Expanding on a post I made in the Plasma Cash thread, and after reading the discussion there.

Coin IDs are variable-length bitstrings, which are encoded into a fix-length bitstring by prepending the unique string that matches the regex 0*1; for example, assuming encoding to uint8, “0” is encoded as “00000010”, “1” is encoded as “00000011”, “110” is encoded as “00001110”. there is a global constant K and the denomination of a coin with coinid of length k is 2^(K-k).

There are three types of transactions:

1. a single coin (with coin id X) changes owner
2. two coins with coin ids X0 and X1 merge into a coin with id X
3. a coin with id X splits into coins with id X0 and X1

Block headers commit to a (binary, not-necessarily-complete) merkle tree of transactions. each transaction is labelled by X and transactions must be stored in the merkle tree at position X. we say that two coins “intersect” if one of their coin ids is a prefix of the other (intuition: we can view a coin as a set of the smallest coins into which it can be split (e.g.: maximum coin id length is 7, then consider “11111” as the union of coins {“1111100”, “1111101”, “1111110”, “1111111”}), and coin intersection reduces to set intersection, equivalently set containment). note that by the consturction of the transaction tree, a commitment that no transactions includes coinid P implies that no transaction include coinid PB for all bitstrings B.

The root plasma contract stores the subset of minimal coins that have been deposited, and no valid transaction can change this subset. if an invalid transaction with id X begins exit, every honest coinholder that holds a coin Y that intersects X knows and can stop the exit.

An assumption here is that users will find it mutually beneficial to swap coins of the same denomination with each other in such a way that they end up with coins that can be merged.