tldr

: Based on a comment by @MaxC here (Plasma Cash: Plasma with much less per-user data checking) we can design down a version of plasma cash with a simpler exit procedure.

Motivation

The new protocol reduces the worst-case number of merkle branches checked on-chain from 4 to 3. In addition, the safety proof is easier to write in a way that generalizes to different designs, eg with splitting and merging.

Specification

To recap, tokens are indivisble and cannot be merged, transactions represent change in coin ownership and are stored in a sparse merkle tree whose root is comitted to the plasma contract. A mapping coinid => denomination

is maintained in storage. Deposits create a new block with a single transaction and update the mappings. So far, this is all identical with the original plasma cash design.

We include the additional feature that transactions must commit to a specific block number in which to be included. The client must ensure that before signing a transaction whose block commitment is n

, he must have validated all blocks before n

(i.e., with block number < n

).

The new exit procedure is:

1. Anyone can exit their coin by providing the last transaction in their coin's history, say C, which consists of a coinid and a block, with the block being later than the coin's deposit
2. An exit can be challenged in two ways: i) by providing a proof of a transaction spending C, or ii) by providing a transaction C* in the coin's history before C
3. A type (i) challenge cancels the exit immediately. a type (ii) challenge can be cancelled by providing the direct child of C*.

Coin Fork Choice Rule

Define the coin fork choice rule (CFCR) as a function from the current state of all committed blocks to the coin that can be withdrawn. We immediately see that the new CFCR is different than the original one. For instance, given this set of transactions,

[

img_20180508_145538

3756×2112 1.36 MB

](https://ethresear.ch/uploads/default/original/2X/d/d296ef0f41045906392cd8502f07c77104de0410.jpg)

The new CFCR specifies that block 3 (and only block 3) can be withdrawn, whereas the original plasma cash CFCR chooses block 4. Even more interestingly, given this set of transactions,

[

img_20180508_145604

3756×2112 1.54 MB

](https://ethresear.ch/uploads/default/original/2X/f/f96f07d0f91812854b3e3fc2fa860a516fffdaba.jpg)

The new CFCR chooses block 3, whereas the original plasma cash CFCR returns the empty set (i.e., every exit can be successfully challenged!)

An important example to check is this one.

[

img_20180508_145650

3756×2112 1.49 MB

](https://ethresear.ch/uploads/default/original/2X/e/eac49d4169f4d3bbb0aba374f706364977d0c62c.jpg)

Both the old and new CFCR chooses block 2. Any CFCR that does not return 2 is unsafe.

Safety Proofs

Even though we defined the CFCR in terms of the exit function, it is actually easier to define the CFCR on its own using just graph-theoretic concepts. For example, the new CFCR can be specified as:

Leftmost Unspent CFCR

: Return the leftmost vertex v

such that there does not exist an edge vw

from v

to another node.

Then, a safety proof involves proving four things:

1. (exiteable): if a coin returned by the CFCR is exited, the exit cannot be cancelled
2. (no other exits): if a coin not returned by the CFCR is exited, the exit can be successfully challenged and cancelled
3. (existence of validity proof): if a CFCR returns c on a given transaction graph and the owner of c does not spend c, then the CFCR still returns c on any later transaction graph
4. (atomic transferability of ownership and validity proof): when spending a coin, if the sender and receiver both follow the proper client procedures (eg validate all blocks < n

before signing a transaction spending n

) then the transfer is atomic in the sense that after the transfer, one of the sender and receiver can exit the coin. in addition, this can be done in a small number of attempted exits (in both original and new plasma cash, this number is 2).

The nice thing is that this safety proof format works for proving the safety of this design, of the original plasma cash, and of designs with splitting. In addition, you can “derive” the exit procedure from the CFCR, which is nice, since in my preliminary work on writing down and exit procedure and safety proof for various designs with both splitting and merging, it appears that they become quite complicated. More details in an upcoming post.

Karl’s CFCR

In https://karl.tech/plasma-cash-simple-spec/ and in a plasma call, @karl asked if it were possible to design a scheme where users can spend their coin even when block withholding is going on. This involves weakening the client procedure. We can come up with a CFCR that allows clients to do this:

Greedy Leftmost Spend CFCR

: Starting from the deposit transaction, recursively find the leftmost spend of a coin. In more detail: let the transaction graph T

be a directed graph over a prefix of the natural numbers, with 1 representing the block containing the deposit transaction. Define d: V(T) \to 2^{V(T)}

as d(n) = {m \in V(T) : nm \in E(T) }

(that is, the neighbours of n

downstream from it). Define f: V(T) \to V(T)

by f(n) = n

if d(n) = {}

and f(\min(d(n)))

otherwise. Then the CFCR is f(1)

.

This satisifies both (3) and (4). Hence we have reduced the problem to a “purely graph theoretic” one of: design an efficient exit procedure that exits a coin if and only if it is the output of this CFCR. So far, I have tried to come up with an exit

procedure that does this in a constant number of moves, but have been unable to do so.