PLASMA

From MVP to general computation

Johann Barbie

parseclabs.org

dapp.acebusters.com

- real time
- secure randomness
- secret state
- low value txns
- liveliness

Requirements for gaming

- High volume
- medium/low value transactions

- Low volume
- High value transactions

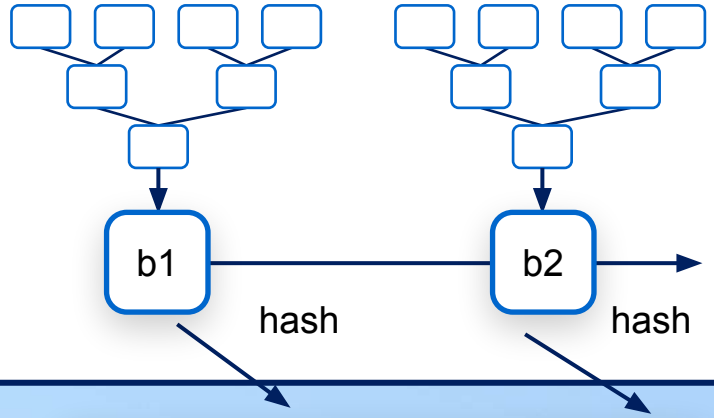
Txns

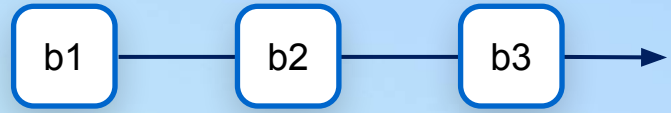
Alice

many transactions

Bob

Plasma Operator





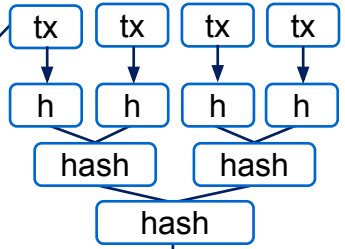
Ethereum Miner

Plasma Deposit

Plasma Operator

Ethereum Miner

PETH

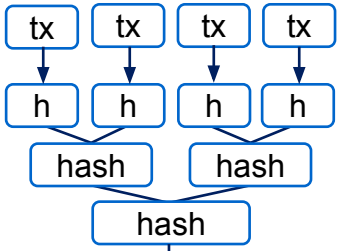


Alice

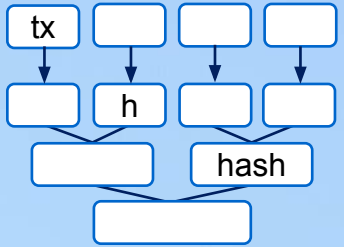
ETH

Plasma Contract

Plasma Exit



merkle proof



Plasma Contract

ETH

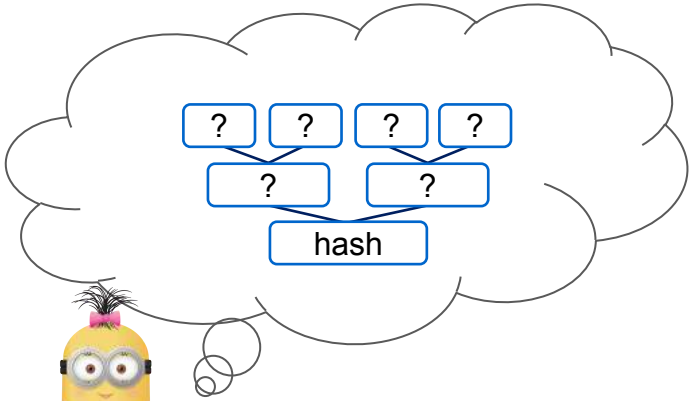
Alice

Ethereum Miner

Data Withholding

Plasma Operator

hash



Alice

Ethereum Miner

Plasma Contract

bogus proof

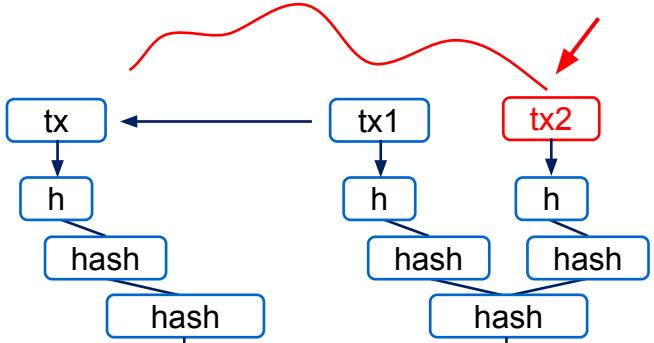
ETH

Operator

Restricted Authority

Plasma Operator

doublespend



Ethereum Miner

Plasma Contract

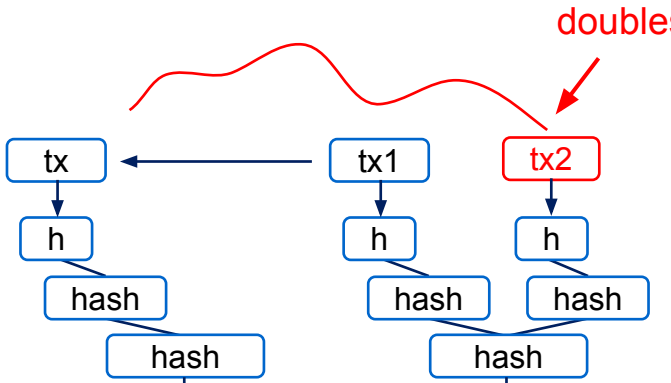
challenge(tx1, tx2, proof1, proof2)

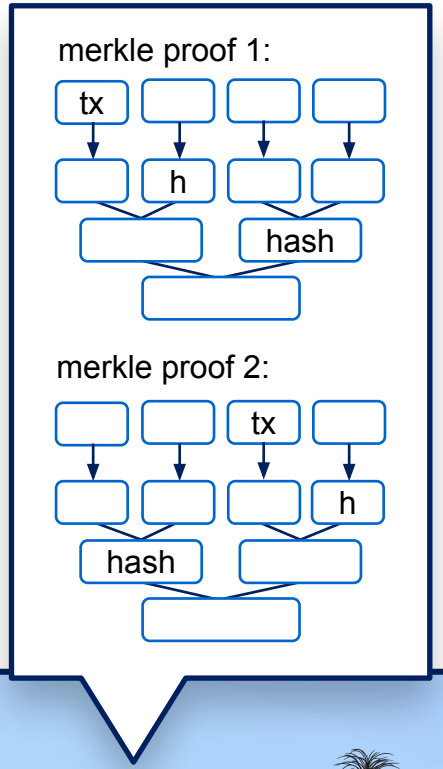
=> revert

Alice

Restricted Authority

Plasma Operator





Ethereum Miner

Plasma Contract

challenge(tx1, tx2, proof1, proof2)

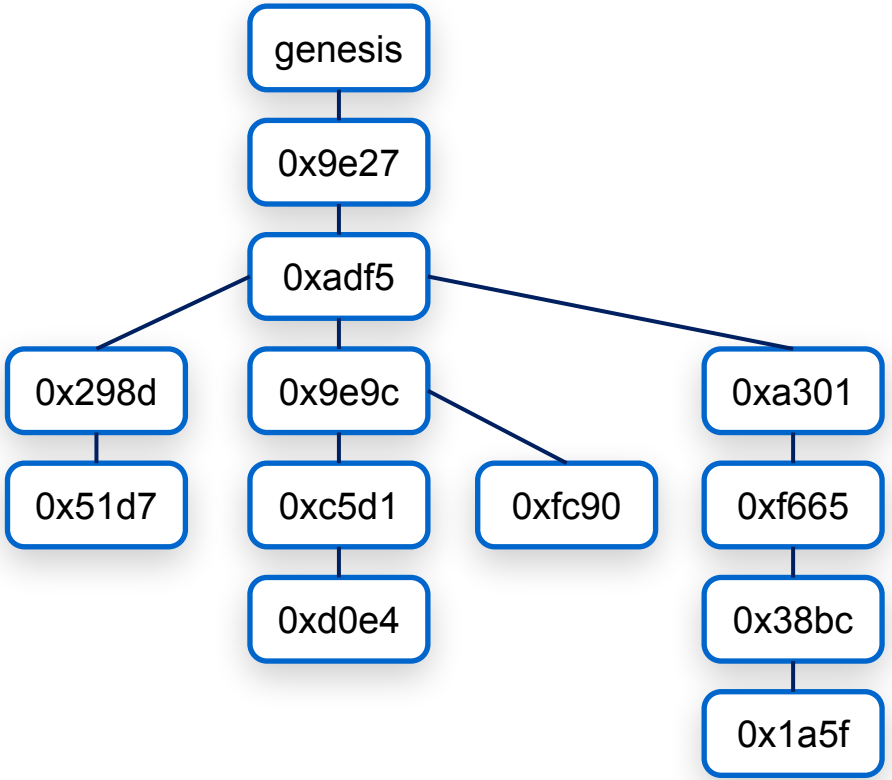
=> revert

Alice

PoS with Nakamoto Incentives

The Nakamoto konsensus incentivises miners to front-run the computation market with block data if they have found a possible solution.

=> creates incentives against data withholding in the whole network



PoS Setup

- the stake is frozen for 3 month.
- the minimum stake amount is 1% of all tokens.
- the maximum stake amount is 5% of all tokens.
- An operator can only propose one block per block height.
- The chain tip is determined by maximum rewards.
- funds are allocated depending on whether the past 100 blocks are representative of all operators.

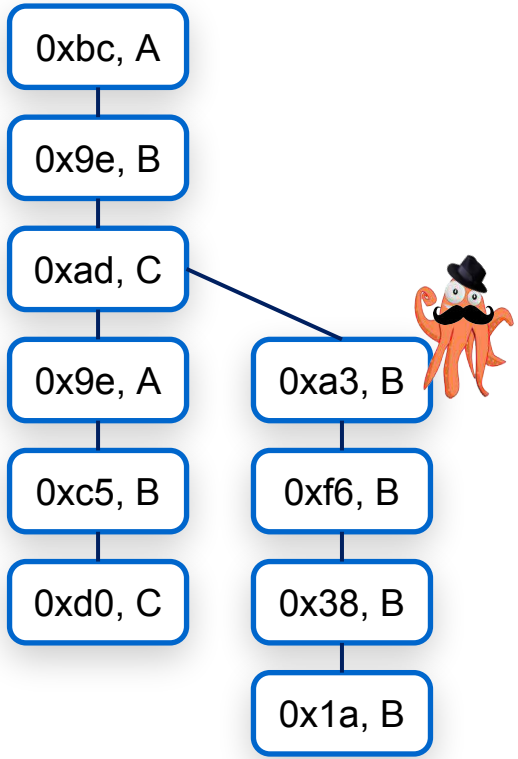
=> dedicated token locks operators into economic union.

Example

- consensus window 6 blocks
- Operators A, B, C each 33%

=> tip **0xd0** has weight of 6

=> tip **0x1a** has weight of 4



Getting Rid of Confirmations

Rule 1: A transaction must be included within two blocks of the time it was created.

Rule 2: A transactions inputs must be created at least 3 child chains blocks before.

source: David Knott https://hackmd.io/o16IqtiJSgG2ez5w9Ug5aw

Transfer without commitments:

t+0: user A spends UTXO_1 to user S in TX creating UTXO_2

t+1: operator O mines an invalid block B_0, submits the block hash, but doesn’t share the data.

t+2: operator starts exiting invalid utxos on parent

t+2: user S notices block withholding, but can not exit UTXO_2, as he doesn't know the position of TX in the invalid block

t+2: user A notices block withholding, tries to exit with UTXO_1, but can be challenged by O with TX (which O knows position of).

Transfer with commitments:

t+0: user A spends UTXO_1 to user S in TX_1 creating UTXO_2

t+1: operator O includes TX_1 in block B_1 and publishes

t+2: A sees hash in root chain, and signes commitment C for S

t+3: S spends UTXO_2 in TX_2 creating UTXO_3 spendable by A

t+4: O creates an invalid block B_2 with a TX_2 where receiver is O and withholds it

t+5: operator can not exit this transaction, as a C is needed signed by S t+5: S notices lack of block and starts exiting using TX_1 and C

Transfer with David Knott's rules:

t+0: user A spends UTXO_1 in TX_1 to user S, creating UTXO_2

t+1: operator O withholds TX_1

t+2: operator creates an invalid block B and doesn't publish it, but submits hash

t+2: S notices lack of B data, but can not exit with UTXO_2 as position in B unknown

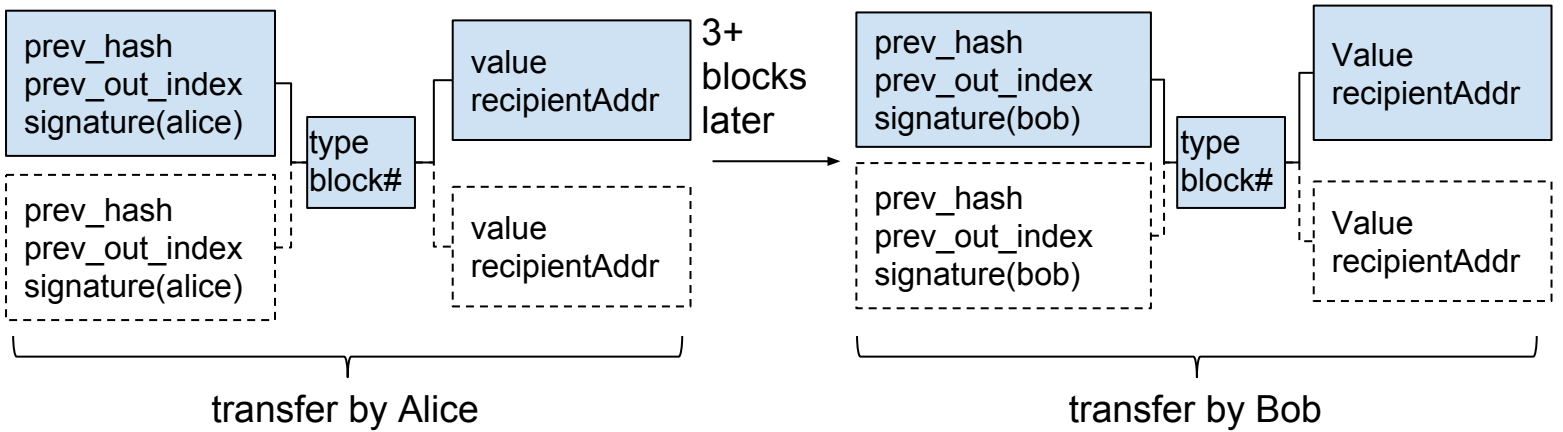
t+2: A starts to exit UTXO_1, last output with known position

t+3: O mines a block including TX_1

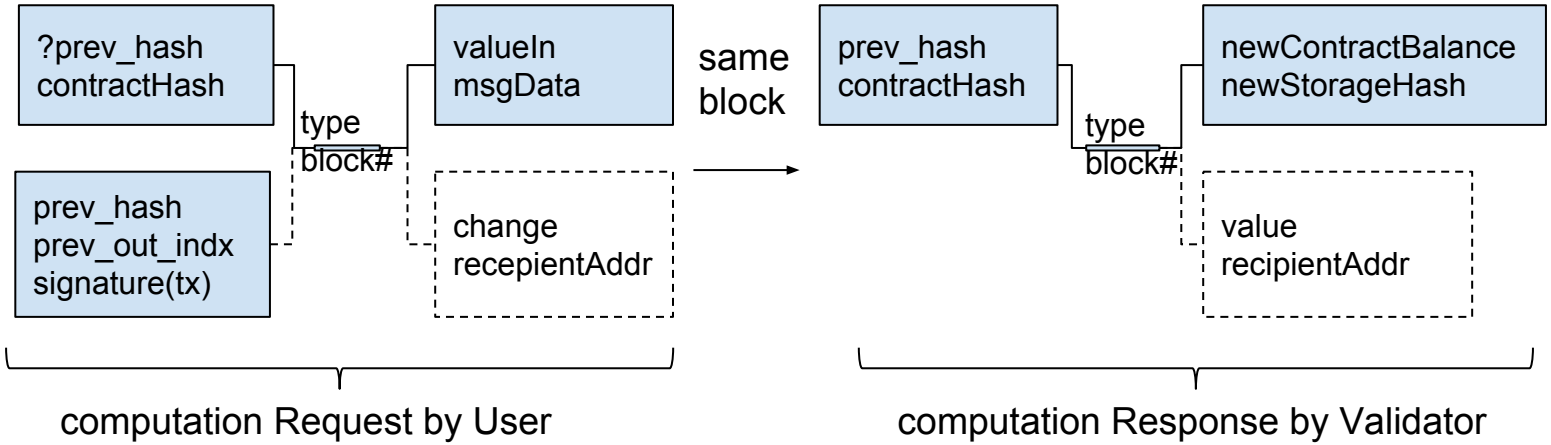
t+4: O challenges with UTXO_1 within 4 days

t+4: Now S knows position of TX_1 from O’s challenge, and can rechallenge with UTXO_2

Transfer Transaction

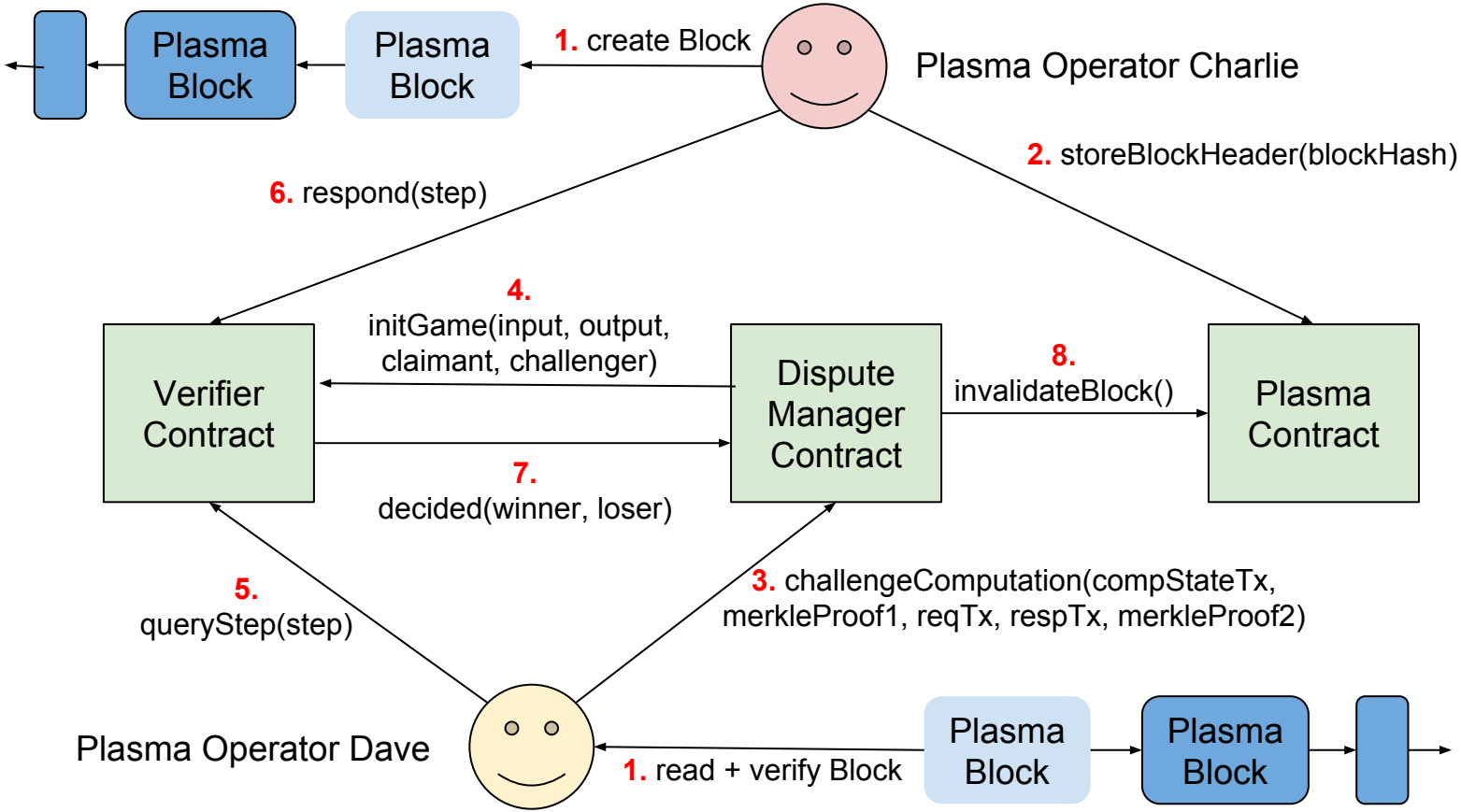


Adding Computation



Truebit Verification Game

Source: https://medium.com/truebit/truebit-the-marketplace-for-verifiable-computation-f51d1726798f - Sina Habibian


Plasma Block
Plasma Block
1. create Block
Plasma Operator Charlie
2. storeBlockHeader(blockHash)
6. respond(step)
4.
initGame(input, output, claimant, challenger)
Verifier Contract
Dispute Manager Contract
8.
invalidateBlock()
Plasma Contract
7.
decided(winner, loser)
5.
queryStep(step)
3. challengeComputation(compStateTx, merkleProof1, reqTx, respTx, merkleProof2)
Plasma Operator Dave
1. read + verify Block
Plasma Block
Plasma Block