This is our solution to validators earning fees for all transaction included in a block, regardless of whether the corresponding confirm signatures was ever broadcasted.

Post in our research repo

tldr:

fees are attached to broadcasting spend messages on the plasma network. If a user chooses to not broadcast their confirm signature for an already included transaction, they must still commit to the fees payed when exiting that utxo. Validator can challenge with a merkle proof otherwise.

Problem

With just a fee attribute in the transaction bytes, the validator is not guaranteed the fee when including the transaction in a block. This is due to the role of confirm signatures. Users can successfully exit the funds of an old utxo as long as the confirm signatures were never broadcasted.

Solution

Validators collect fees corresponding to the inputs

of all utxos in his/her block. As specified in the txBytes, the fees must be deducted from the inputs. A valid spend message on the child chain follows this equality:

Amount1 + Amount2 - Fee = Output1 + Output2

The fees are deducted from the first input and determined by the participants and not validators. This allows validators to choose which spend messages to include from the mempool. However due to the high volume, the market will most likely settle at an average low transaction fee

While processing each tx, the validator aggregates all input fees, and creates a new utxo with no history owned by them as the very last utxo in the block. By placing this new utxo last, all watchers of this child chain can verify the validity of the denomination of the new utxo by aggregating the fees for each tx themselves. Malicious activity of this last tx is grounds for a mass exit of the child chain. Placing the new utxo last also ensures that the validator has the lowest priority to exit with that block as the current head.

There are two different cases in which we assure that the validator is guarunteed their fee utxo for a given block.

1. For every transaction included in a block, any spend of the new utxos allows the validator to challenge an invalid exit that would steal the validator fees

2. For transactions included in a block but not spent, perhaps the confirmsig was never broadcasted, the previous owner retains the right to withdraw their previous utxo. In this scenario, we uphold the right for the previous owner to exit but they must commit to any fee payed by broadcasting a spend message by including the fee amount in startExit. The validator can challenge an invalid exit that does not commit to fees payed from the first input by submitting a merkle proof that the user had payed a different fee amount than described in the startExit. In a successful challenge, the validator keeps their earned fees and also the startExit bond.

Note: The validator has the ability to steal all current fees in the mempool by submitting invalid blocks. However, this is not an issue because there is zero incentive to steal fees. By mining honest blocks, the validators will eventually claim all fees in the mempool. Additionally, by attaching fees to spend messages in the network, this incentivies users to only broadcast transaction in which they will eventually send confirm signatures.

BONUS

: Because the brand new utxo is owned by the proposer of the current block, this solutions scales to any consensus mechanism with more than 1 validator. All without changing much of the root contract code at all!