```
BIP: 17
Layer: Consensus (soft fork)
Title: OP_CHECKHASHVERIFY (CHV)
Author: Luke Dashjr <luke+bip17@dashjr.org>
Comments-Summary: No comments yet.
Comments-URI: https://github.com/bitcoin/bips/wiki/Comments:BIP-0017
Status: Withdrawn
Type: Standards Track
Created: 2012-01-18
License: BSD-2-Clause
```

Abstract

This BIP describes a new opcode (OP_CHECKHASHVERIFY) for the Bitcoin scripting system, and a new 'standard' transaction type that uses it to enables the receiver of bitcoins to specify the transaction type needed to re-spend them.

Copyright

Motivation

The purpose of pay-to-script-hash is to move the responsibility for supplying the conditions to redeem a transaction from the sender of the funds to the redeemer.

The benefit is allowing a sender to fund any arbitrary transaction, no matter how complicated, using a fixed-length 20-byte hash that is short enough to scan from a QR code or easily copied and pasted.

Specification

OP_CHECKHASHVERIFY will re-define the existing OP_NOP2 opcode, and will function as follows when executed:

- First, hash the end of the prior script (in the general case, scriptSig; if no prior script, a null string is hashed) beginning from the last evaluated OP_CODESEPARATOR onward (or from the beginning of the script, if no OP_CODESEPARATOR was present)
- Then, compare this with the item on the top of the stack (if there is none, the script fails immediately)
- If the hashes match, do nothing, proceed as if an OP_NOP; if they do not match, the script fails immediately.
- Note that in the case of a matched hash, the top stack item (the hash being compared with) is not popped off the stack. This is for backward compatibility.

This opcode reassignment shall be applied when validating transactions in blocks only with timestamps after February 23, 2012 (see the Backwards Compatibility section for details).

A new standard transaction type that is relayed and included in mined blocks is defined:

```
[20-byte-hash-value] OP_CHECKHASHVERIFY OP_DROP
```

[20-byte-hash-value] shall be the push-20-bytes-onto-the-stack opcode (0x14) followed by exactly 20 bytes.

This new transaction type is redeemed by a standard scriptSig:

```
...signatures... OP_CODESEPARATOR {script}
```

Transactions that redeem these pay-to-script outpoints are only considered standard if they contain exactly one OP_CODESEPARATOR and the appended *script* is, itself, one of the other standard transaction types.

Example

For example, the scriptPubKey and corresponding scriptSig for a one-signature-required transaction is:

```
   scriptSig: [signature] OP_CODESEPARATOR [pubkey] OP_CHECKSIG
scriptPubKey: [20-byte-hash of {[pubkey] OP_CHECKSIG} ] OP_CHECKHASHVERIFY OP_DROP
```

2-of-3:

```
   scriptSig: [signatures...] OP_CODESEPARATOR 2 [pubkey1] [pubkey2] [pubkey3] 3 OP_CHECKMUL
scriptPubKey: [20-byte-hash of {2 [pubkey1] [pubkey2] [pubkey3] 3 OP_CHECKMULTISIG} ] OP_CHE
```

Rationale

This BIP replaces BIP 12 and BIP 16, which propose evaluating a Script from the stack after verifying its hash.

The Motivation for this BIP (and BIP 13, the pay-to-script-hash address type) is somewhat controversial; several people feel that it is unnecessary, and complex/multisignature transaction types should be supported by simply giving the sender the complete {serialized script}. The author believes that this BIP will minimize the changes needed to all of the supporting infrastructure that has already been created to send funds to a base58-encoded-20-byte bitcoin addresses, allowing merchants and exchanges and other software to start supporting multisignature transactions sooner.

There is a 1-confirmation attack on old implementations, but it is expensive and difficult in practice. The attack is:

1. Attacker creates a pay-to-script-hash transaction that is valid as seen by old software, but invalid for new implementation, and sends themselves some coins using it.
2. Attacker also creates a standard transaction that spends the pay-to-script transaction, and pays the victim who is running old software.
3. Attacker mines a block that contains both transactions.

If the victim accepts the 1-confirmation payment, then the attacker wins because both transactions will be invalidated when the rest of the network overwrites the attacker's invalid block.

The attack is expensive because it requires the attacker create a block that they know will be invalidated by the rest of the network. It is difficult because creating blocks is difficult and users should not accept 1-confirmation transactions for higher-value transactions.

Backwards Compatibility

These transactions are non-standard to old implementations, which will (typically) not relay them nor include them in blocks.

Old implementations will not validate that the {script}'s hash value matches when they validate blocks created by software that fully support this BIP.

Avoiding a block-chain split by malicious pay-to-script transactions requires careful handling of one case:

- A pay-to-script-hash transaction that is invalid for new clients/miners but valid for old clients/miners.

To gracefully upgrade and ensure no long-lasting block-chain split occurs, more than 50% of miners must support full validation of the new transaction type and must switch from the old validation rules to the new rules at the same time.

To judge whether or not more than 50% of hashing power supports this BIP, miners are asked to upgrade their software and put the string "p2sh/CHV" in the input of the coinbase transaction for blocks that they create.

On February 8, 2012, the block-chain will be examined to determine the number of blocks supporting pay-to-script-hash for the previous 7 days. If at least 60% contain "p2sh/CHV" in their coinbase, then all blocks with timestamps after 23 Feb 2012, 00:00:00 GMT shall have their pay-to-script-hash transactions validated.

If a majority of hashing power does not support the new validation rules, then rollout will be postponed (or rejected if it becomes clear that a majority will never be achieved).

OP_NOP2 is used, so existing OP_EVAL (BIP 12) transactions in the block chain can still be redeemed.

Reference Implementation

Validation, sending, and receiving for bitcoind git master

Validation only for 0.3.19+

See Also

- The Address format for Pay to Script Hash BIP
- M-of-N Multisignature Transactions (BIP 11)
- Example BIP 17 transaction chain: a b c d