Plasma Classic

At Leverj, we hope to implement a simplified version of plasma that is specialized for decentralized exchanges. This is a review of the plasma classic spec, based on our understanding of the plasma paper from the perspective of a DEX implementation. This review does not cover plasma cash or plasma MVP.

Here we will not discuss optional, ancillary and speculative parts of the paper such as Lightning over plasma, zk-Snarks and staking since it would distract from the central idea. The only exception is the multi-level Plasma tree with map-reduce, which is relevant since it is used to illustrate a sample DEX.

Plasma: The Central Idea:

1. Create a plasma blockchain P

that's anchored to root chain R

1. A Deposit in R

creates UTXO in P

1. A Withdrawal from R

removes UTXO from P

1. Periodically store block headers of P

into R

to commit state changes

1. Block headers contain sufficient information to prove correctness of state transitions
2. Incorrect state transitions in P

are rolled back (up to a point) using fraud-proofs on R

1. Exits are prioritized by earlier P

blocks, to ensure fraudulent outputs fail

1. If information needed for fraud-proofs is withheld, mass exit from P

to R

Plasma Compact Spec

Deposit

:

1. Depositor sends asset to Plasma contract C
2. C

creates proof of deposit transaction TxD

in root chain

1. TxD

is committed in Plasma block Pb1

with compact proof (who does this)

1. If this is withheld, deposit is recoverable (how? race condition)
2. Depositor signs and submits TxD

to Plasma

1. TxD

is confirmed in plasma block Pb2

1. Ready to spend
2. If 5 is not complete:
3. Submit withdraw tx to root chain with “sizable” bond
4. Wait “extra long” time
5. If fraud proven, lose bond, else proceed to withdraw (race condition)

Spending:

1. Sign spend TxS

and broadcast on P

.

1. If block withheld after, Alice can withdraw spent coin on root chain
2. Wait TxS

to be confirmed in block Pb3

1. If TxS

/block withheld, Either party could withdraw (non-deterministic)

1. Bob signs “acknowledgement” and transmits TxAck
2. If block withheld after, Bob can withdraw (race condition)
3. TxAck

is included in Pb4

1. Challenges:

a. Verify no withdrawal is in progress for coins being spent

b. Verify coins not already spent

c. Verify valid block

Withdrawals:

1. Submit signed withdrawal TxW

to root chain (whole outputs only) with

a. output bit positions to withdraw

b. “sizable” bond

1. Wait for challenge period
2. Challenge is proof-of-output already spent
3. If successful challenge, bond is forfeited
4. Else: Wait for timeouts of lower block timeouts
5. Anyone can call a public method finalizeExit
6. finalizeExit processes prioritized by UTXO age.

Mass withdrawal:

1. Alice “coordinates” with others to conduct mass exit
2. Pat “coordinates” with destination plasma chain to send funds (the destination chain is sending funds?) and is committed to automatically “recognize” funds (??)
3. Pat “shows” pending destination ledger to participants

4. Pat “takes“ signatures from all participants and verifies correctness to availability
5. Drop anyone who’s data is invalid
6. Create Exit TX with “massive” bond.
7. “Get all users” to sign off on Exit TX
8. Drop users who did not sign
9. “Observe” if there are other exit tx (race condition)
10. Remove duplicates
11. Go to 9.
12. Broadcast to parent/root chain
13. “Publish” bitmap of state being exited
14. If there are duplicate withdrawals (other mass exits?) update bitmap and balance
15. If there is a Mass withdrawal dispute:
16. Broadcast challenge with bond
17. If withdrawal challenge is not disputed Mexit is cancelled
18. If challenge is disputed, challenge the dispute with “large” bond
19. Dispute challenger should produce a valid spend or dispute is cancelled
20. Else bond is forfeited
21. Wait “a few weeks” for finalization

Major Issues:

Race conditions:

There are race conditions where event causality is not enforced but a specific temporal ordering is a necessity for expected behavior. In deposit and spending, this occurs due to the fact that one party updates a state on the root chain while another updates it on the plasma chain. Since there is no sync mechanism described to enforce order, race conditions can cause loss of funds.

1. The depositor can withdraw their deposit from the root chain after the validator observes the deposit but just before the validator sends a deposit entry in the plasma chain. This could be done using timing analysis or side-channel attack.
2. The depositor could wait “a very long time” and transmit the deposit acknowledgement to the plasma chain just after submitting the withdrawal to the root chain.
3. When Bob receives a coin from Alice, Bob can withdraw on root chain after a block confirm in the Plasma chain just before he submits a signed acknowledgement to the plasma chain. If his acknowledgement is confirmed, he could immediately try to spend the coin on the Plasma chain.

Note:

These race conditions are fixed in plasma MVP by having the Plasma contract create the UTXO in a new block on deposit and removing recipient’s acknowledgement for spending.

Mass Withdrawal is not clearly specified.

We list some obvious issues but the key weakness is that it’s unrealistic to assume that people would rely on a unclear spec of a complex process with a significant portion of their wealth.

1. Alice and Pat seem to be coordinating this operation. It’s unclear why we need Alice since Pat seems to be doing most of the work.
2. Reliance on other actors to exist and be willing to take your exit signature during a critical time introduces an undesirable centralization element.
3. “Coordinates”, “Shows”, “takes”, “Observe” are undefined. Are these out-of-band?

4. How would this work if “all the world’s trading” is taking place on a plasma chain with a hundred million people? Is this going to be one gigantic transaction?
5. The parent (or target chain) may get clogged with a huge amount of exit transactions in a very short time clogging the parent chain.
6. Verifying an exit Tx could take an unusually long time and is likely to be beyond gas limits.

Note:

There is no separate mass withdrawal in Plasma MVP. Everyone rushes to the exit when they see something is wrong.

Scaling concerns:

“scalable to billions of computations per second with minimal on-chain updates”

There is no information on how the above was calculated. Deposits and withdrawals have mandatory lock times and bond submissions making them a lot slower. Actual spending of outputs also has a block confirm time. The block time is likely to be of the order of milliseconds at the fastest. In addition, accessing the utxo (possibly from disk), assembling the data structure and signing them would take tens of milliseconds at the very least and waiting for the recipient to receive, countersign and rebroadcast the transaction would encounter network round trips that is likely to be tens of milliseconds even when geographically close. Realistically, block times are probably in seconds at the very least.

Overall, 100tx/sec per node is optimistic based on Teechan benchmarks where the burdens of sending a transaction are a lot less. Add amortized dispute and resolution times and it could be a lot worse. Even so, such a plasma blockchain would require 10 million nodes to reach 1 billion tx/seconds.

“parties are responsible for monitoring the particular chain they are interested in periodically to penalize fraud, as well as personally exiting the chain rapidly in the event of block withholding attacks.”

Monitoring “billions of transactions/sec” is simply not practical. Even at hundreds of transactions/sec, there wouldn’t be sufficient CPU/bandwidth to do this on a global scale.

Note:

This is addressed somewhat in Plasma Cash by requiring to only monitor individual coins on the root chain.

“Map-Reduce”

The ability to farm computations to a global size cluster is an exciting possibility. Map-Reduce is suitable for problems where the input is trivially partitioned into independent sets and partial solutions are easily combined. There is an unaddressed issue in the Plasma construction of how a huge data set is sent from root chain to child chains. Is the data set loaded into the root chain’s state via multiple transactions? This could be prohibitively expensive and could take an extremely long time. Also, how would the results be retrieved without severe bandwidth pressure? Perhaps it’s possible to do move the communication out-of-band using protocols yet to be developed.

In the case of monetary transactions, map-reduce is an unsuitable candidate since monetary transactions require historical and contextual information. The faulty decentralized example in the paper illustrates the issue clearly:

“As part of the commitment to the parent, all orders are coalesced into a merkleized commitment of a single order book represented as a single order by the chain itself. This step recursively reduces all children’s order books into a single orderbook represented by that chain, until it reaches the highest Plasma chain parent. ”

Its simply not possible to merge all orders of an orderbook into a single order. The above neglects the basic operations of trading, namely: placing and cancelling orders and price-time priority execution. Multiple chains for multiple pairs is also flawed and neglects that some tokens (say ETH) will be on multiple orderbooks and apportioning them in advance to the proper chain and withdrawing to a different chain with timeouts when you choose to trade something else is impractical.

“It is theoretically possible to conduct all of the world’s trading activity in this framework”

Not until we figure out how to do stop orders, liquidations, margining, short selling and much much more.

Conclusion

Plasma is an exciting development that is on the brink of breaking the scalability barrier for blockchains. We hope that the blockchain community will iron out the remaining issues and enable rapid large scale decentralized financial systems.