BIP 0143 - Transaction Signature Verification for Version 0 Witness Program (soft fork)

return to the Bitcoin Improvement Proposals

Table 1: Data sheet for Transaction Signature Verification for Version 0 Witness Program

Title	Transaction Signature Verification for Version 0 Witness Program
Layer	Consensus (soft fork)
Author	Johnson Lau , Pieter Wuille
Comments-Summary	No comments yet.
Comments-URI	https://github.com/bitcoin/bips/wiki/Comments:BIP-0143
Status	Final
Type	Standards Track
Created	2016-01-03
Post History	
Description	https://github.com/bitcoin/bips/blob/master/bip-0143.mediawiki
License	PD

> **Note**: The following is an excerpt from the official Bitcoin site. It is provided here as a convenience and is not authoritative. Refer to the original document(s) as the authoritative reference.

Abstract

This proposal defines a new transaction digest algorithm for signature verification in version 0 witness program, in order to minimize redundant data hashing in verification, and to cover the input value by the signature.

Motivation

There are 4 ECDSA signature verification codes in the original Bitcoin script system: CHECKSIG, CHECKSIGVERIFY, CHECKMULTISIG, CHECKMULTISIGVERIFY ("sigops"). According to the sighash type (ALL, NONE, SINGLE, ANYONECANPAY), a transaction digest is generated with a double SHA256 of a serialized subset of the transaction, and the signature is verified against this digest with a given public key. The detailed procedure is described in a Bitcoin Wiki article. [1)]

Unfortunately, there are at least 2 weaknesses in the original SignatureHash transaction digest algorithm:

- *For the verification of each signature, the amount of data hashing is proportional to the size of the transaction. Therefore, data hashing grows in O(n2) as the number of sigops in a transaction increases. While a 1 MB block would normally take 2 seconds to verify with an average computer in 2015, a 1MB transaction with 5569 sigops may take 25 seconds to verify. This could be fixed by*

optimizing the digest algorithm by introducing some reusable "midstate", so the time complexity becomes O(n).[2)3)4)]

- *The algorithm does not involve the amount of Bitcoin being spent by the input. This is usually not a problem for online network nodes as they could request for the specified transaction to acquire the output value. For an offline transaction signing device ("cold wallet"), however, the unknowing of input amount makes it impossible to calculate the exact amount being spent and the transaction fee. To cope with this problem a cold wallet must also acquire the full transaction being spent, which could be a big obstacle in the implementation of lightweight, air-gapped wallet. By including the input value of part of the transaction digest, a cold wallet may safely sign a transaction by learning the value from an untrusted source. In the case that a wrong value is provided and signed, the signature would be invalid and no funding might be lost.*[5)]

Deploying the aforementioned fixes in the original script system is not a simple task. That would be either a hardfork, or a softfork for new sigops without the ability to remove or insert stack items. However, the introduction of segregated witness softfork offers an opportunity to define a different set of script semantics without disrupting the original system, as the unupgraded nodes would always consider such a transaction output is spendable by arbitrary signature or no signature at all.[6)]

1)
https://en.bitcoin.it/wiki/OP_CHECKSIG
2)
CVE-2013-2292 https://web.nvd.nist.gov/view/vuln/detail?vulnId=CVE-2013-2292
3)
New Bitcoin vulnerability: A transaction that takes at least 3 minutes to verify, https://bitcointalk.org/?topic=140078
4)
The Megatransaction: Why Does It Take 25 Seconds?, http://rusty.ozlabs.org/?p=522
5)
SIGHASH_WITHINPUTVALUE: Super-lightweight HW wallets and offline data, https://bitcointalk.org/index.php?topic=181734.0
6)
BIP141: Segregated Witness (Consensus layer), https://github.com/bitcoin/bips/blob/master/bip-0141.mediawiki