

$n = 33 \quad (32+1)$

$2^{33} \approx 8$ billion

3. The sum of the committed amounts $\sum x_i$ binds to the publicly verifiable sum $\sum X$

Implementation

- ZoKrates
- Script generates the script
- ~4.6K constraints per account
- ≈50 SLOC
- ≈ 300 SLoc Sage Math for support

Inputs

$C_0 \dots C_n$

total_amount

amounts $(x_0 \dots x_n)$ (also as bits)

blinding_factors $(r_0 \dots r_n)$ (as bits)

$n = \#\text{Accounts}$

$\max \|x\|$

Source Code

https://github.com/
emilbayes/ZK-proof-of-reserves/
emilbayes/abstract-algebra-fun/