BANKEX FOUNDATION

Plasmas

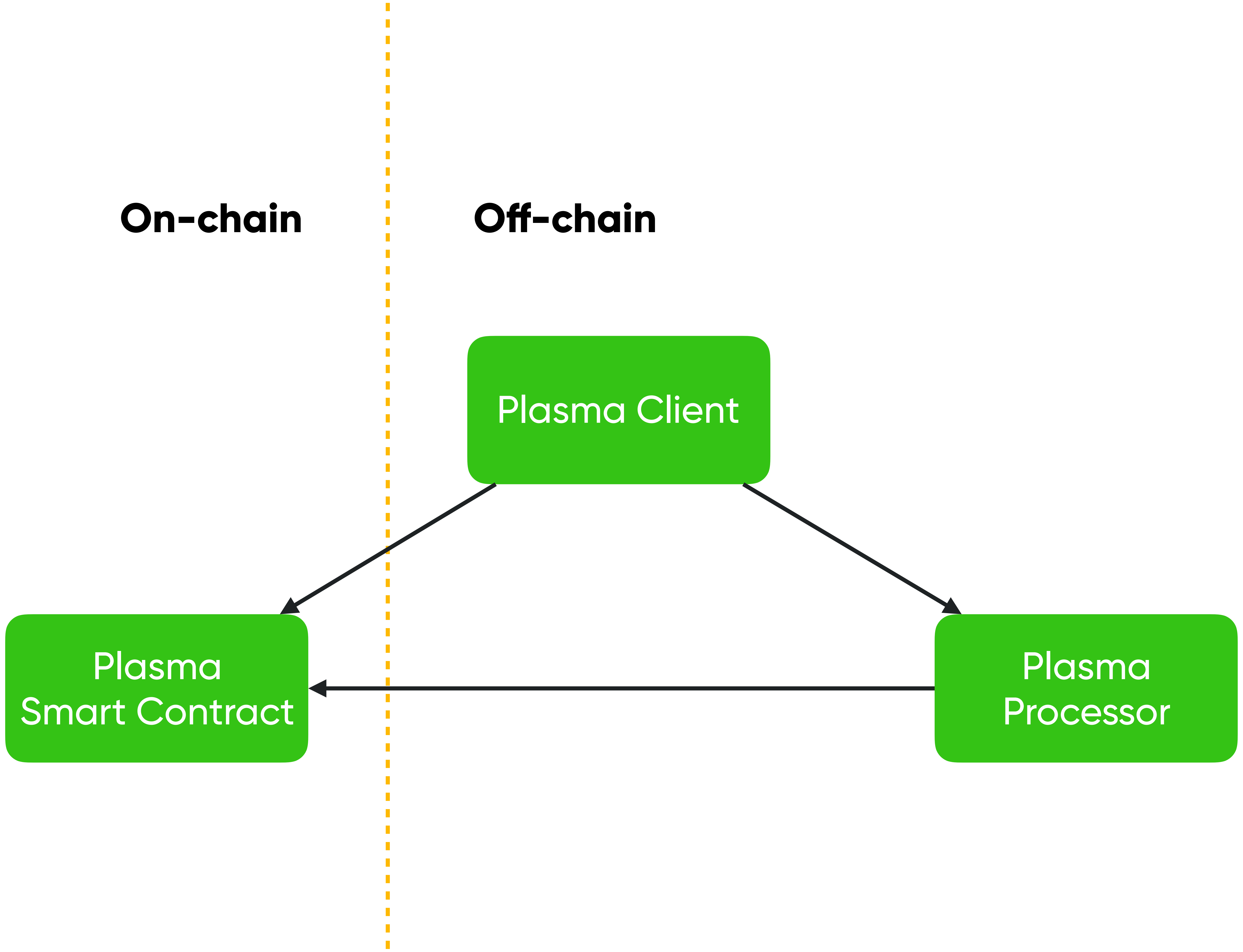
2018

25 TPS

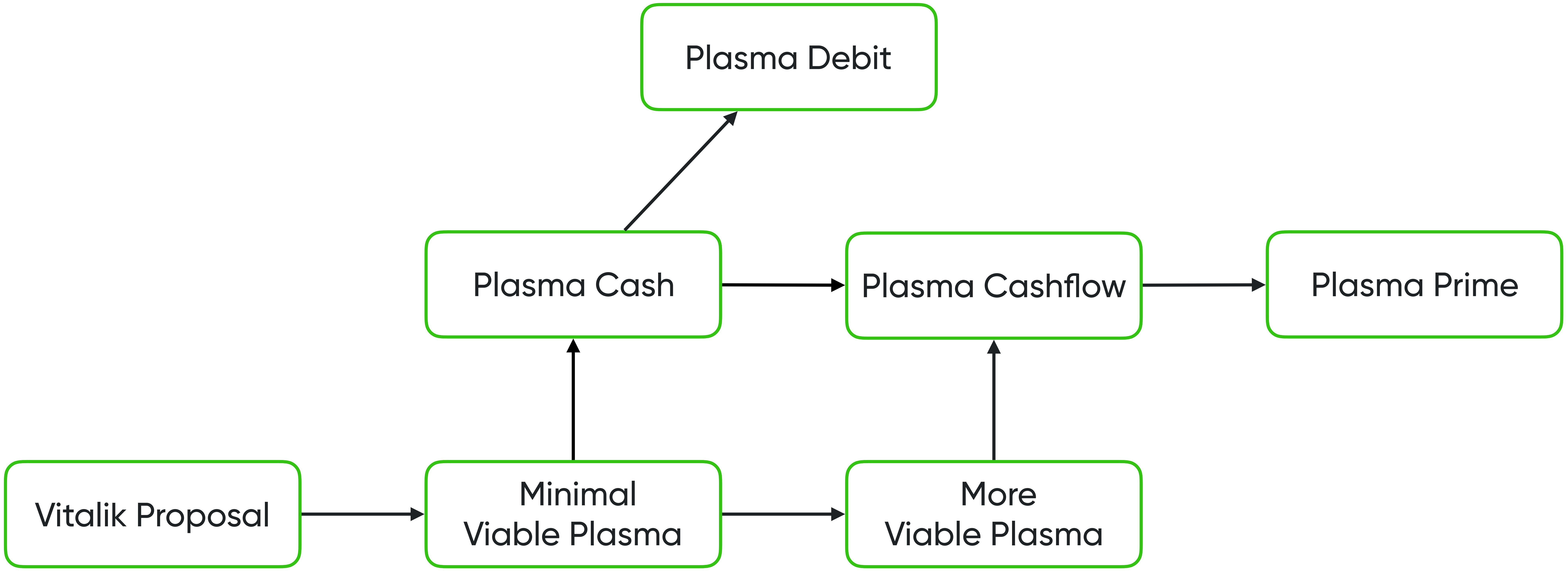
Plasma

Off-chain protocol proposed by Vitalik Buterin

Plasma


On-chain
Off-chain
Plasma Client
Plasma
Smart Contract
Plasma
Processor

Plasma


Plasma Debit
Plasma Cash
Plasma Cashflow
Plasma Prime
Vitalik Proposal
Minimal Viable Plasma
More Viable Plasma

Deposit

1. Any Ethereum Call a smart contract with Sum
2. SC will make a special block
3. Plasma Operator will put this block to the chain

BANKEX FOUNDATION

Transaction

1. Plasma Client creates a transaction, signs it and sends to operator

2. Plasma operator checks it and puts to block

3. Plasma operator push a block root hash to smart contract

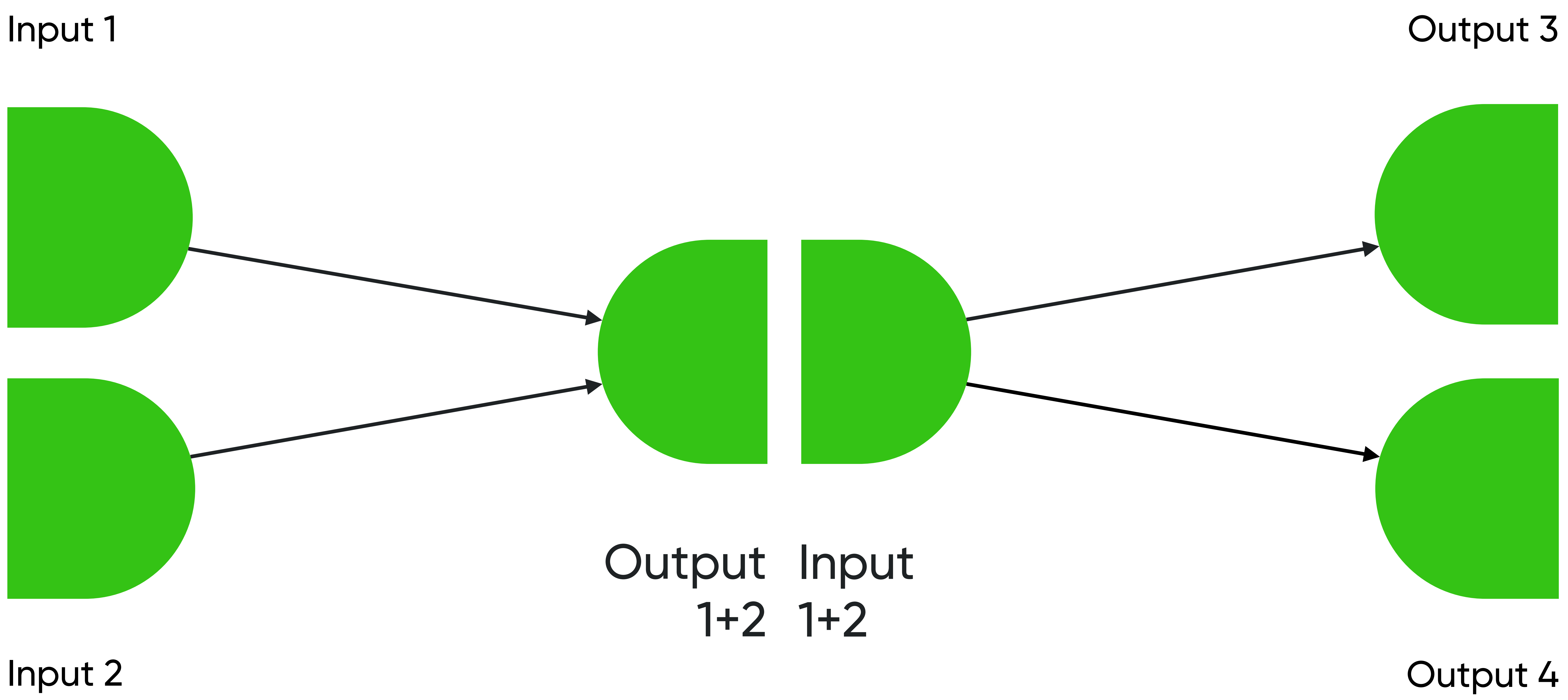
BANKEX
FOUNDATION

Exit

1. Plasma client call an exit method on SC
2. SC needs time to challenge this exit
3. If exit is not challenged - success

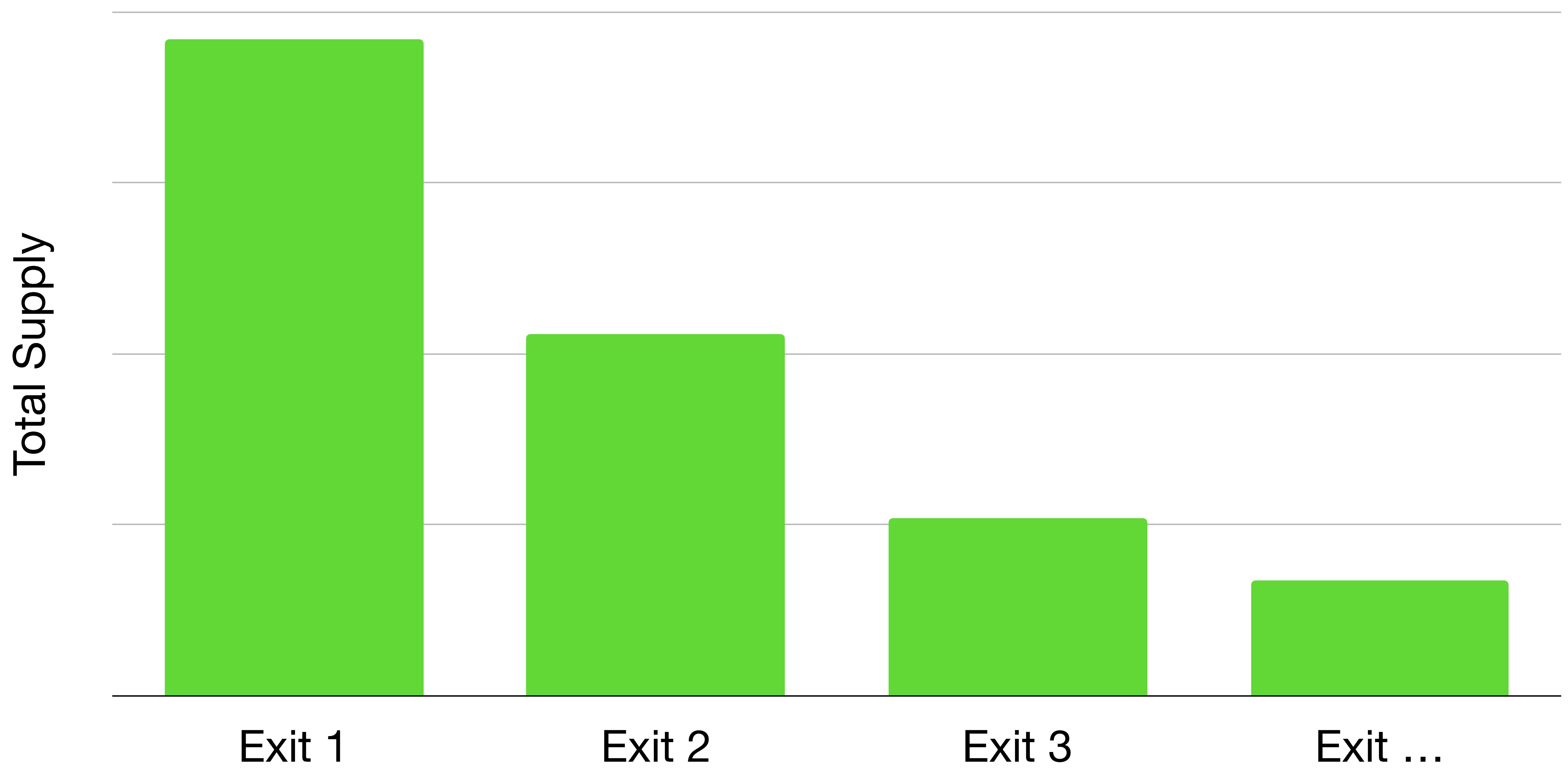
Minimal Viable Plasma

UTXO

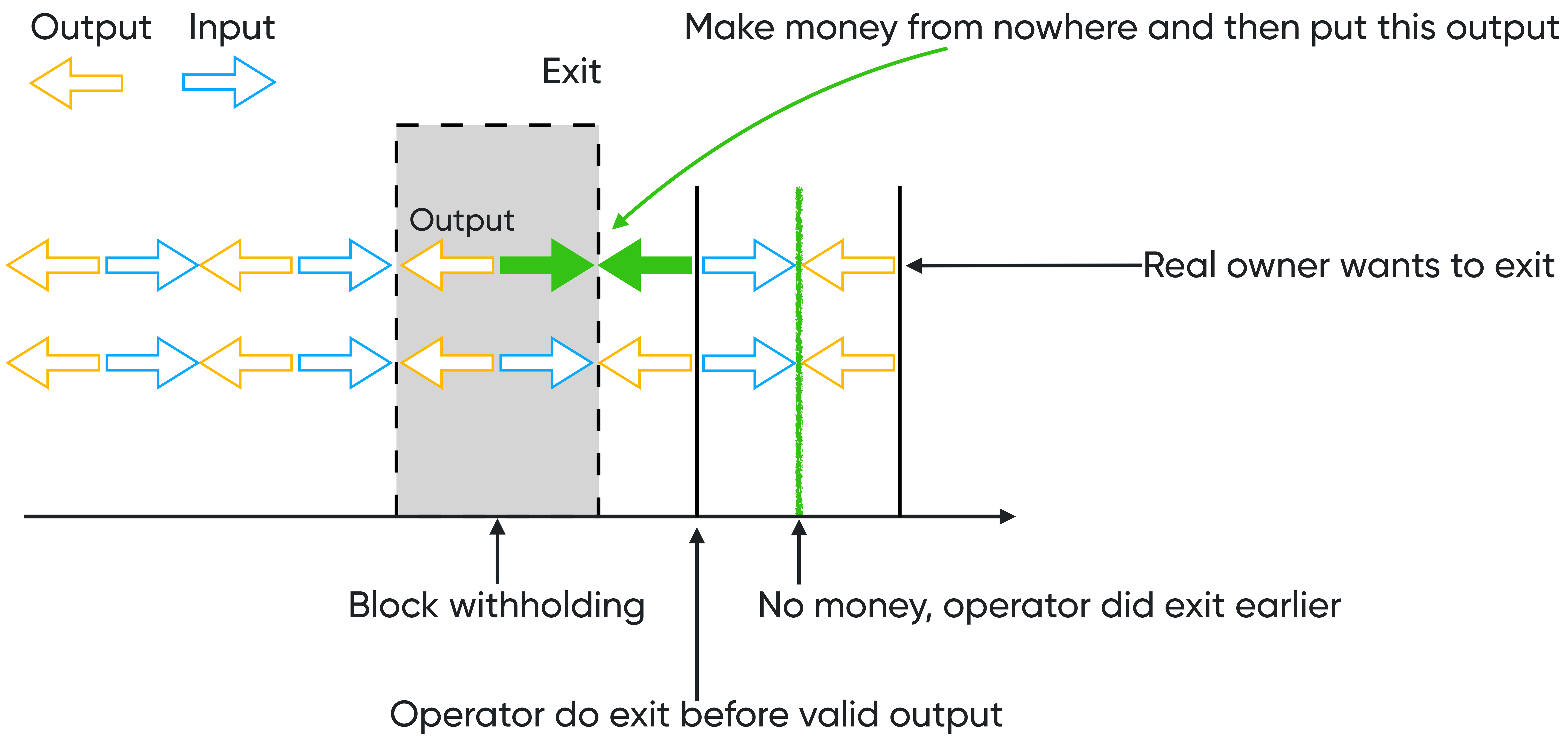


Minimal Viable Plasma

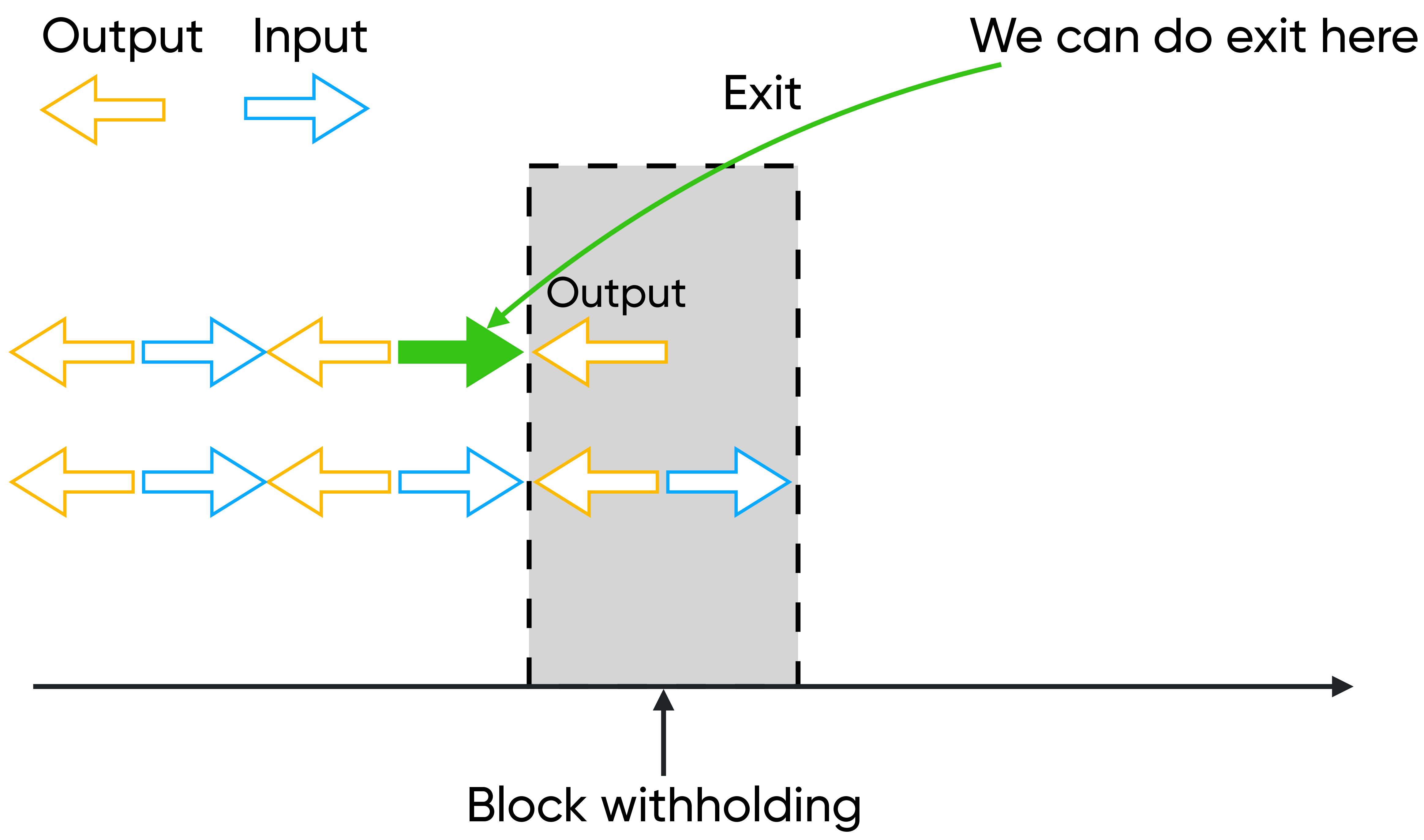
EXIT ORDER



Minimal Viable Plasma



Minimal Viable Plasma



Minimal Viable Plasma

$$competitors(t) = \{t_i : i \in (0, n], I(t_i) \cap I(t) \neq \varnothing\}$$

$$first(T) = t \in T : \forall t' \in T, t \neq t', min(O(t)) < min(O(t'))$$

$$canonical : TX \rightarrow bool$$

$$canonical(t) = (first(competitors(t)) \stackrel{?}{=} t)$$

Minimal Viable Plasma

$$txo(t) = O(t) \cup I(t)$$

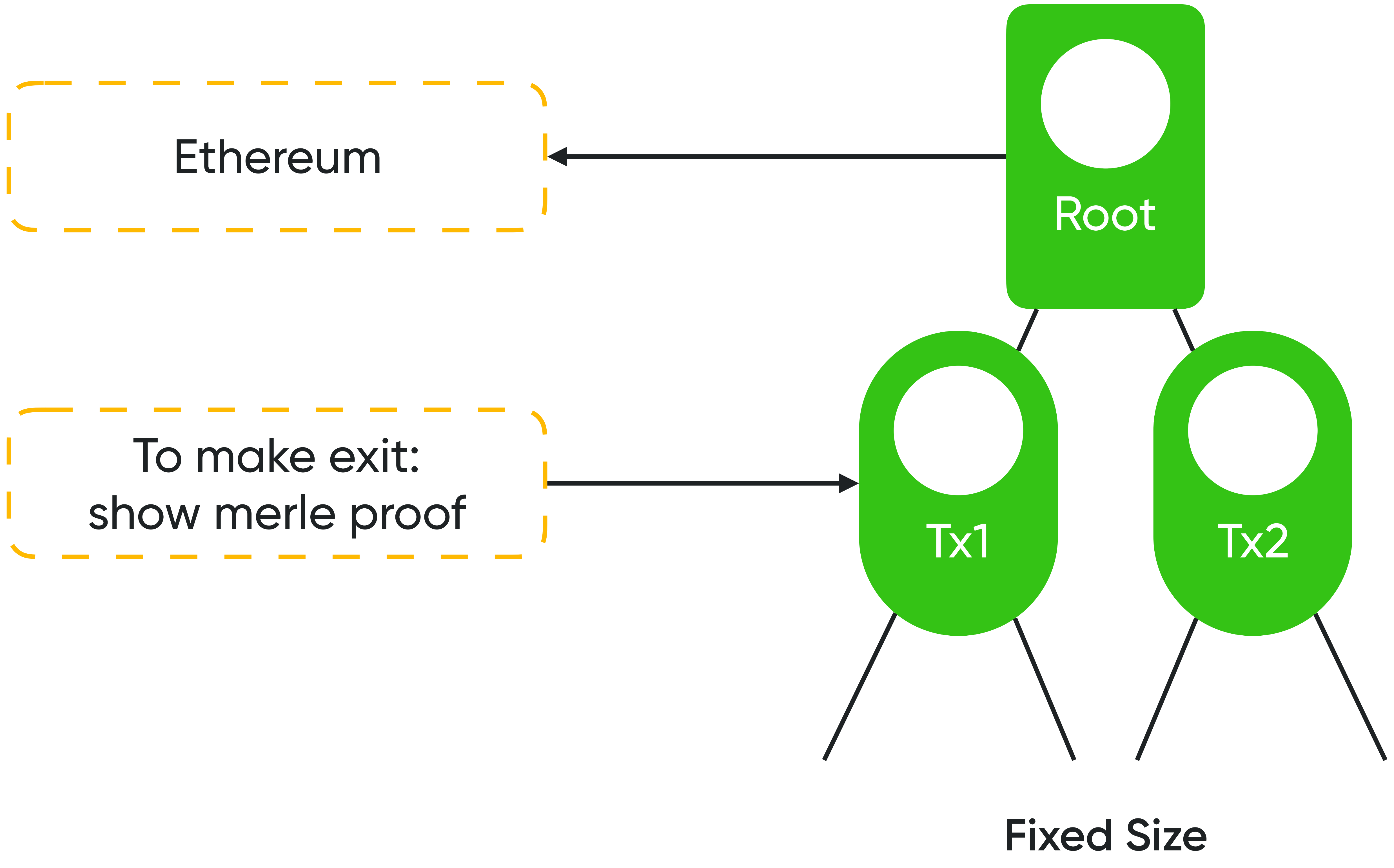
$$TXO(T_n) = \bigcup_{i=1}^{n} txo(t_i)$$

$$unspent(T) = \{o \in TXO(T) : \forall t \in T, o \notin I(t)\}$$

$$double\_spent(T) = \{o \in TXO(T) : \exists t, t' \in T, t \neq t', o \in I(t) \wedge o \in I(t')\}$$

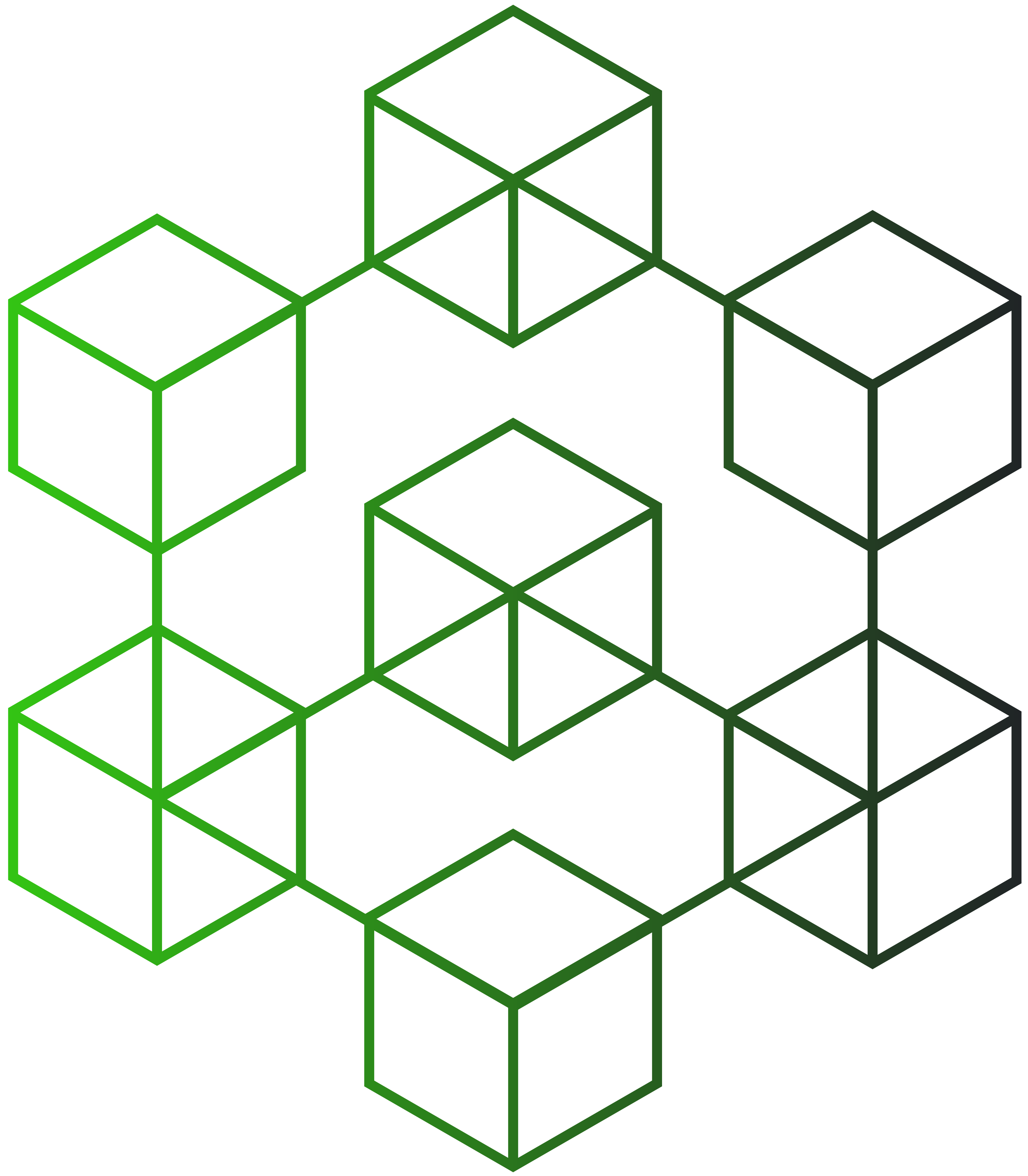
1. $canonical(t)$
2. $o \in unspent(T_n)$
3. $o \notin double\_spent(T_n)$

Plasma Cash

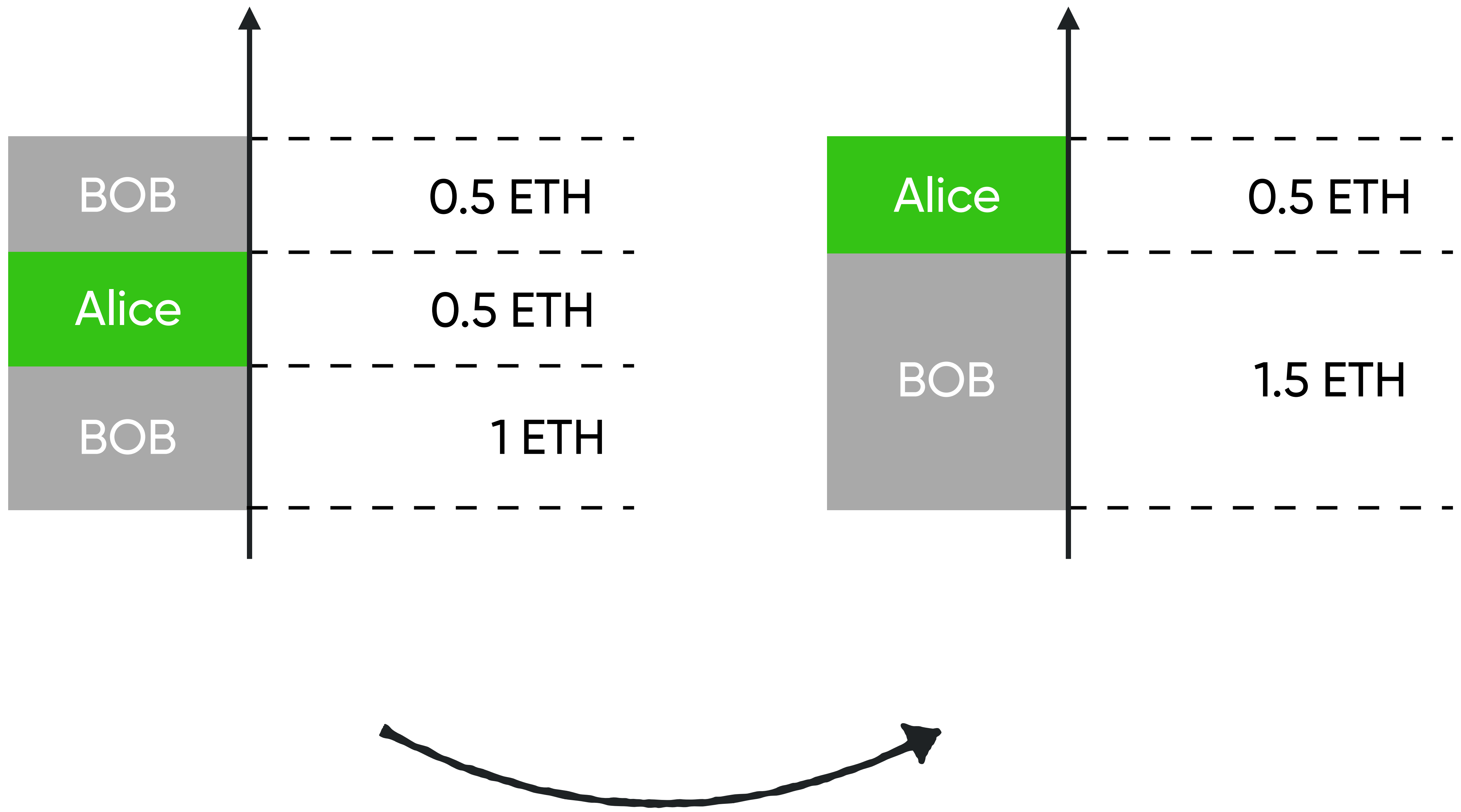

Ethereum
To make exit:
show merle proof
Root
Tx1
Tx2
Fixed Size

Plasma Cash

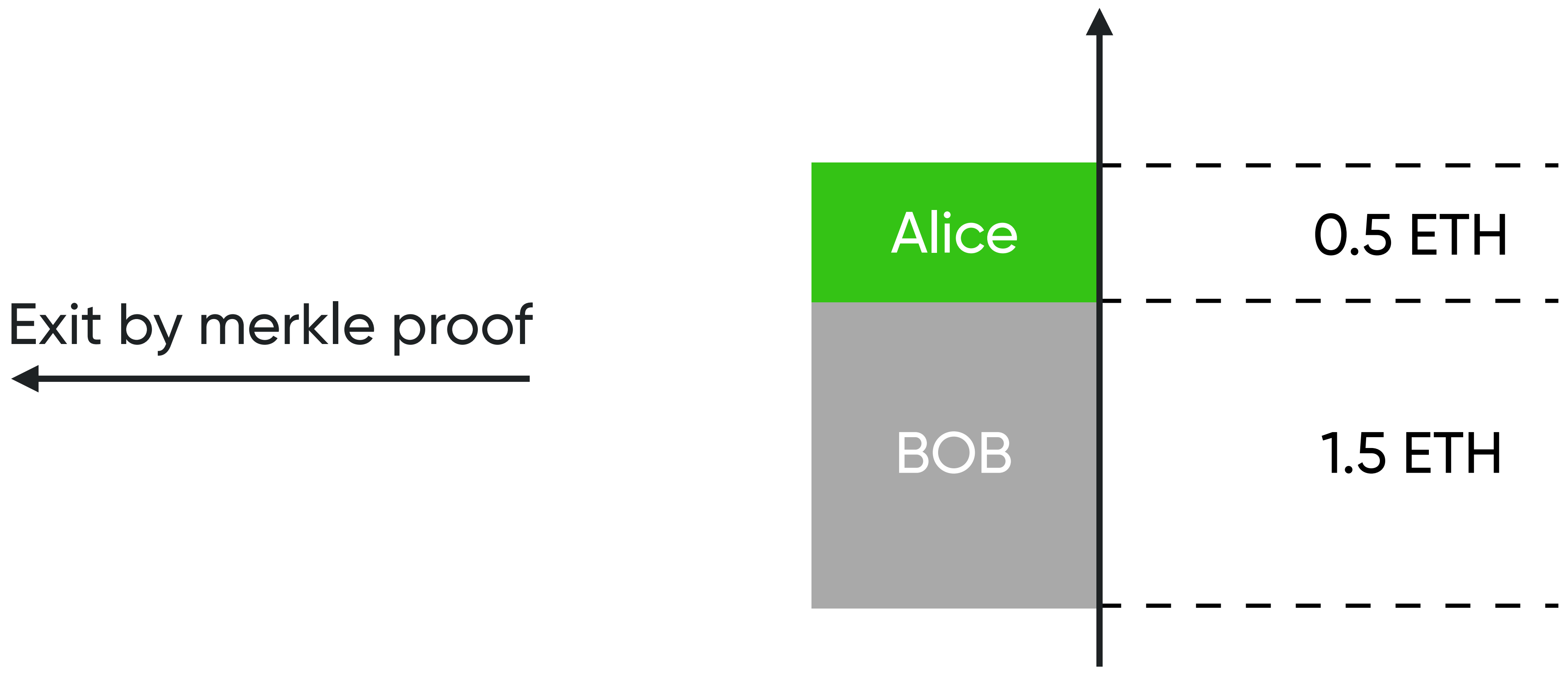
- Exit proofs become really big because you need to provide full tx history
- Token amount is not a small one



Plasma Cashflow



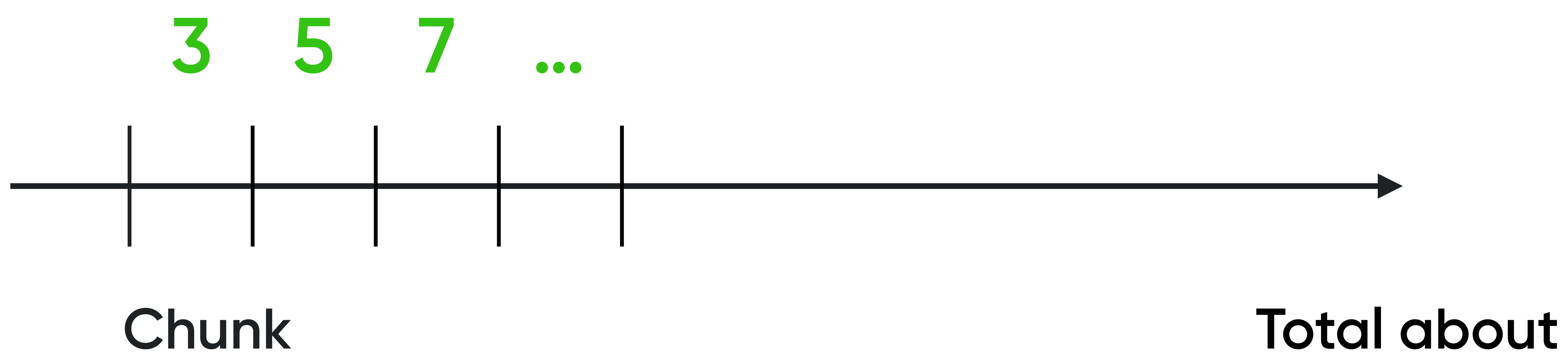
Plasma Cashflow



We sign a FULL transaction, so if plasma operator will try to exit, you can challenge it by earliest unused output

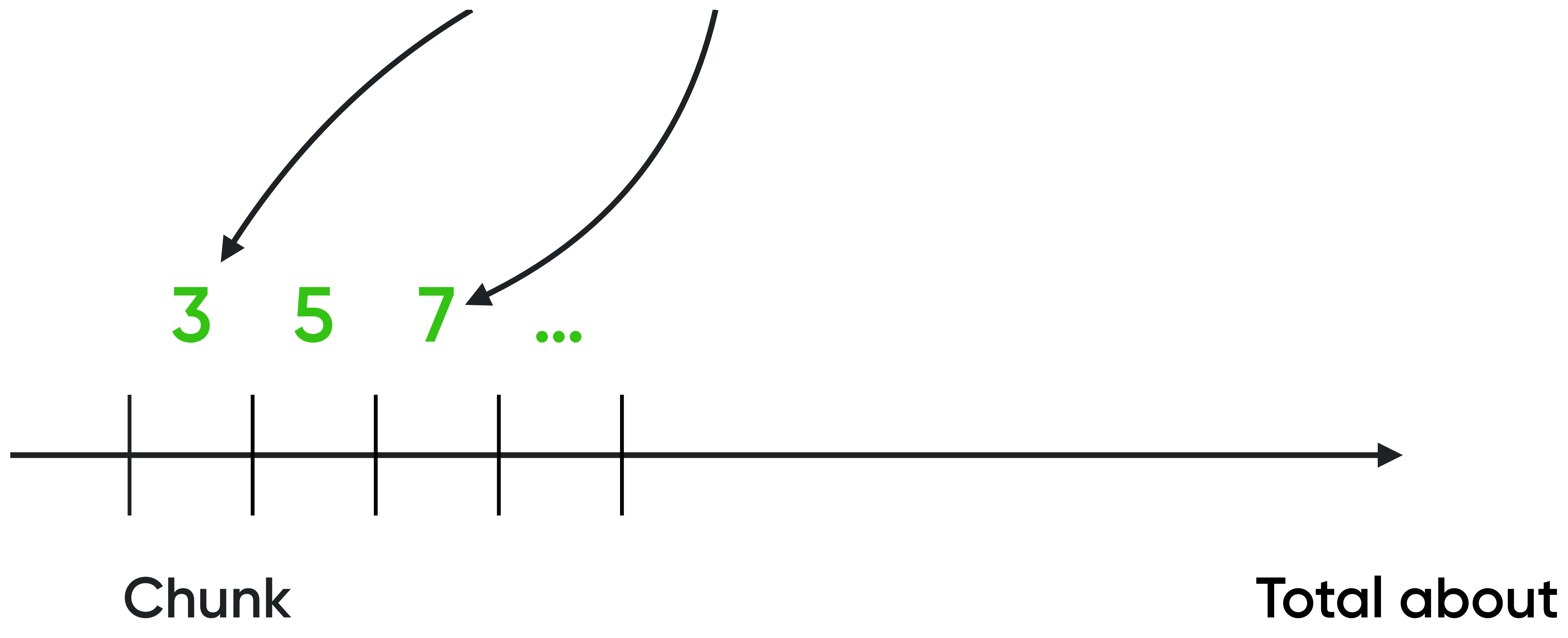
Plasma Prime

Plasma Prime



Plasma Prime

When we are making transactions we are using chunks



RSA Accumulator

Allows to proof that chunk wasn't used before

Block 1	Block 2	Block 3
A1	A2	A3

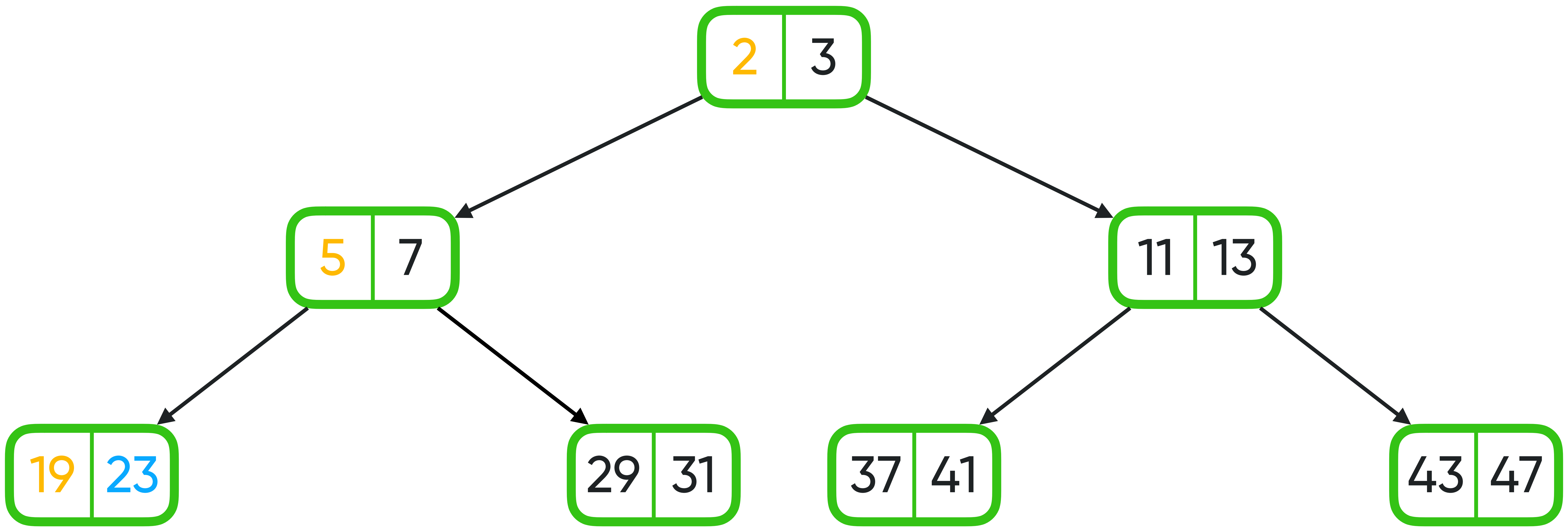
An+1 = An ^ (special prime numbers associated with used chunks)

If

$$\frac{An+k}{An} = 1$$

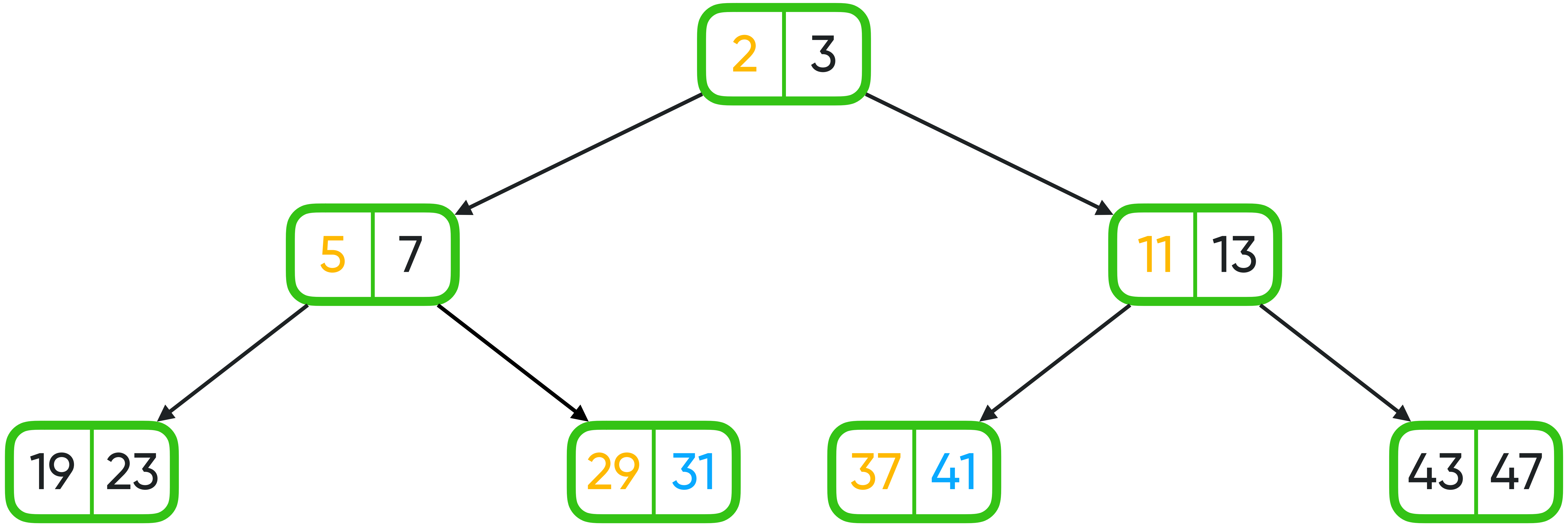
We can proof that chunks wasn't used during k blocks

Plasma Prime



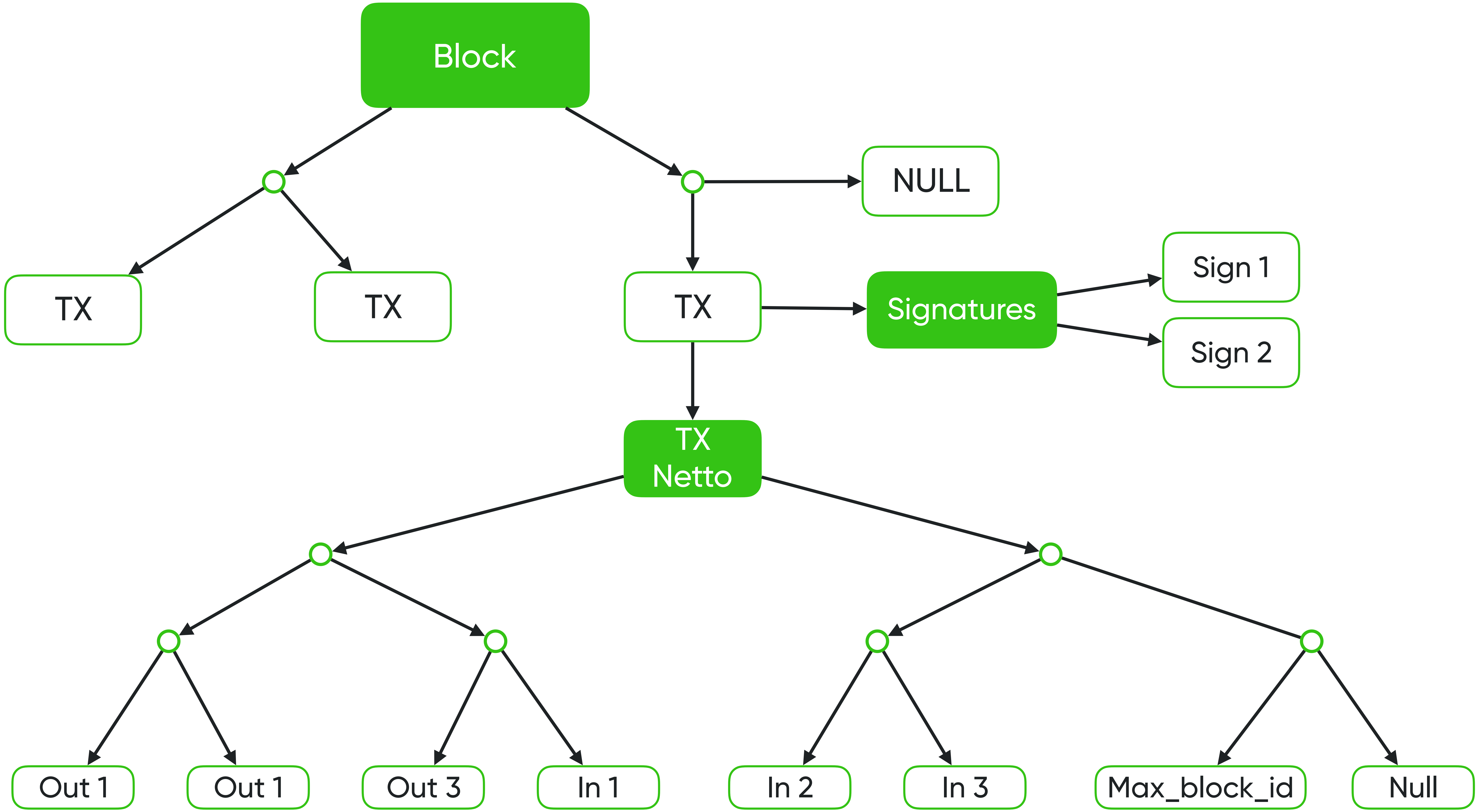
$$A' = A^{19*23*2}$$

Plasma Prime



$$A' = A^{29*31*37*41*5*11*2}$$

Plasma Prime



Plasma based on zk-SNARKS

$$f\,(x0,\, W0) = y$$

W0 – secret
x0 – public

Prover

1. Do f(x0 , W0) = y and get a ProoverKey
2. Send ProoverKey to Verifier

Prover

1. Get a VerificationKey that was created during setup
2. Get a ProoverKey
3. Check that operations are correct

Plasma based on zk-SNARKS

i = (i_1, i_2):

I_1 is Merkle root hash of all valid, spendable utxo of our plasma chain,
i_2 is the list of new deposits into the plasma chain and exits from it.

w = (w_1, w_2):

w_1 is the list of valid spendable utxo,
w_2 is the list of transactions of a plasma block.

o = (o_1, o_2):

o_1 is the Merkle hash of the new valid, spendable utxo after processing the block’s transactions,
o_2 is the Merkle hash of the w_2.

Plasma based on STARKS

✓ Doesn't depends on input or output data

✓ Don't need a trusted setup

✓ Faster proofs

◉ Have a big proof size - cannot be hosted at one ethereum block