```
BIP: 115
Layer: Consensus (soft fork)
Title: Generic anti-replay protection using Script
Author: Luke Dashjr <luke+bip@dashjr.org>
Comments-Summary: No comments yet.
Comments-URI: https://github.com/bitcoin/bips/wiki/Comments:BIP-0115
Status: Rejected
Type: Standards Track
Created: 2016-09-23
License: BSD-2-Clause
```

Abstract

This BIP describes a new opcode (`OP_CHECKBLOCKATHEIGHT`) for the Bitcoin scripting system that allows construction of transactions which are valid only on specific blockchains.

Copyright

Specification

`OP_CHECKBLOCKATHEIGHT` redefines the existing `OP_NOP5` opcode.

When this opcode is executed:

- If the stack has fewer than 2 elements, the script fails.
- If the top item on the stack cannot be interpreted as a minimal-length 32-bit CScriptNum, the script fails.
- The top item on the stack is interpreted as a block height (ParamHeight).
- If the blockchain (in the context of the execution) does not have ParamHeight blocks prior to the one including this transaction, the script fails (this failure must not be cached across blocks; it is equivalent to non-final status).
- If ParamHeight specifies a block deeper than 52596 blocks in the chain (including negative values), the opcode completes successfully and script continues as normal.
- The second-to-top item on the stack is interpreted as a block hash (ParamBlockHash).
- If ParamBlockHash is longer than 28 bytes, the script fails.
- If ParamBlockHash does not match the equivalent ending bytes of the block hash specified by ParamHeight, the script fails.

Otherwise, script execution will continue as if a NOP had been executed.

Deployment

This BIP will be deployed by "version bits" BIP9 with the **name** "cbah" and using **bit** TBD.

For Bitcoin **mainnet**, the BIP9 **starttime** will be TBD (Epoch timestamp TBD) and BIP9 **timeout** will be TBD (Epoch timestamp TBD).

For Bitcoin **mainnet**, the BIP9 **starttime** will be TBD (Epoch timestamp TBD) and BIP9 **timeout** will be TBD (Epoch timestamp TBD).

Motivation

Securely recovering from double spends

In some circumstances, users may wish to spend received bitcoins before they have confirmed on the blockchain (Tx B1). However, if the transaction sending them those bitcoins (Tx A1) is double-spent, the wallet must re-issue their own transaction spending them (Tx B2). So long as the double-spend of the incoming transaction (Tx A2) also pays the wallet, this can be managed by simply updating the outgoing transaction with the new outpoint and resigning. However, if the double-spend does not pay the wallet, the situation is presently irrecoverable: it must spend different, non-conflicting TXOs in Tx B2, which allows an attacker to then reorganise the chain (reversing the incoming transaction's double-spend) and confirm both of his transactions Tx B1 and Tx B2.

By adding `OP_CHECKBLOCKATHEIGHT`, the wallet can issue Tx B2 with a condition that the block confirming Tx A2 is in the history, thus eliminating this risk.

Replay protection in the event of a persistent blockchain split

In the event of a persistent blockchain split, some mechanism is desired by which the UTXOs valid in either chain may be spent without the transaction being validly replayable on the other chain.

This can be guaranteed by choosing a block which exists only on either side of the split, and pinning (using `OP_CHECKBLOCKATHEIGHT`) common UTXOs to be spent only on chains based on that block.

Best practices for wallets

To avoid unnecessary conflicts when a chain is reorganized, wallets should always avoid specifying the last 100 blocks when practical. Wallets that use recent blocks when unavoidable SHOULD actively monitor the network and re-create transactions that are reorganised out with updated block hashes. Unless it conflicts with local/user security policies, wallets SHOULD retain the private key in memory to re-sign such transactions until the pinned block is at least 100 blocks deep into the chain.

For ordinary usage, wallets SHOULD specify the ParamBlockHash as 16 bytes.

Rationale

How is this different from the transaction's lock-time?

- The lock-time specifies a time or block height before a transaction becomes valid. `OP_CHECKBLOCKATHEIGHT`, on the other hand, specifies a specific block's hash.

Why are block heights required to be absolute, rather than relative?

- A relative block height would allow for creation of transactions which are valid at block N, but not N+1. This is carefully avoided by Bitcoin to ensure that if any given block is reorganised out, non-malicious transactions can be simply re-confirmed in a later block.

Why are blocks older than 52596 deep in the chain not verified?

- This is to avoid creating an infinite storage requirement from all full nodes which would be necessary to maintain all the block headers indefinitely. 52596 block headers requires a fixed size of approximately 4 MB.
- In any case where you might want to specify a deeper block, you can also just as well specify a more recent one that descends from it.
- It is assumed that 1 year is sufficient time to double-spend any common UTXOs on all blockchains of interest.
- If a deeper check is needed, it can be softforked in. Making the check more shallow, on the other hand, is a hardfork.

Why is ParamBlockHash allowed to match less than the full block hash?

- In a chain split, it is sufficient to check only a few bytes to avoid replay.
- In all scenarios, it is likely sufficient to check only a minority of the full hash to avoid any realistic chance of replay.
- Allowing less than the full hash to be specified saves space in transaction data.
- Using a single byte can be combined with other opcodes (such as `OP_LESSTHAN`) to enable on-chain gambling logic.

What if ParamBlockHash has leading zeros? Should this be prevented?

- If leading zeros are included, they should be compared to the actual block hash. (If they were truncated, fewer bytes would be compared.)
- It is unlikely that the leading zeros will ever be necessary for sufficient precision, so the additional space is not a concern.
- Since all block hashes are in principle shorter than than 29 bytes, ParamBlockHash may not be larger than 28 bytes.

Why is it safe to allow checking blocks as recently as the immediate previous block?

- This should only be used when necessary (ie, the deeper block is not sufficient), and when the wallet can actively issue updates should the

blockchain reorganise.
- While this allows intentionally creating a transaction which may be invalid in a reorganization, the same can already be accomplished by creating double spends.

Backwards Compatibility

`OP_NOP5` ought to be forbidden by policy by all miners for future extensions such as this, so old miners will under no circumstances produce blocks which would now be considered invalid under the new rules. However, miners must still upgrade to avoid accepting and building on top of such a possible invalid block as part of an attack.

Old nodes will likely also not relay transactions using this opcode for the same extensibility reasons, but this is not important since the rule cannot be verified deterministically outside the context of a block.

Reference Implementation

https://github.com/bitcoin/bitcoin/compare/master...luke-jr:cbah