[EDIT: @gakonst pointed out that Lucidity proposed and prototyped the "invalid history challenges are just prior exits" idea several months ago!

As for the other piece of the proposal (the "cannot cross pending exit" rule), after discussion with @gakonst, @benj0702, @karl, and others, we found a rearrangement attack that it does not prevent. Fortunately, we think we can use a version of limbo exits, as has been suggested by @ldct, to not only protect against rearrangement attacks, but also safely reduce the entire challenge period back from three days to one day.

A revised design will be forthcoming!]

[Thanks to Will Robinson, @benj0702, and @nginnever for helping me think through this.]

The standard exit game for Plasma Cash involves three types of challenges. Type (i) challenges are simple, but type (ii) challenges require one round of interaction, and type (iii) challenges are pretty ugly (since they require otherwise unnecessary details about the transaction's parent to be included in the exit attempt). There have been proposals for replacing the type (iii) challenges with other mechanisms, but those come with their own drawbacks.

I think there is a simpler exit game that only uses a single

challenge type, which is non-interactive.

1. Anyone can initiate an exit of a coin. There can be multiple pending exits for a coin at one time. Once a coin has had pending exits for 3 consecutive days, no new exit attempts for that coin are allowed. (So far, none of these rules are new—they're the same ones we think are already required for classic Plasma Cash to be secure.)

2. There is only one type of challenge: "cancel". You can instantly cancel any exit by showing a proof of a valid transaction that spends it. This is equivalent to the type (i) challenge in the original Plasma Cash game (sometimes called "challenge spent coin").

3. If your exit has been pending for 3 days, you can complete the exit, but only if there are no pending exits for that coin from an earlier block height

. (Pending exits from a later block height are OK! In fact, if there are pending later exits of a coin when you successfully exit, you get their exit bonds.)

1. A spent-coin challenge fails if it "crosses" a pending exit—that is, if there is a pending exit from a transaction at a block height between the height of the transaction being challenged and the spend being used to challenge it.

2. (Optional: optimistic exits) If there are no pending exits for a coin, then instead of showing the full Merkle proof when you initiate an exit, you can just declare the block height you are exiting from. Within one day, anyone can cancel your exit by providing a Merkle proof that your transaction is not included at that height. (Allowing exits after the first one to be optimistic appears to cause problems.)

Type (ii) challenges ("challenge invalid history") are replaced by rule 3. You simply initiate an exit attempt from the earlier transaction, which automatically blocks the later exit. Similarly, instead of "responses" to invalid-history challenges, you can cancel the earlier exit by showing a spend from it.

Type (iii) challenges ("challenge double spend") are replaced by rule 4. That challenge type was designed to handle the edge case in which the operator includes an invalid transaction in between two valid transactions. I believe this rule guarantees that the coin will still be exitable in that case. (This is explained below under "Edge cases").

Advantages

- Simpler conceptually. This is a game to find the earliest transaction which hasn't been (validly) spent.

- Much simpler interface. The only things you can do with respect to exits are "initiate," "cancel," and "complete," and the interfaces to those functions are pretty simple. In contrast, in the classic Plasma Cash exit game, you can "initiate," "challengeSpentCoin," "challengeInvalidHistory," "challengeDoubleSpend," "respond," "complete," and "timeout."

- More efficient invalid history challenges. In the classic game, if the operator initiates many simultaneous invalid exit attempts, you have to individually challenge each one. In this new exit game, a single transaction simultaneously "challenges" all exits from later block heights, and

initiates an exit of your coin from the compromised chain. (The logic for evaluating whether a pending exit is the earliest, and whether a cancelling transaction crosses a pending exit, is a little more involved than the corresponding logic in the classic exit game. But I think it can be fairly efficiently implemented by storing pending exits for each coin in the right data structure).

- No need for exits to include details about their parent transactions, or for transactions to commit to a target block height.

Edge cases

Suppose Alice owns a coin at block 1 and submits a spend to Bob, but the operator puts an invalid state transition in block 2 and then includes Alice’s transaction in block 3 (without making any of the data available). Alice can use the following protocol to exit her coin (cooperatively with Bob):

a) Alice attempts to exit from block 1.

b) Within 1 day, operator cancels it by revealing the transaction in block 3. (Otherwise, Alice’s exit succeeds after 2 more days.)

c) Bob attempts to exit the transaction in block 3.

d) Within 3 days, operator attempts to exit the transaction in block 2. (Otherwise, Bob’s exit succeeds.)

e) Alice attempts to exit the transaction in block 1. Operator cannot do a spent-coin challenge using the transaction in block 3, because that challenge would cross the pending exit from block 2, violating rule 3. Operator cannot complete their exit from block 2, because there is a pending prior exit.

f) Alice waits 3 days and completes the exit from block 1.

Finally, let’s look at another edge case that has recently been pointed out by@ldct and @gakonst (and which explains the 1-day vs. 3-day timeouts). Suppose the operator includes Alice’s spend to Bob in block 2 (without revealing it to Alice or Bob), puts an invalid transaction in block 3, and immediately initiates an exit of the coin in block 3. Alice needs to initiate an exit from block 1 (which, she might take up to 1 day to do, under our security assumptions). The operator has 1 day to challenge that exit by revealing the spend (in block 2). Bob then has 1 more day to initiate an exit from block 2, which will block the exit from block 3.