Plasma

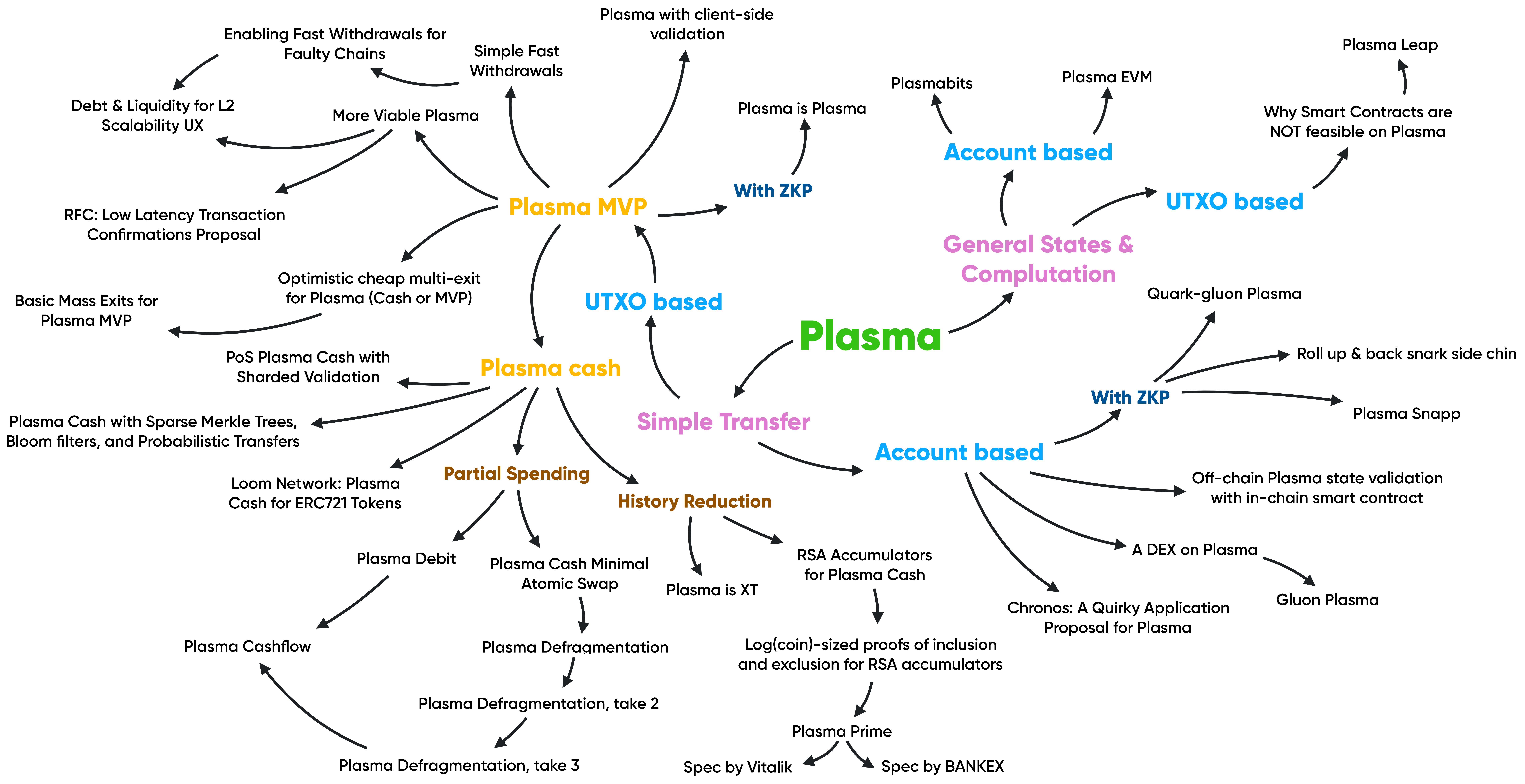
What do we have right now (December 2018)

Exit proof size

Calculations size on SC

TX history size

Plasma World Map



Plasma

Simple Transfer

UTXO based

Plasma MVP

With ZKP

Plasma is Plasma

Plasma with client-side validation

Simple Fast Withdrawals

Enabling Fast Withdrawals for Faulty Chains

Debt & Liquidity for L2 Scalability UX

More Viable Plasma

RFC: Low Latency Transaction Confirmations Proposal

Optimistic cheap multi-exit for Plasma (Cash or MVP)

Basic Mass Exits for Plasma MVP

Plasma cash

PoS Plasma Cash with Sharded Validation

Plasma Cash with Sparse Merkle Trees, Bloom filters, and Probabilistic Transfers

Loom Network: Plasma Cash for ERC721 Tokens

Partial Spending

Plasma Debit

Plasma Cashflow

Plasma Cash Minimal Atomic Swap

Plasma Defragmentation

Plasma Defragmentation, take 2

Plasma Defragmentation, take 3

History Reduction

Plasma is XT

RSA Accumulators for Plasma Cash

Log(coin)-sized proofs of inclusion and exclusion for RSA accumulators

Plasma Prime

Spec by Vitalik

Spec by BANKEX

Account based

With ZKP

Quark-gluon Plasma

Roll up & back snark side chin

Plasma Snapp

Off-chain Plasma state validation with in-chain smart contract

A DEX on Plasma

Gluon Plasma

Chronos: A Quirky Application Proposal for Plasma

General States & Complutation

Account based

Plasmabits

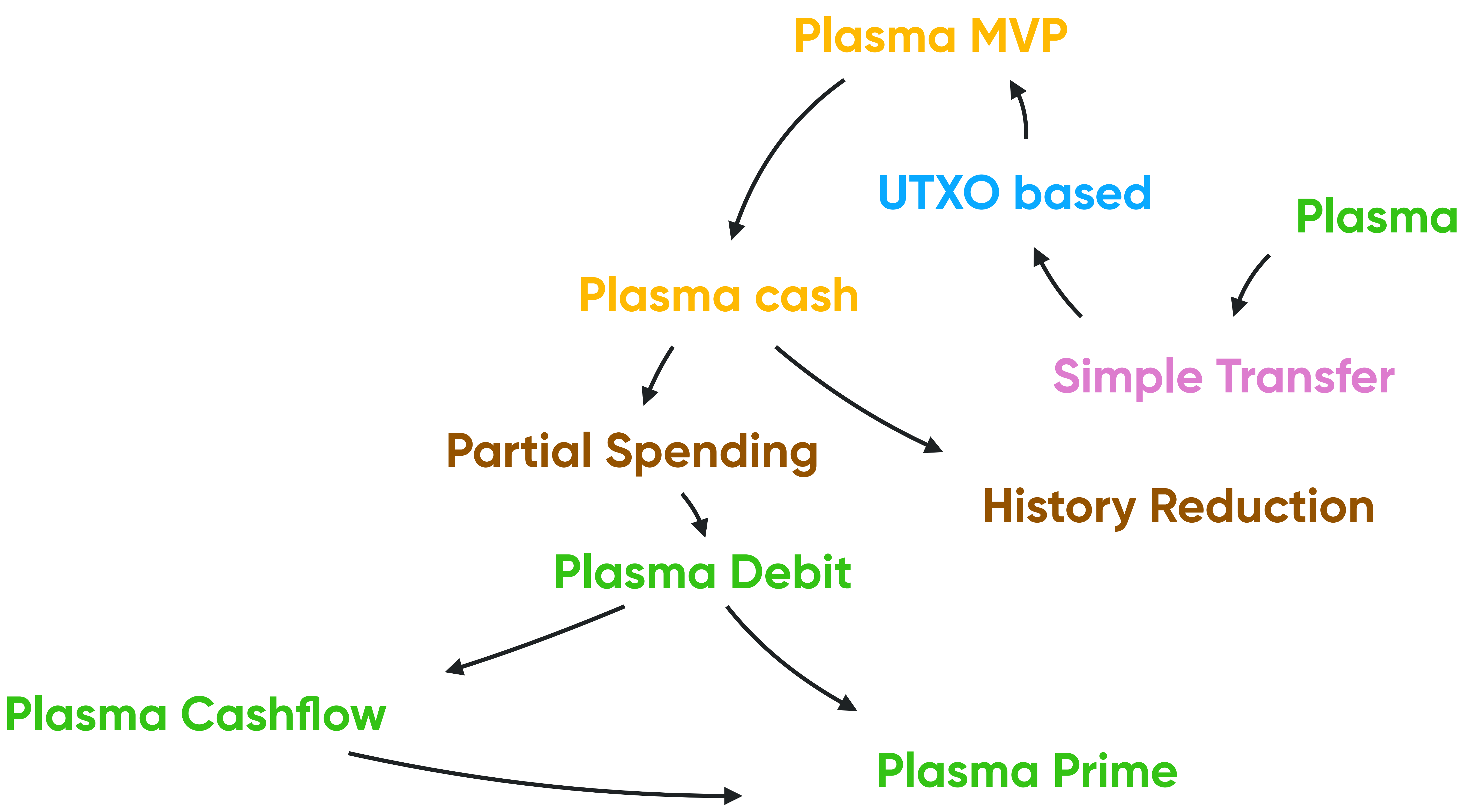
Plasma EVM

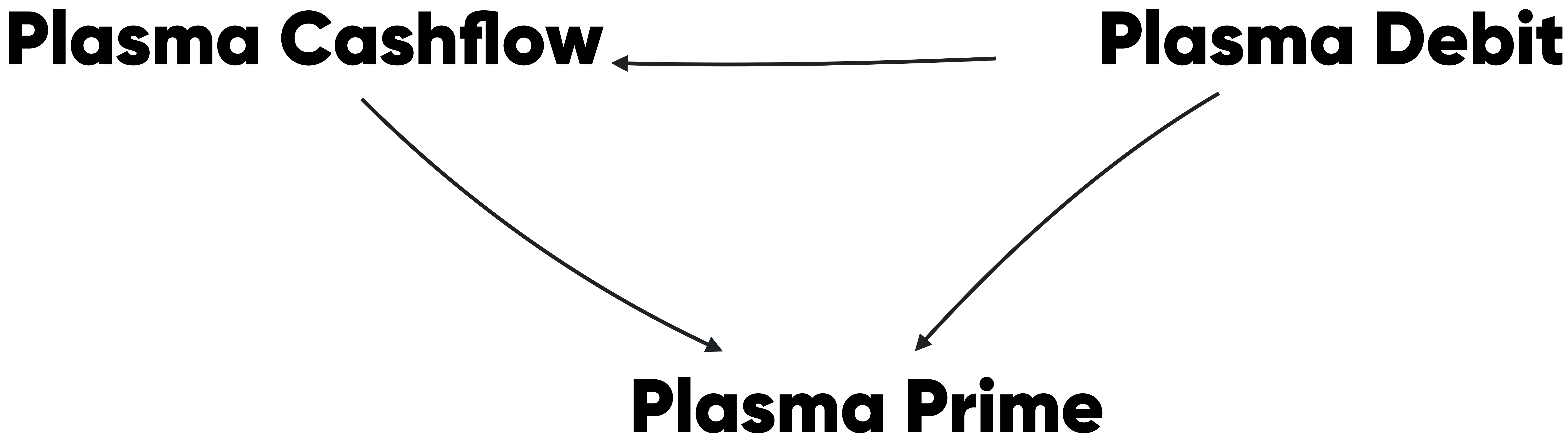
UTXO based

Why Smart Contracts are NOT feasible on Plasma

Plasma Leap

Plasma World Map

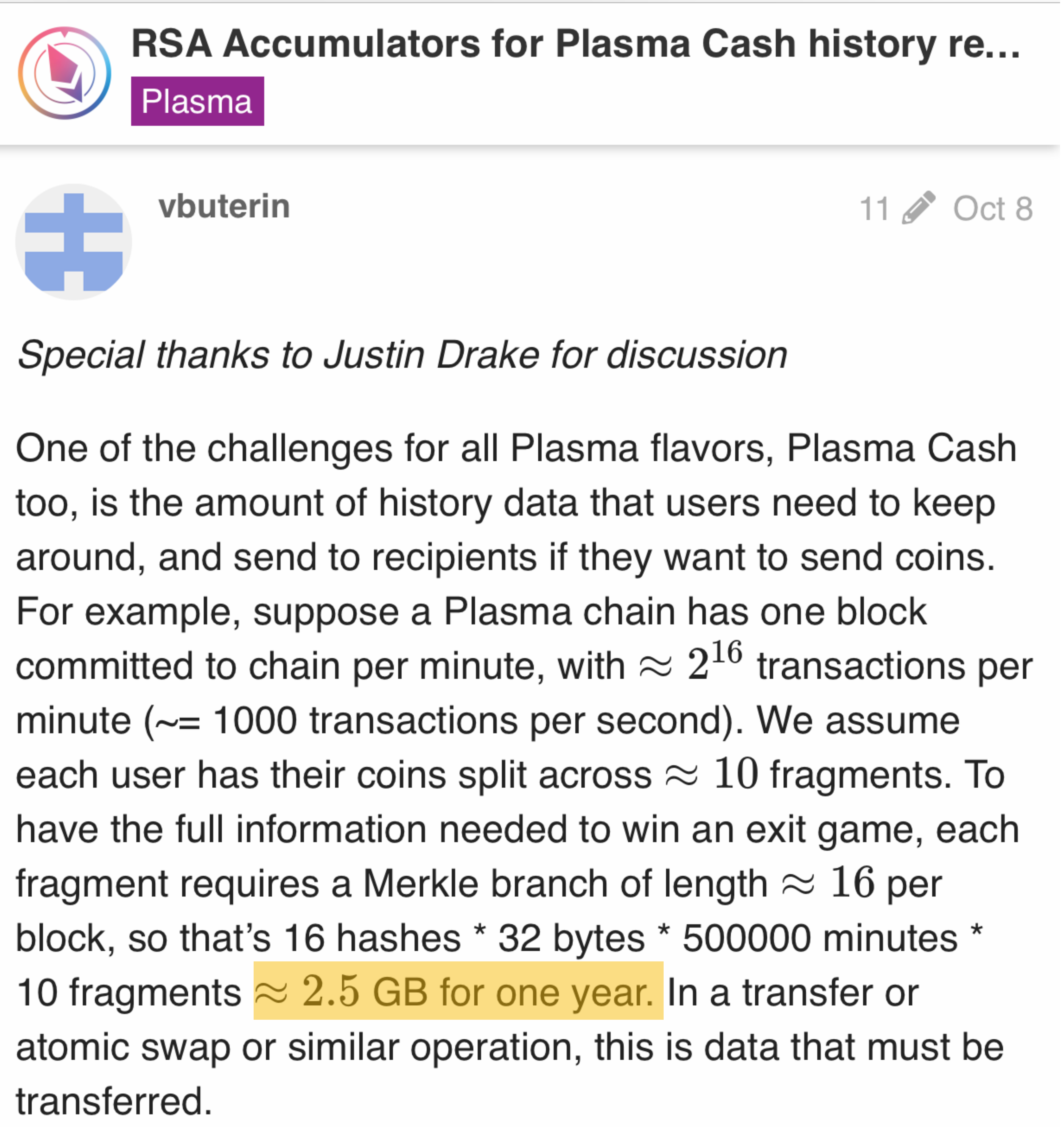




Plasma Cashflow

Plasma Debit

Plasma Prime



RSA Accumulators for Plasma Cash history re…

Plasma

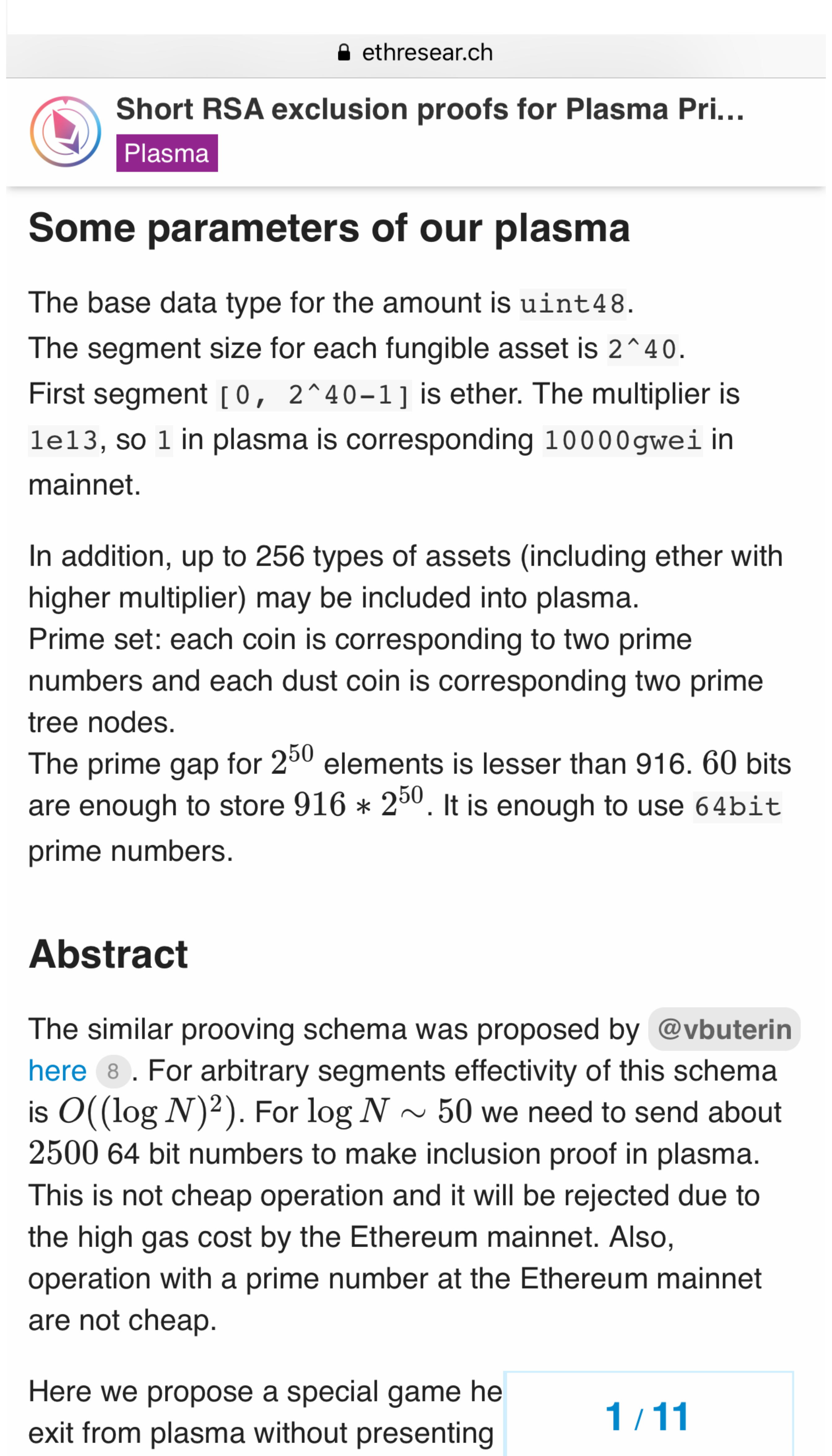
vbuterin 11 Oct 8

Special thanks to Justin Drake for discussion

One of the challenges for all Plasma flavors, Plasma Cash too, is the amount of history data that users need to keep around, and send to recipients if they want to send coins. For example, suppose a Plasma chain has one block committed to chain per minute, with $\approx 2^{16}$ transactions per minute (~= 1000 transactions per second). We assume each user has their coins split across ≈ 10 fragments. To have the full information needed to win an exit game, each fragment requires a Merkle branch of length ≈ 16 per block, so that's 16 hashes * 32 bytes * 500000 minutes * 10 fragments ≈ 2.5 GB for one year. In a transfer or atomic swap or similar operation, this is data that must be transferred.

Proof size about 2.5 GB

– "Short RSA exclusion proofs for Plasma Prime"



ethresear.ch

Short RSA exclusion proofs for Plasma Pri...

Plasma

Some parameters of our plasma

The base data type for the amount is `uint48`.
The segment size for each fungible asset is `2^40`.
First segment `[0, 2^40-1]` is ether. The multiplier is `1e13`, so `1` in plasma is corresponding `10000gwei` in mainnet.

In addition, up to 256 types of assets (including ether with higher multiplier) may be included into plasma.
Prime set: each coin is corresponding to two prime numbers and each dust coin is corresponding two prime tree nodes.
The prime gap for 2^{50} elements is lesser than 916. 60 bits are enough to store $916 * 2^{50}$. It is enough to use `64bit` prime numbers.

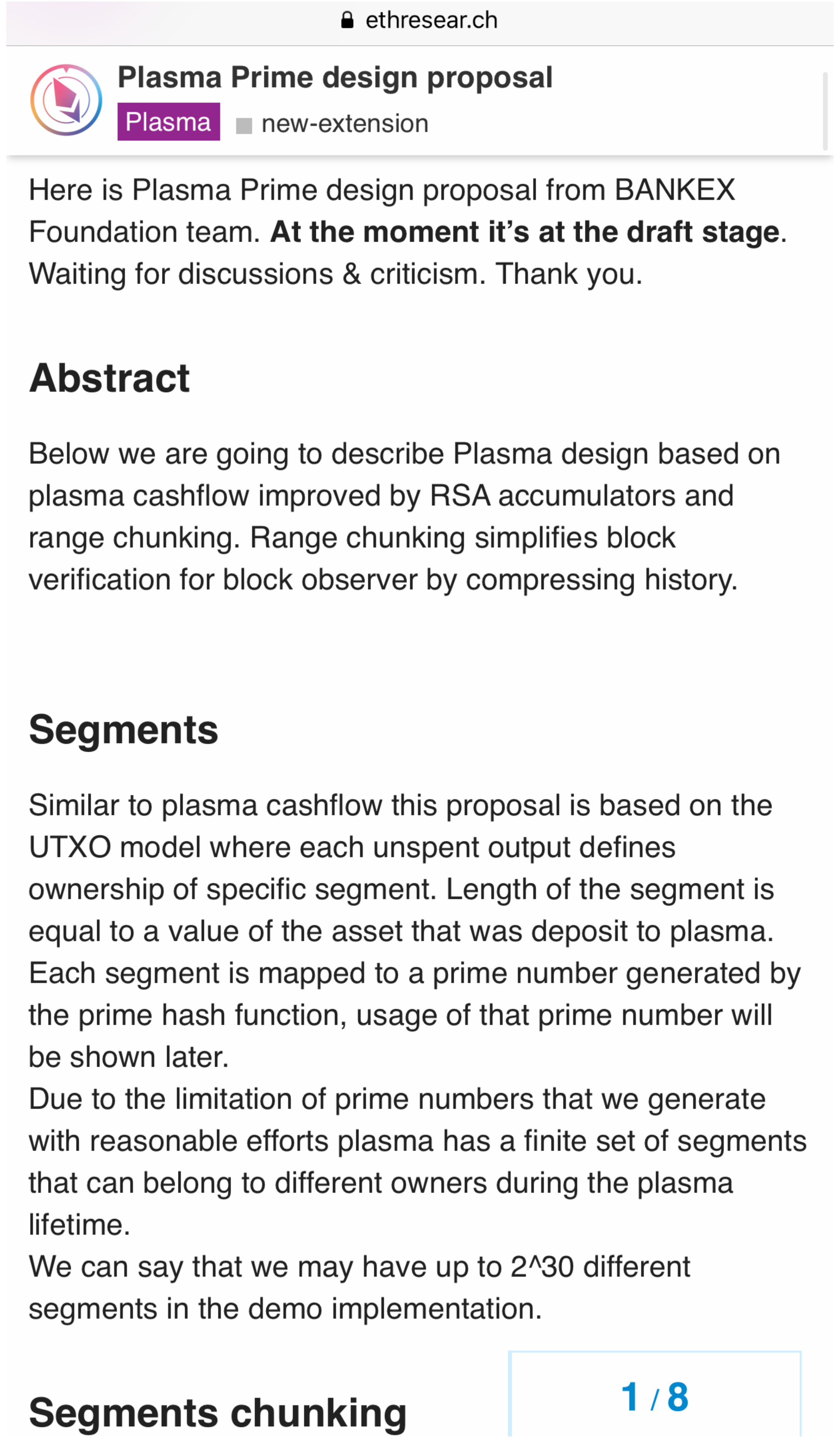
Abstract

The similar prooving schema was proposed by @vbuterin here 8. For arbitrary segments effectivity of this schema is $O((\log N)^2)$. For $\log N \sim 50$ we need to send about 2500 64 bit numbers to make inclusion proof in plasma. This is not cheap operation and it will be rejected due to the high gas cost by the Ethereum mainnet. Also, operation with a prime number at the Ethereum mainnet are not cheap.

Here we propose a special game he
exit from plasma without presenting

1 / 11

– "Plasma Prime design proposal"



ethresear.ch

Plasma Prime design proposal

Plasma ▪ new-extension

Here is Plasma Prime design proposal from BANKEX Foundation team. **At the moment it's at the draft stage**. Waiting for discussions & criticism. Thank you.

Abstract

Below we are going to describe Plasma design based on plasma cashflow improved by RSA accumulators and range chunking. Range chunking simplifies block verification for block observer by compressing history.

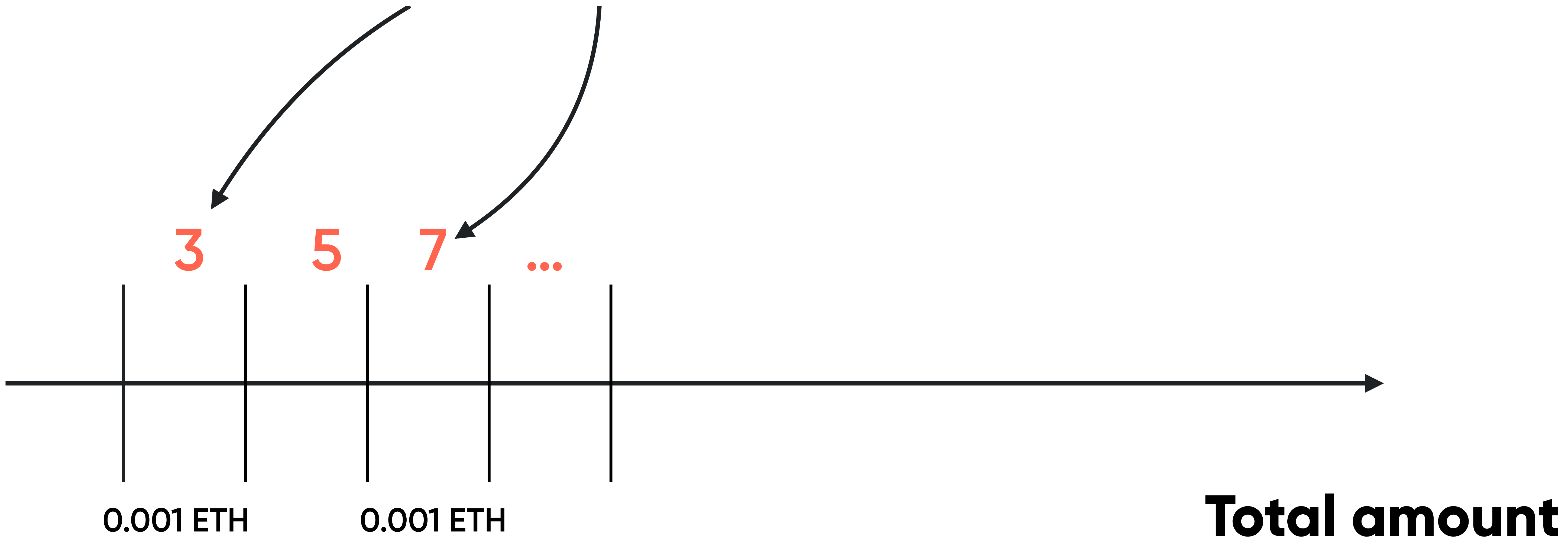
Segments

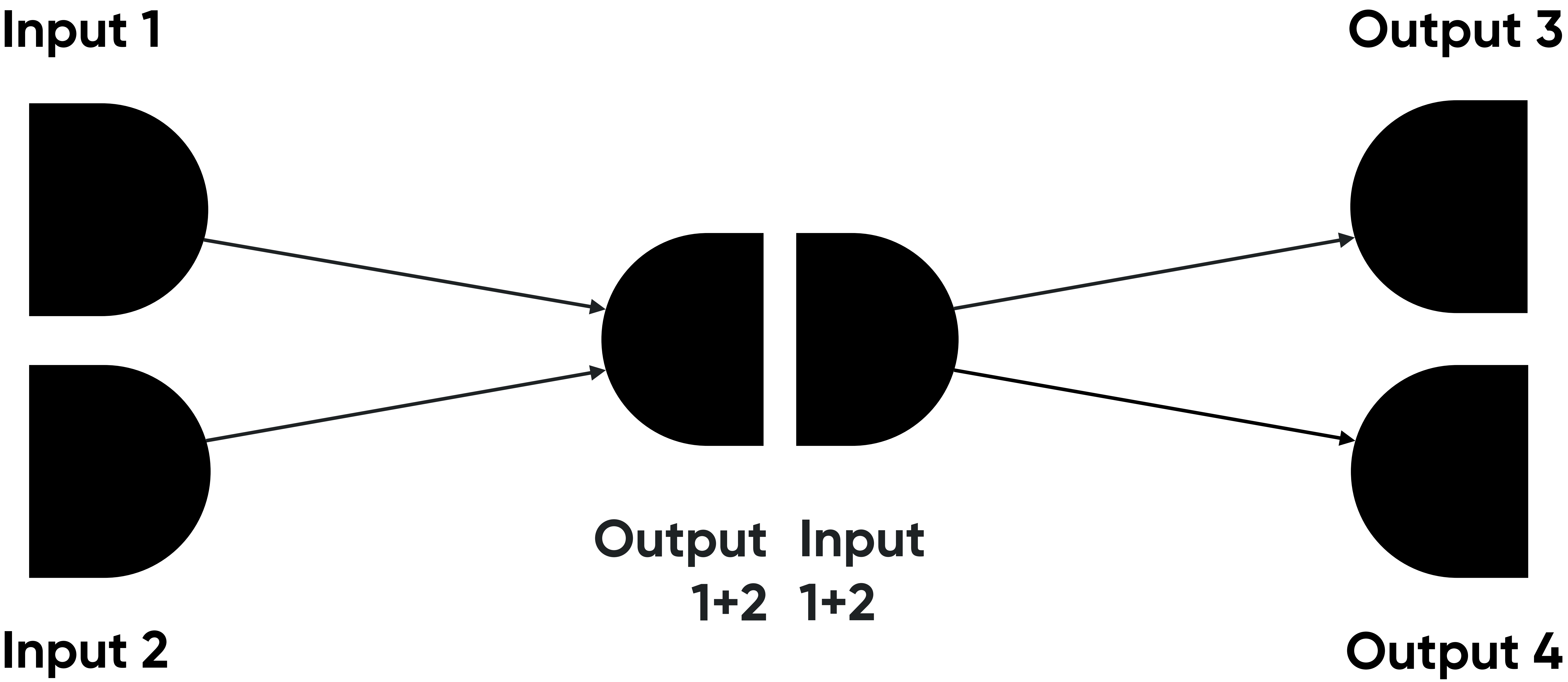
Similar to plasma cashflow this proposal is based on the UTXO model where each unspent output defines ownership of specific segment. Length of the segment is equal to a value of the asset that was deposit to plasma. Each segment is mapped to a prime number generated by the prime hash function, usage of that prime number will be shown later.
Due to the limitation of prime numbers that we generate with reasonable efforts plasma has a finite set of segments that can belong to different owners during the plasma lifetime.
We can say that we may have up to 2^30 different segments in the demo implementation.

Segments chunking

1 / 8

When we are making transactions we are using chunks





Input 1

Input 2

Output 1+2

Input 1+2

Output 3

Output 4

Using RSA Accumulators for exclusion proof

We can proof that we didn’t use specific chunk in block

Problem with calculations
On exit game

We have STARKS

f (x0, W0) = y

W0 - secret
x0 - public

Proving

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

Verification

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