Plasma on Bitcoin:

What It Could Look Like

https://plasma.io/
Joseph Poon <joseph@lightning.network>
Scaling Bitcoin 2017-11-05

Context

Nothing in This Presentation is Prescriptive. Plasma is very much in development, including refining the architecture, therefore this isn't a presentation about how things should be, but more about how things could be

Not a Formal Proposal to Change Bitcoin. Not proposing this will go into Bitcoin anytime soon, but the goal is to have experiments of how to make Plasma work with Bitcoin and UTXO formats. Could be experimented within sidechains

Second Layer is More Than Payment Channels. This presentation is more about exploration around cryptoeconomics and its relationship with data availability

Bitcoin Specific Properties. Will give an overview of Plasma, with bitcoin-specific properties and how it affects Plasma's design, not a presentation on Rootstock

Blockchain Scaling: Security and Scalability

Security. Security of blockchains are derived from everyone computing the same thing at once

Self-validated assurance. Don’t outsource assurance to others, fully validate all relevant activity yourself, e.g. chain of payments.

Ledger Scalability. The capacity of the UTXO set is necessarily constrained, but results in limitations with the number of participants (and combinations of participants with multisig outputs), even with second layer payments which helps scale transaction volume.

Minimize Ledger State Storage

Minimize global validation. If the blockchain is the ground truth, then we should minimize the amount of ground truth being validated

Disagreement around global consensus is insanity. Minimizing the amount of assertions, especially with incomplete information runs the risk of consensus faults

State synchronization across participants

Need to enforce current state, with global data it’s easy

Withdrawals are the most complex aspect

Data availability is the hardest part. Making proofs require evidence. If the blockchain is the adjudication layer, evidence is the foundation for attestation

Intuition Around the Channels and Lightning Network

Background. Not going over how LN works, but is foundational to understanding the mechanisms for Plasma

Off-chain activity, on-chain enforcement. All activity is bonded on-chain and is ultimately enforceable if there's any disagreement

> Infinite transactions between two parties, but only net-settle current state
>
> A network of these channels can create contingent payments across multiple participants

Two-party synchronicity. Two parties agree on the current state before moving forward. Since both parties need the data to move forward, data availability for those two parties.

Increasing Participants Increases Complexity

Multiparty output enforcement. Two parties can come to agreement, but for ledger state, that requires many parties to come to agreement

Multiparty liveness constraints. If you're dealing with hundreds of people, not all of them will be online. Not as significant of an issue with payments over LN, but definitely a problem when thousands of people are in a shared ledger with synchronicity requirements

Incentives against halting. With computation, there is increased incentives towards halting the chain or acting byzantine

Scaling the Ledger. Channels require on-blockchain transactions for data availability

The data availability problem permeates all of these issues.

Possible Properties of Plasma on Bitcoin

Blockchains in Blockchains. Plasma is a construction of blockchains in blockchains. State of these child blockchains are committed to the root chain

Fraud Proofs. If an incorrect state is committed, anyone else can submit proof to a parent/root chain and disagree

A set of opcodes and outputs. A set of opcodes determine fraud proofs, outputs constantly updated. Flexible output signing for malleability

Helps With Some Second Layer Security. Worst-case reduces m-of-n constructions to 1-of-n in the event of faulty behavior

Compatible with Lightning. More ledger availability improves reliability and performance

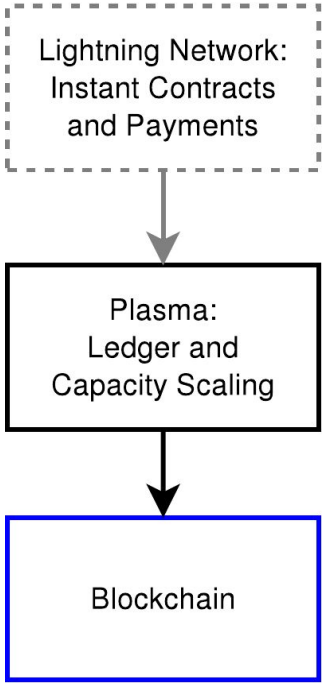
Plasma as an Intermediate Layer

Plasma is not fast. Relies on Lightning for fast transactions. Lightning may likely be the direct interface for clients, with lower levels behind the scenes

Runs between Lightning and the base layer chain. Build layers of blockchain scaling

Ledger Capacity. Increases the capacity to hold the UTXO set and increases the ability to close out transactions at once. Greater flexibility for many Lightning channel exits.

Ultimately secured by the underlying chain. Both Lightning and Plasma rely upon the underlying chain for security



Design Goals

One blockchain can encompass all worldwide ledger state. Data is committed to the root blockchain and it is only in the event of disputes of byzantine behavior that the fraud is proven and rolled back

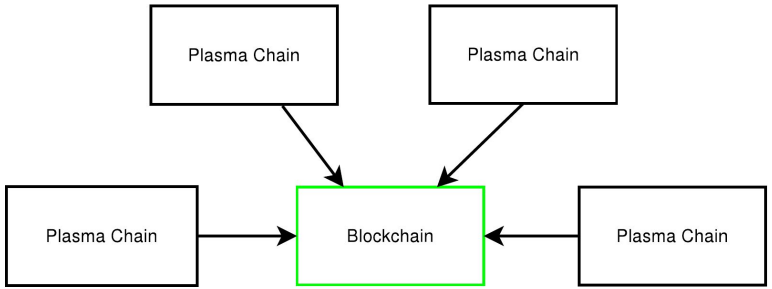
Trust minimization. The primary risks left in the design is around chain halting and blockspace availability, which is significantly mitigated with good parent chain selection

Payment and ledger scalability. Can hold an incredible amount of ledger entries, more robust against byzantine actors than multisig for holding a user's single output of a hundredth of a cent

Many Blockchains on a Blockchain

Initialize the Plasma blockchain. Send to an output defining the closure conditions with a special output

> E.g. **OP_PUSHDATA2** with a new Transaction version for Plasma opcodes. Encodes script in a pushdata similar to BIP 0016 or SegWit.
>
> **OP_PUSHDATA2[OP_CREATEPLASMABLOCK 5 20 <pubkey_validators_schnorr>]**



Localized ledger state. Only periodic commitments (Plasma blockchain blockhashes) submitted to the blockchain.

Consensus rules defined in fraud proofs. If any of the blocks are invalid, anyone can submit proof of fraudulent state transition to roll back blockchain

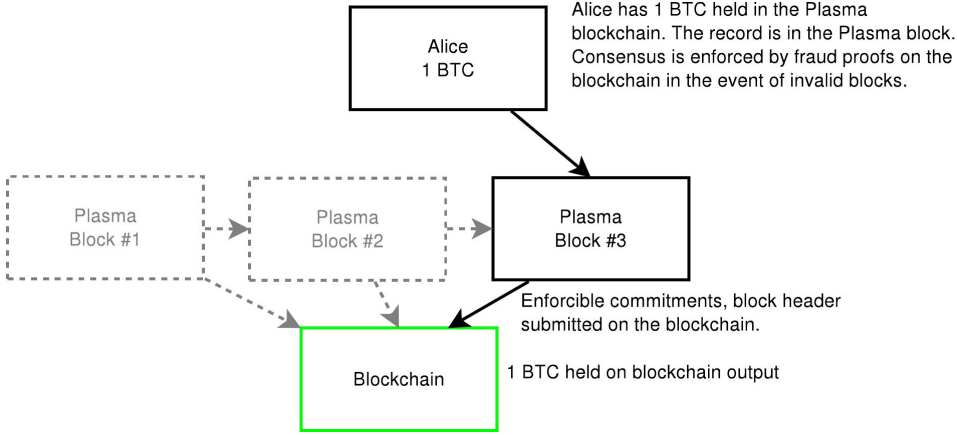
Periodic Commitments to the Root Chain

Only the Plasma blockhash/header committed to the root chain. This means that a minimum amount of data is submitted to the chain. This submission is a commitment to blockstate as well as creates ordering. **The output is updated/spent every time.**

> **Enforcement may require all outputs to be scripts, not P2SH due to data availability (potentially controversial!)**

Complex computation and value transfer. State transitions can occur in the Plasma blocks (e.g. between Plasma Block #2 and #3), but only tiny commitments are included to the root chain

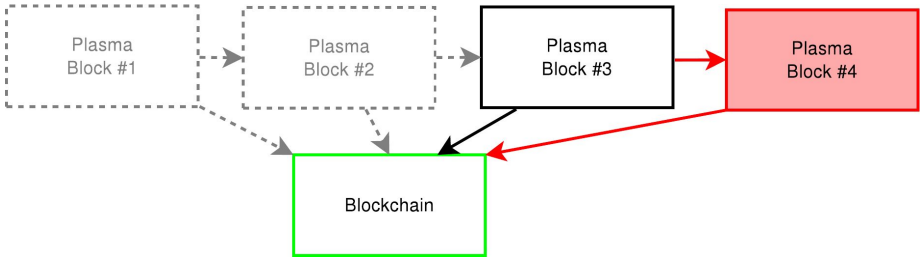
Scale. The root chain doesn't bother evaluating state unless someone disputes the data. However, data availability is needed to prove fraud! **This has been the fundamental problem of non-global computation!**



Fraudulent State Transitions

Invalid state transitions is the complexity around child blockchains. Moving funds in and out of the Plasma chain is only viable if one can be assured of state transitions.

Proving fraud only in entry and exit is insufficient. If you can reliably prove funds committed to the Plasma chain is valid by doing full-node validation of Bitcoin, as well as exits by providing proofs of the child chain to the root chain, you still haven't proven validity of state transitions, as the fraud in the child chain can occur between entry and exit

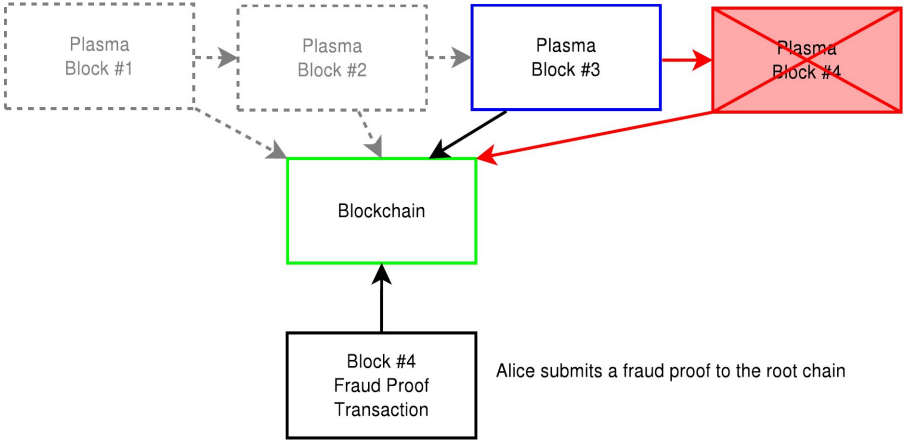


Fraud Proof Enforcement

The Plasma output on the root blockchain contains mechanisms to enforce using fraud proofs. If a fraudulent Plasma block is created and propagated, anyone who receives it can verify it is fraudulent as the rules are part of the chain consensus.

Requires a LOT of new opcodes to prove fraud

Example. Let's say a block is created and is invalid. Alice notices her ledger entry balance was removed in block #4, but she didn't spend her funds in the Plasma chain. She (or anyone else) submits a proof that it was removed (proof of exclusion) by including the merkleized commitment of the ledger and transactions. Block #4 gets rolled back and the signers of that block get penalized

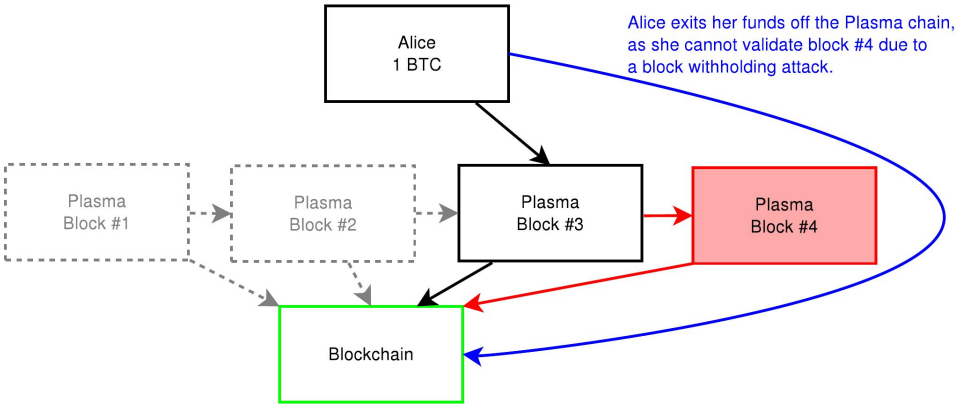


The Core Novelty in Plasma is around Exits

Exiting Byzantine Behavior. Pre-design the consensus rules so that all Plasma blockchains allows for orderly exits

Data unavailability. When blocks are withheld, we need to be able to exit. The solution for data unavailability is not to make it available (may be impossible), but rather to create mitigations in the event that occurs. **This is the core insight/novelty around Plasma**

Exit upon unavailability. When you discover blocks being withheld, you exit to another Plasma blockchain or the root blockchain as soon as possible



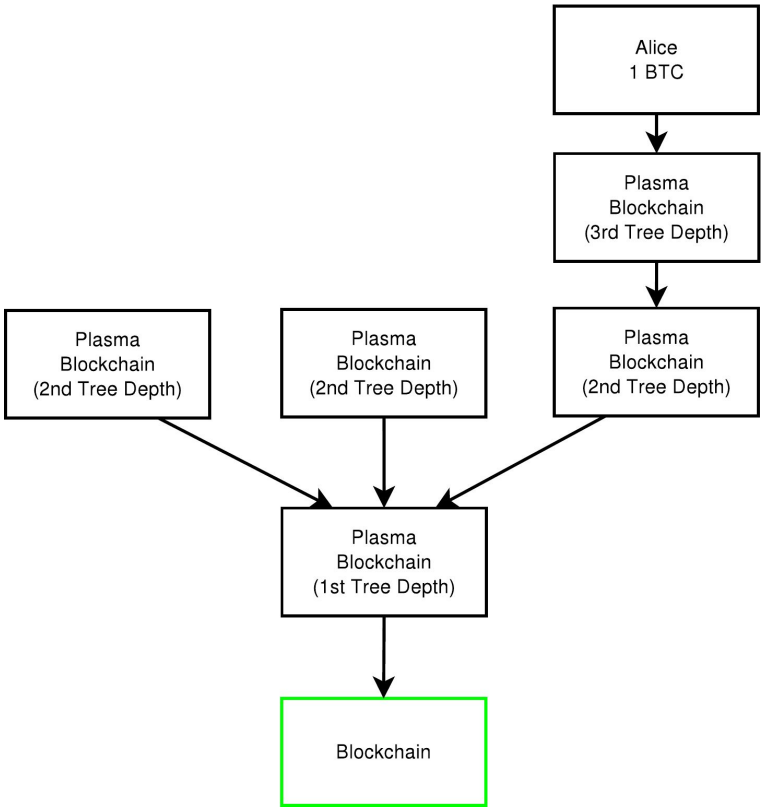
Blockspace Availability Using Nested Trees

Tree of blockchains. If we are able to create scaling by creating a blockchain within a blockchain, we can go deeper

Blockspace availability and fault recovery. If the validator set is different between parents of chains, there is greater chance of block availability to cheaply recover from faults

Many blockchains, one commitment. The block commitments flows up, with only one commitment to the root chain in the best case

Merkleized proofs of chains. The scriptSig contains proof of the merkle branch of the chain being referenced



Data Availability of Script Outputs

Problem 1: Data Availability is necessary for spending from Plasma block commitments

Every commitment spends into a new output. The output must have sufficient data to generate merkleized proofs. Therefore, the output script must be provided.

Problem 2: One needs Data Availability of old Plasma blocks!

This means that since the output updates every time, one must commit to the past *n* blocks in every output.

However, this creates huge memory requirements.

TXO Commitments to the rescue! Build a TXO commitment of old blocks

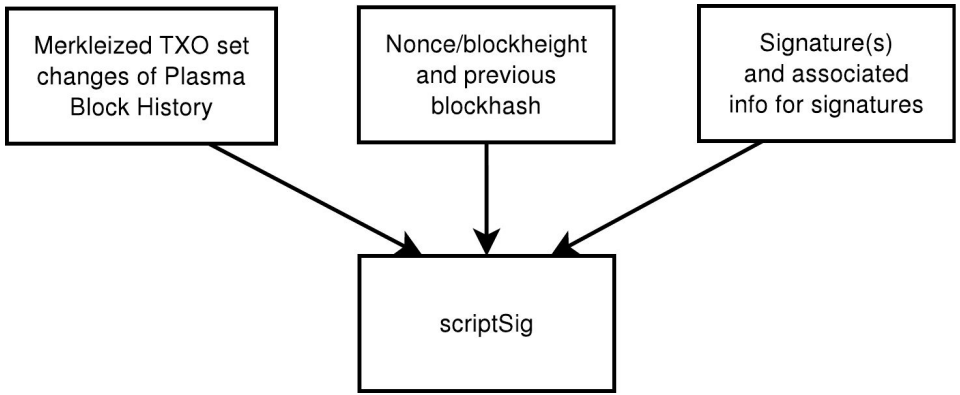
Example TXO Commitment of old blocks

Include as part of the output the data necessary for the new block. Example scriptSig for update:

OP_PUSHDATA2[OP_NEWPLASMABLOCK <merkle tree data required to add new state> <schnorr_sig and associated data to prove schnorr sig> <prev_blockhash>]

In order to update, the output merkle root must match the computed branch changes in the scriptSig spending the previous output

Note: this presumes the sig is malleable



Proof of Stake on Proof of Work

Proof-of-Authority is possible. A single party creates the blocks and signs off on it. If it's invalid, that party gets penalized. Hardcoded.

Can also do multiparty agreement such as PoS. A set of participants must agree to finalize Plasma blocks via Schnorr signature

Second Layer Proof of Stake. But how to do PoS participant selection? Presumes the participant set remains the same. Changing affects incentive and Plasma currently assumes token issuance for cryptoeconomic security

Create a true reserve currency. History rhymes, not repeats. Reserve currency in a multi-blockchain world. Could move Bitcoin based on activity on other blockchains if the other chain knows how to represent itself as a Plasma block

Future Work

Mechanism Improvements. Creating more incentives for correct exits

Finality designs. Presumes root chain reaches finality similar to LN, has a hard dependency, more research on chain finality incentives

Minimize block withholding attack incentives. Recursive ZK-SNARKs/STARKs could possibly provide mitigations around withheld fraudulent state transitions, minimizing the need for exit timing mechanisms. Something something accumulators (Suggested by Laolu and Vitalik, but idklol)

Minimizing load on a UTXO model in the root chain. It's possible construct it much simpler if we can hold more complex state, but has tradeoffs

Introducing UTXO computation. Output requires SNARKs/STARKs proof to spend

Thank you!

https://plasma.io
Joseph Poon <joseph@lightning.network>