Special thanks to Joseph Poon and David Knott for discussions that led to this specification.

The following aims to provide a specification for a “minimal viable plasma implementation”. It aims to provide the basic security properties of Plasma in a very simplified way, though it leans heavily on users being willing to immediately exit as soon as they detect any kind of malfeasance.

The Plasma Contract

The Plasma contract maintains the following data structures:

- The owner

(set at initialization time)

- A list of Plasma blocks

, for each block storing (i) the Merkle root, (ii) the time the Merkle root was submitted.

- A list of submitted exit transactions

, storing (i) the submitter address, and (ii) the UTXO position (Plasma block number, txindex, outindex). This must be stored in a data structure that allows transactions to be popped from the set in order of priority.

A Plasma block can be created in one of two ways. First, the operator of the Plasma chain can create blocks. Second, anyone can deposit any quantity of ETH into the chain, and when they do so the contract adds to the chain a block that contains exactly one transaction, creating a new UTXO with denomination equal to the amount that they deposit.

The contract has the following functions:

- submitBlock(bytes32 root)

: submits a block, which is basically just the Merkle root of the transactions in the block

- deposit()

: generates a block that contains only one transaction, generating a new UTXO into existence with denomination equal to the msg.value

deposited

- startExit(uint256 plasmaBlockNum, uint256 txindex, uint256 oindex, bytes tx, bytes proof, bytes confirmSig)

: starts an exit procedure for a given UTXO. Requires as input (i) the Plasma block number and tx index in which the UTXO was created, (ii) the output index, (iii) the transaction containing that UTXO, (iv) a Merkle proof of the transaction, and (v) a confirm signature from each of the previous owners of the now-spent outputs that were used to create the UTXO.

- challengeExit(uint256 exitId, uint256 plasmaBlockNum, uint256 txindex, uint256 oindex, bytes tx, bytes proof, bytes confirmSig)

: challenges an exit attempt in process, by providing a proof that the TXO was spent, the spend was included in a block, and the owner made a confirm signature.

startExit

must arrange exits into a priority queue structure, where priority is normally the tuple (blknum, txindex, oindex) (alternatively, blknum * 1000000000 + txindex * 10000 + oindex). However, if when calling exit, the block that the UTXO was created in is more than 7 days old, then the blknum of the oldest Plasma block that is less than 7 days old is used instead. There is a passive loop that finalizes exits that are more than 14 days old, always processing exits in order of priority (earlier to later).

This mechanism ensures that ordinarily, exits from earlier UTXOs are processed before exits from older UTXOs, and particularly, if an attacker makes a invalid block containing bad UTXOs, the holders of all earlier UTXOs will be able to exit before the attacker. The 7 day minimum ensures that even for very old UTXOs, there is ample time to challenge them.

The Plasma Chain

Each Merkle root should be a root of a tree with depth-16 leaves, where each leaf is a transaction. A transaction is an RLP-encoded object of the form:

[blknum1, txindex1, oindex1, sig1, # Input 1 blknum2, txindex2, oindex2, sig2, # Input 2 newowner1, denom1, # Output 1 newowner2, denom2, # Output 2 fee]

Each transaction has 2 inputs and 2 outputs, and the sum of the denominations of the outputs plus the fee must equal the

sum of the denominations of the inputs. The signatures must be signatures of all the other fields in the transaction, with the private key corresponding to the owner of that particular output. A deposit block has all input fields, and the fields for the second output, zeroed out. To make a transaction that spends only one UTXO, a user can zero out all fields for the second input.

User Behavior

The process for sending a Plasma coin to someone else is as follows:

1. Ask them for their address.
2. Send a transaction that sends some of your UTXOs to their address.
3. Wait for it to get confirmed in a block.
4. Send them a confirm message, signed with the keys that you use for each of your UTXO inputs.

Emergency exiting

A user should continually validate (or validate at least once per 7 days) that the Plasma chain is fully available and valid; if it is not, they should exit immediately.

Proof of correctness sketch

Approximate claim: a UTXO with denomination D will entitle its owner to withdraw D coins, and that (i) fraudulent cancellations and (ii) invalid UTXOs successfully withdrawing and draining the contract before the user can fully withdraw will not prevent them from doing so.

Suppose that:

- The first invalid or unavailable transaction is at position (blknum_i, txindex_i).
- There exist TXOs before that point of total denomination M, of which M-N is spent and N is unspent. We call a TXO spent if a transaction spending it has been included in a block, and a commit from the owner of the TXO is in the hands of the owner of at least one of the child TXOs.

Consider any UTXO with denomination D that was confirmed in a position before (blknum_i, txindex_i), call it (blknum_e, txindex_e). We assume that within 1 day of the first invalid or unavailable transaction getting confirmed, the owner of that UTXO publishes an exit. This exit is assigned a priority of (blknum_e, txindex_e), and so it will be processed before (blknum_i, txindex_i). We also assume that if there is a transaction “in flight” spending this UTXO, and this gets included in a future block, then the owner will refuse to sign the commit. We know that:

- By the validity assumption, there are >=N coins deposited in the contract.
- There are no UTXOs with commits spending that UTXO, so a challenge is not possible.
- All TXOs chronologically before (blknum_e, txindex_e) are valid. We ignore TXOs chronologically after (blknum_e, txindex_e) because they have no ability to influence the given UTXO’s ability to exit successfully (TXOs before it can, by draining the balance first)
- TXOs chronologically before (blknum_e, txindex_e) are of two types: (i) unspent, with total denomination N-D, (ii) spent, with total denomination M-N. Exits of the second type can be challenged, and exits of the first type will succeed.

Hence, there will be at least D coins left in the contract’s deposit to pay the owner of the deposit.

The following aims to provide a specification for a “minimal viable plasma implementation”. It aims to provide the basic security properties of Plasma in a very simplified way, though it leans heavily on users being willing to immediately exit as soon as they detect any kind of malfeasance.