A very special thanks to @JustinDrake for all the yummy availability help!

Overview

Plasma Cash, as defined in the Simple Spec, relies on a single Plasma operator. This operator has two strategies which can cause some harm to users of the Plasma Cash chain. The strategies are as follows:

i)

The Plasma operator can withhold a single block, forcing all users to exit the chain before they can spend their coins.

ii)

The Plasma operator can censor and order transactions.

Because these two strategies exist, users must place significant trust in the central Plasma operator. With these constraints, Plasma Cash would be most comparable to a much safer centralized exchange. We might see multiple Plasma Cash instances, each one being run by a single operator who looks similar to the current centralized exchange operators we have today. This is great for removing any chance of coin theft, but each Plasma Cash chain would end up competing.

An alternative vision for Plasma Cash is, through the use of a PoS sharding solution, constrain the Plasma operators in order to minimize trust. To do this we extend the Plasma Cash Simple Spec with the following alterations:

a)

Decentralize block finalization with proof of stake consensus, to mitigate operator strategy (i).

b)

Rotate block proposers, to mitigate operator strategy (ii).

With these changes, Plasma Cash operators no longer have privileged status. Multiple exchange-like entities can use the same Plasma Cash chain as block proposers. These proposers are constantly cycled in and out. Without privileged status, users and service providers can pool their liquidity into a single large Plasma Cash chain.

Sharding Validation

Plasma Cash is special because it allows for extremely large blocks. This is what provides its scalability. However, this is problematic for PoS validation because naively all validators would have to download and validate each block. With huge bandwidth and verification requirements, validators would be restricted to well connected data centers–a large centralization vector.

How do we solve this? Well we shard validation! How do we shard validation? There are a number of reasonable schemes. One possibility is a DFinity-style scheme using BLS signatures & an honest majority assumption. This is the scheme assumed to be used in this spec, but further exploration regarding sharded availability schemes is very

valuable. Please research!

Protocol Overview

Bonded Participants

1. A large set of validators, each with a bond of X

tokens at stake in the Plasma contract on the mainchain.

1. A set of block proposers with a large bond of tokens on the mainchain.

Step by step

1. A Plasma block proposer is selected. This can be done with limited predictability or even privately.
2. The block proposer collects huge numbers of transactions and creates a huge block (merkle tree). I don't want to make promises, but I'd say the number of transactions can be pretty massive
3. Proof of Stake validators are pooled into committees, which are randomly sampled to validate subsections of the block's merkle tree.

4. Block proposer distributes subsections of the merkle tree to relevant validators.
5. Validators check availability & validity of the transactions, and sign a message approving the subsection of the merkle tree.
6. If a majority of validators in each committee sign off on availability & validity, the Plasma Cash block is included in the main chain. We use BLS signatures so the on-chain overhead is simply checking a single signature. It is important to note this signature verification does cost ~200,000 gas

which is rather pricy.

1. Validators propagate merkle branches of coins to relevant users.
2. Repeat step (1).

The following is a diagram which further describes this process:

[

1182×1482 253 KB

](https://ethresear.ch/uploads/default/original/2X/9/9a9c67ceeb21364c5314c78bef410d82961a61fd.jpg)

Closing Thoughts

Data Availability

Fundamentally the most critical component to this mechanism is its guarantees around data availability. If all data is available, a Plasma Cash chain can live forever. However, data availability is one of the most difficult problems in the blockchain scalability space.

A DFinity-style scheme guarantees data availability with an honest majority assumption (66%+). However, ideally we can weaken these assumptions in the long term with fraud proofs and erasure encoding.

Slashing

Slashing conditions can be added to the Plasma block proposers. Some example slashing conditions can be:

1. Slash if an invalid state transition is included in a block.
2. Slash if two conflicting blocks are signed for the same block height.

The first rule protects against invalid state transitions, and the second rule provides stronger guarantees around a kind of “soft finality” when a block is included in the mainchain Plasma contract. The only way for a block which is signed by a proposer to not be included would be if the Plasma operator slashed themselves. The amount of coins slashed can be tuned but potentially could go very high.

Censorship

In a single block proposer approach, transaction censorship is a big problem. The operator could blacklist coins and just never include transactions which reference these coins. With block proposer rotation, a single censorship-free block proposer can ensure censorship resistance.

Next Steps

- Better sharding schemes with weaker honesty assumptions.
- Concise exit challenge-response scheme even on coins which have had invalid transactions included in the Plasma chain.
- State channels for instant finality!

This could be useful for trading with high throughput. Please please please tell me how to do this!

One Love