```
BIP: 30
Layer: Consensus (soft fork)
Title: Duplicate transactions
Author: Pieter Wuille <pieter.wuille@gmail.com>
Comments-Summary: No comments yet.
Comments-URI: https://github.com/bitcoin/bips/wiki/Comments:BIP-0030
Status: Final
Type: Standards Track
Created: 2012-02-22
License: BSD-2-Clause
```

Abstract

This document gives a specification for dealing with duplicate transactions in the block chain, in an attempt to solve certain problems the reference implementation has with them.

Copyright

Motivation

So far, the Bitcoin reference implementation always assumed duplicate transactions (transactions with the same identifier) didn't exist. This is not true; in particular coinbases are easy to duplicate, and by building on duplicate coinbases, duplicate normal transactions are possible as well. Recently, an attack that exploits the reference implementation's dealing with duplicate transactions was described and demonstrated. It allows reverting fully-confirmed transactions to a single confirmation, making them vulnerable to become unspendable entirely. Another attack is possible that allows forking the block chain for a subset of the network.

Specification

To counter this problem, the following network rule is introduced:

- Blocks are not allowed to contain a transaction whose identifier matches that of an earlier, not-fully-spent transaction in the same chain.

This rule initially applied to all blocks whose timestamp is after March 15, 2012, 00:00 UTC (testnet: February 20, 2012 00:00 UTC). It was later extended by Commit Apply BIP30 checks to all blocks except the two historic violations. to apply to all blocks except the two historic blocks at heights 91842 and 91880 on the main chain that had to be grandfathered in.

Rationale

Whatever solution is used, the following law must be obeyed to guarantee sane behaviour: the set of usable transactions outputs must not be modified by adding blocks to the chain and removing them again. This happens during a reorganisation, and the current Bitcoin reference implementation does not obey this law in case the temporarily added blocks contain a duplicate transaction.

There are several potential solutions to this problem:

1. Guarantee that all coinbases are unique, making duplicate transactions very hard to create.
2. Remember previous remaining outputs of a given transaction identifier, in case a new transaction with the same identifier is added.
3. Only allow duplicate transactions in case the previous instance of the transaction had no spendable outputs left. Removing a block from the chain can then safely reset the removed transaction's outputs to nothing.

The first option is probably the most complete one, as it also guarantees transaction identifiers are unique. However, implementing it requires several changes that need to be accepted throughout the network. Furthermore, it does not prevent duplicate transactions based on earlier duplicate coinbases. The second option is impossible to implement in a forward-compatible way, as it potentially renders currently-invalid blocks valid. In this document we choose for the third option, because it only requires a trivial change.

Fully-spent transactions are allowed to be duplicated in order not to hinder pruning at some point in the future. Not allowing any transaction to be duplicated would require evidence to be kept for each transaction ever made.

Backward compatibility

The addition of this rule only makes some previously-valid blocks invalid. This implies that if the rule is implemented by a supermajority of miners, it is not possible to fork the block chain in a permanent way between nodes with and without the new rule.

Implementation

A patch for the reference client can be found on https://github.com/sipa/bitcoin/tree/nooverwritetx

This BIP was implemented in Commit Do not allow overwriting unspent transactions (BIP 30) There have been additional commits to refine the implementation of this BIP.

Acknowledgements

Thanks to Russell O'Connor for finding and demonstrating this problem, and helping test the patch.