I saw an example of Plasma chains with multiple layers in Plasma white paper.

Therefore, I propose an implementation method, problems, and solutions for creating a Plasma Chain that is deeper than 2.

Purpose of hierarchical Plasma

Further reductions in fees and data compression can be considered. However, I could not come up with a specific usage.

Plasma MVP implementation ideas

[

Hierarchiecal%20plasma%20overview

2160×1239 33.3 KB

](https://ethresear.ch/uploads/default/original/2X/5/5ac8780a4d5c7feaa205015a49843a61999971f7.png)

As a Plasma MVP Plasma contract, the contract must work with Plasma MVP. Express with special UTXO. This is expressed as the UTXO Contract.

- Create a new ContractFlag field and if it is 0x01, consider it as Plasma address
- Create a new State field and put root and timestamp there
- Submitted Block data will be included in the State of the transaction to send money to yourself
- When UTXO whose ContractFlag is 0x1 is set as Input, it means that it has exited.
- When exiting, create a transaction where ContractFlag is 0x01, Input1, ContractFlag is 0x02, Output1 (newowner1) is its own address, and Output2 (newowner2) is the exit address.

As a result, it is possible to determine whether a transaction is being used in the upper chain and whether a valid transaction is not.

The reason for not using general-purpose virtual machines such as EVM and Bitcoin script this time is for simplicity. You can probably do the same with them. This time, the behavior is determined by the client's built-in implementation.

Incorrect Merkle tree change and user challenge

UTXO exiting, as usual, will give a proof of spend.

1. "The height of the block that contains the transaction you are trying to exit"
2. "Proof that there is a transaction that uses UTXO as input to exit"
3. "Proof that 2 is included in the block after 1"
4. "Proof that the transaction is correct (signature, etc.)"
5. Submit the above 4 items.
6. The submission method is to broadcast a transaction with ContractFlag of 0x01 as Input1 and Output, ContractFlag as 0x03, State including more than 4 in State

The operator re-creates a transaction with ContractFlag of 0x01, using a transaction with ContractFlag of 0x02 as input. Transactions with ContractFlag of 0x02 can be input only for transactions with ContractFlag of 0x01.

Rules that Ethereum contract and UTXO contract must follow

Ethereum contract Indicates the rules to be followed in UTXO contract.

- Transactions with ContractFlag of 0x02 cannot be exited
- Transactions with ContractFlag of 0x01 cannot be exited

- Execute skip exit
- Implementation of exportBlocks for rebase exit

When competing in the lower chain, that is, when a challenge against a fraud exit is met, the upper chain must reject the exit. It is possible to appeal more and more. Therefore, the chain needs to be able to check all the chains below it.

Two levels of things to note about remittance with Plasma

Users must monitor all chains above the chain that contains the transaction that they sent to ensure that their remittance has Finality to ensure consistency.

Possible attack methods

1. Attack to exit operator-spent transaction (conventional)
2. In case of harassment that the operator does not generate a block in the chain below or harassment not including a transaction in the block

As the first solution, you can still submit and challenge the Merkle proof to the higher chain.

Explain the second problem.

If all the operators below are bad guys, there will be users who leave money. The most important point is that the smart contract that actually pools tokens and ETH at the top runs on EVM, there is no operator, and it always behaves as EVM bytecode. The safety of the top chain is secured by PoW.

There are three exit methods for Exit. Of these, skip exit and rebase exit will be effective solutions to this attack.

Normal exit

Exit as usual.

Skip exit

Submit all the information of all subordinate chains to the top Ethrerum contract as a Merkle proof (UTXO and blockchain are both large hash trees)

All Ethereum contracts should be verified and the amount of ETH that the user has from the Deposit eth pool can be forced to exit. This is called a skip exit.

Rebase exit

The second and third plasma chains create an Ethereum Contract directly for exit and transfer the total amount of ETH that the chain has from the pool to the new contract.

It also invalidates the UTXO contract for the second Plasma in the first Plasma chain.

Export the block data there.

In this case, you can exit without waiting for the chain above you.

In other words, you can safely exit in exchange for a fee and slowdown.

Implementation

Now in progress.

GitHub

onokatio/plasma-on-plasma

Hierarchical plasma implementation. based on omisego's Plasma MVP - onokatio/plasma-on-plasma