I noticed a lot of work around eg. RSA accumulators in an effort to reduce the sizes of Plasma Cash histories.

If the Plasma design required the operator to submit a validity proof[0] along with each block posted, I believe the owner of any coin would only need to keep track of the inclusion of the single transaction that sent the coin to them, since it's inclusion and the validity proof imply valid ownership. This would mean that you only need to store data proportional to the number of coins you have[1].

To attempt an exit, you could use the proof of inclusion of a txn that sent the token to you. For someone to challenge an invalid exit, they'd only need to prove that a spend was included later on, whether it's the direct next spend or a spend way later in the coin's history, thanks to the validity proof.

Most of the work I could find in the zk-sidechain direction was towards roll-up/roll-back. I think the advantage to this kind of system comparatively is that once your txn is included in the block and available, there's no chance of it rolling back, regardless of the availability of the rest of the data[2].

There is the tradeoff compared to roll-back of invalid exits being attempted still, but the exit game should be relatively simple[3] (only one round needed for any exit game).

I wanted to see if anyone is working in this direction, and if there's already a name and some discussion around this flavor of Plasma, or if it stops being Plasma once we include the zk stuff. Or maybe there's a good reason that I'm not noticing for this kind of design not being discussed as much as roll-up/roll-back

.

Thanks for reading!

[0] Something like a snark or stark that the merkle root of the set of transactions are valid given a prestate and poststate

[1] We can probably do even better than this by using the trick where you treat the coins as a number line, and let people own/send intervals of coins at a time; now the amount of history you need to keep track of is proportional to the number of fragments you have.

[2] To ensure txns aren't valid until the availability of its inclusion proof, you could have the scheme only consider txns valid after the sender signs the inclusion proof. So the sender signing the inclusion proof and sending the whole payload to the receiver is the atomic action that makes the txn valid. The receiver could only exit/challenge an invalid exit with the signed inclusion proof; only the inclusion proof without the signature would not be sufficient.

[3] To initiate an exit without the operator's assistance, Alice would post a txn, its inclusion proof, and the signature of whoever sent it to her, to initiate the exit. To invalidate the exit, Bob could post a similar payload for any txn of that coin that occurs later than Alice's txn.