BIP 9 - LINK

Version Bits with Timeout and Delay

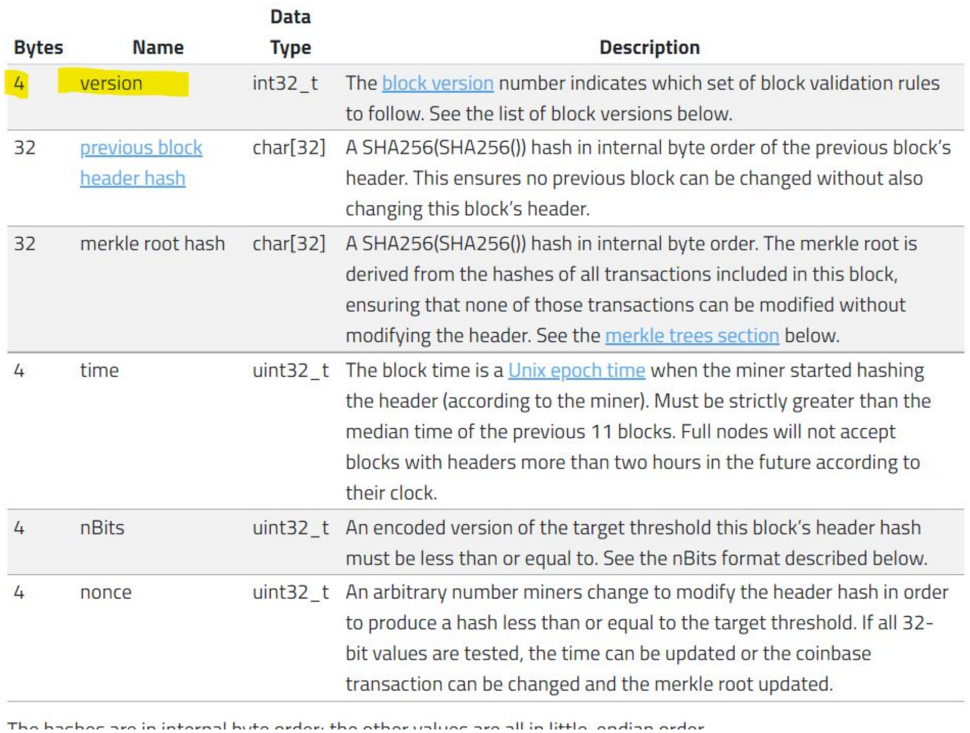
DISCLAIMER - ⚠️Potentially Contentious ⚠️

- BIP 9 Places the power of a soft fork activation in with the miners (nodes with hashpower)
- BIP 8 drafts a replacement for BIP 9 - NOT ‘FINAL’
- Speedy Trial Activation
- Pure and simple it involves consensus regarding the implementation of 1 or more softfork(s)

VS.

&

Version Bits in a Bitcoin Block - Where are they?



Bytes	Name	Data Type	Description
4	version	int32_t	The block version number indicates which set of block validation rules to follow. See the list of block versions below.
32	previous block header hash	char[32]	A SHA256(SHA256()) hash in internal byte order of the previous block's header. This ensures no previous block can be changed without also changing this block's header.
32	merkle root hash	char[32]	A SHA256(SHA256()) hash in internal byte order. The merkle root is derived from the hashes of all transactions included in this block, ensuring that none of those transactions can be modified without modifying the header. See the merkle trees section below.
4	time	uint32_t	The block time is a Unix epoch time when the miner started hashing the header (according to the miner). Must be strictly greater than the median time of the previous 11 blocks. Full nodes will not accept blocks with headers more than two hours in the future according to their clock.
4	nBits	uint32_t	An encoded version of the target threshold this block's header hash must be less than or equal to. See the nBits format described below.
4	nonce	uint32_t	An arbitrary number miners change to modify the header hash in order to produce a hash less than or equal to the target threshold. If all 32-bit values are tested, the time can be updated or the coinbase transaction can be changed and the merkle root updated.

The hashes are in internal byte order; the other values are all in little-endian order

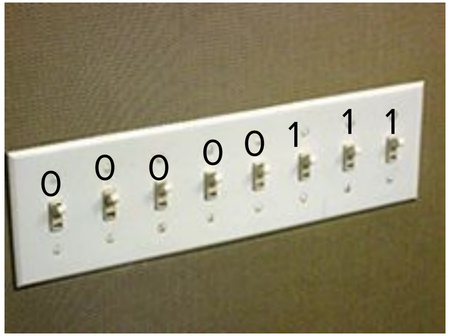
The version is the first item included in a bitcoin block, block header - These version bits are signals used by miners signaling to nodes what consensus rules are compatible (or potentially incompatible.)

Soft forks should have no relevance to and older node, but signal additional features to newer nodes.

So what is a ‘Bit Vector’ and why is it important to BIP 9

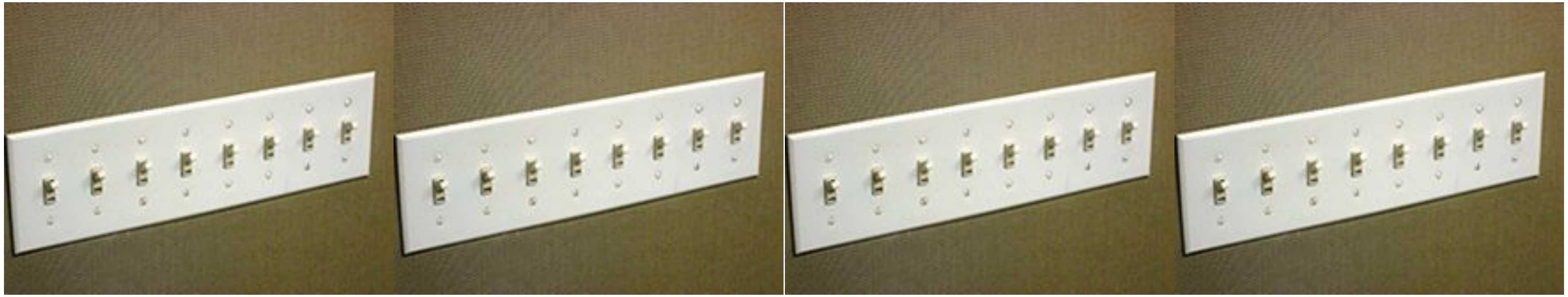
All data in computer memory is arranged into a sequence of ‘0’s and ‘1’s these represent a True or False. With 0 usually meaning False and 1 meaning True.

‘Bit Vectors’ are a series of 0s and 1s such as *‘00000111’* where there the zeros mean something is False and the ones represent something is true. The best example that I can come up with is it’s similar to a light switch panel in your home. Each switch maps to one light and the light is on or off.

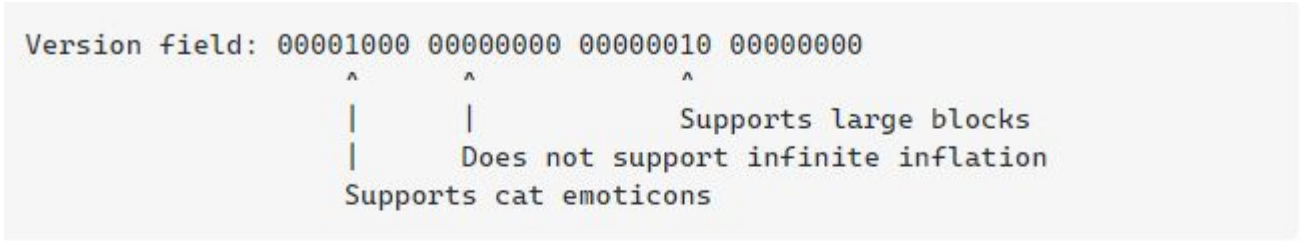


Each switch maps to one light, the number of active lights in the home would be equal to the number of switches in the on position. In our previous example of 00000111 that would mean the 5 switches on the left are off and 3 on the right are on.

The Block Header Version Field Visualized



A series of 32 switches that can be turned on or off. Currently the first 3 switches on the left are (OFF OFF ON)



```
Version field: 00001000 00000000 00000010 00000000
                   ^      ^            ^
                   |      |            Supports large blocks
                   |      Does not support infinite inflation
                   Supports cat emoticons
```

Example Version Field Source - https://bitcoin.stackexchange.com/a/39224

BIP 9 - Version Bits with Timeout and Delay - What is it?

- BIP 9 is a method to use the version field of a blocks header to signal activation support of a softfork.
- BIP 9 allows parallel softforks by representing each softfork via it’s own bit in the version field. A ‘bit vector` is the data structure used to store the version bits.

BIP9 RESERVED BITS

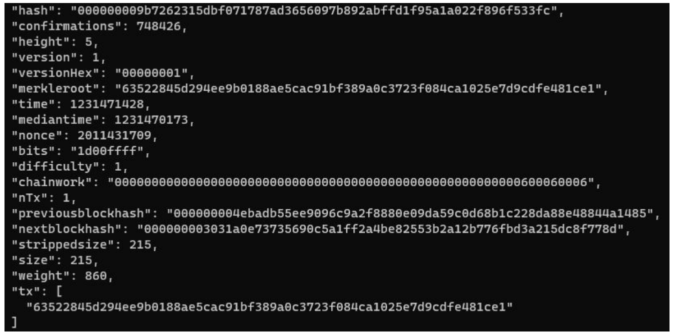
ASIC BOOST MANGLING

Used Version Number Portion ...

ORIGINAL VERSION BITS

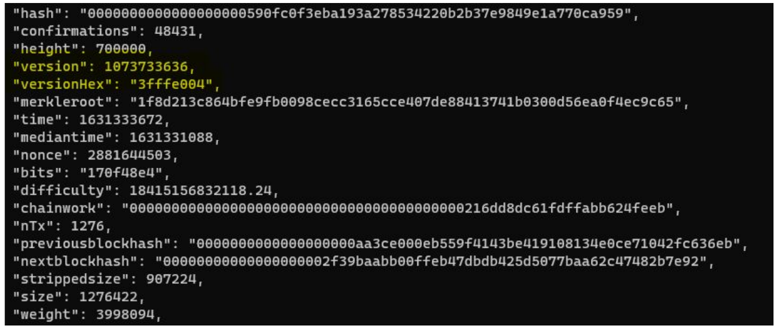
Block #5 Version = 32b’000....001

Block #700000 Version =32b’00111111111111111110000000000100



```
"hash": "000000009b7262315dbf071787ad3656097b892abffd1f95a1a022f896f533fc",
"confirmations": 748426,
"height": 5,
"version": 1,
"versionHex": "00000001",
"merkleroot": "63522845d294ee9b0188ae5cac91bf389a0c3723f084ca1025e7d9cdfe481ce1",
"time": 1231471428,
"mediantime": 1231470173,
"nonce": 2011431709,
"bits": "1d00ffff",
"difficulty": 1,
"chainwork": "0000000000000000000000000000000000000000000000000000000600060006",
"nTx": 1,
"previousblockhash": "000000004ebadb55ee9096c9a2f8880e09da59c0d68b1c228da88e48844a1485",
"nextblockhash": "000000003031a0e73735690c5a1ff2a4be82553b2a12b776fbd3a215dc8f778d",
"strippedsize": 215,
"size": 215,
"weight": 860,
"tx": [
  "63522845d294ee9b0188ae5cac91bf389a0c3723f084ca1025e7d9cdfe481ce1"
]
```

Version 1



```
"hash": "0000000000000000000590fc0f3eba193a278534220b2b37e9849e1a770ca959",
"confirmations": 48431,
"height": 700000,
"version": 1073733636,
"versionHex": "3fffe004",
"merkleroot": "1f8d213c864bfe9fb0098cecc3165cce407de88413741b0300d56ea0f4ec9c65",
"time": 1631333672,
"mediantime": 1631331088,
"nonce": 2881644503,
"bits": "170f48e4",
"difficulty": 18415156832118.24,
"chainwork": "00000000000000000000000000000000000000000216dd8dc61fdffabb624feeb",
"nTx": 1276,
"previousblockhash": "0000000000000000000aa3ce000eb559f4143be419108134e0ce71042fc636eb",
"nextblockhash": "0000000000000000002f39baabb00ffeb47dbdb425d5077baa62c47482b7e92",
"strippedsize": 907224,
"size": 1276422,
"weight": 3998094,
```

SEGWIT AND CSV BITS

Version 4

So how do BIP 9 and Soft forks convergere?

- Since nodes without hashpower can't effectively signal support for a soft fork and/or feature implementation in blocks. Nodes with hashpower are the primary signaling method for support of a soft fork. In the case of BIP 9. Soft forks are backwards compatible so the network of nodes with or without hashpower can reject these version flags and just not utilize the soft fork features.
- BIP 9 essentially is a method for miners to signal in a block minted onto the timechain their support for a specific soft fork. Bip 9 due to the removal of using integer based versioning make it so miners can signal for multiple soft forks in parallel.

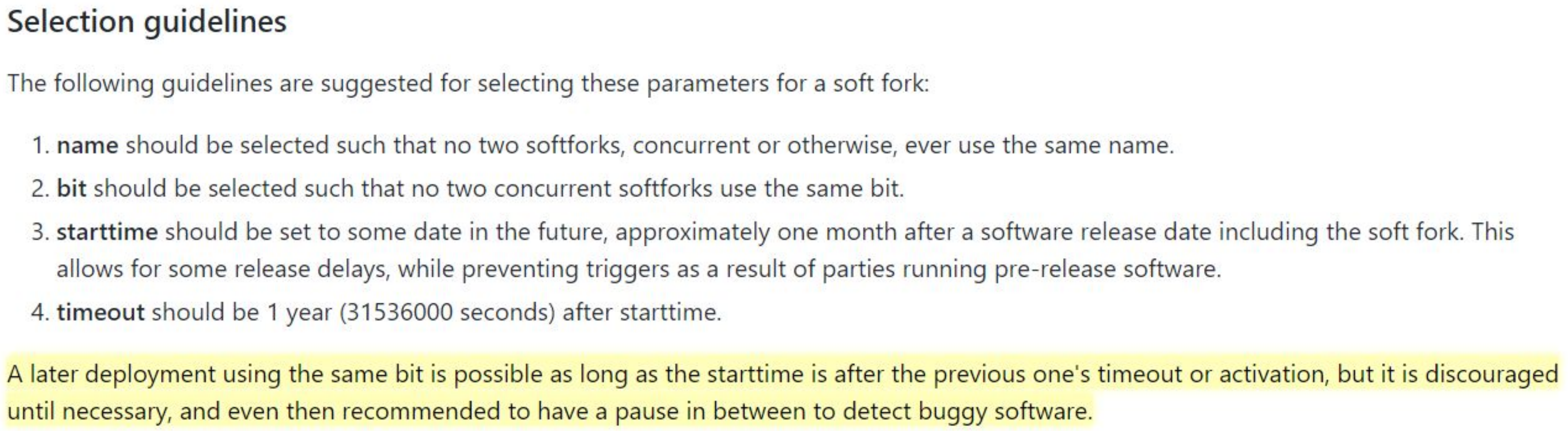
Motivation Summary - Improve on BIP34

- BIP 34 - BIP 9 Fixes the inability to permanently reject a proposal
- BIP 34 - BIP 9 Fixes need to coordinate soft fork proposals
- BIP 34 - BIP 9 Removes the need to compare version numbers which limits it to one set of features per versions, flags in a bit vector allow for multiple features to be signaled in parallel.
- BIP 34 - Creates unnecessary restrictions in the version bit field. Fix this with a 'Bit Vector' Implementation.

Specification

- The proposed soft fork is coded into bitcoin core with the following data
    - Name
    - Bit Position
    - Start Time (time which the bit position gains meaning)
    - TImeout (if signaling ratio is not achieved the softfork is considered failed)

Selection continued… Gloss over this - Show for Clarity



Selection guidelines

The following guidelines are suggested for selecting these parameters for a soft fork:

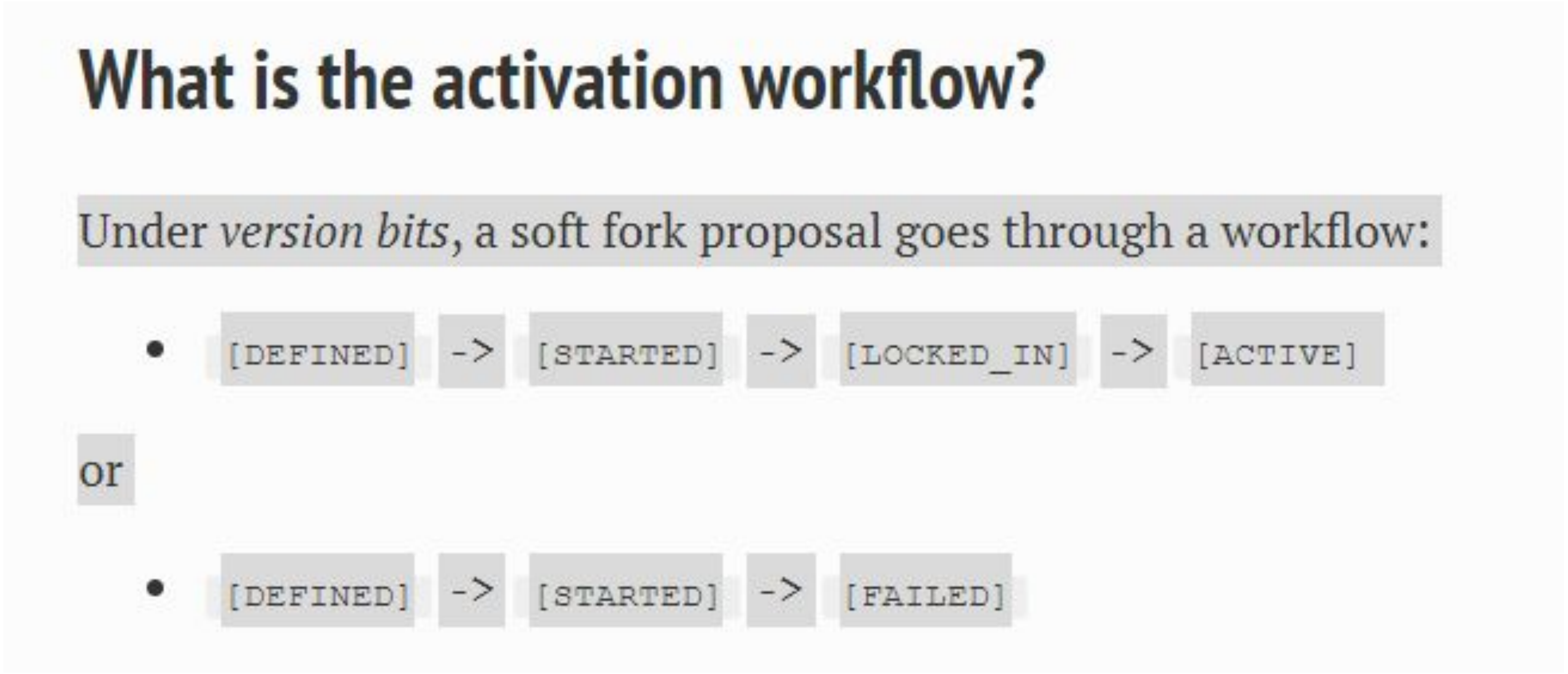
1. **name** should be selected such that no two softforks, concurrent or otherwise, ever use the same name.
2. **bit** should be selected such that no two concurrent softforks use the same bit.
3. **starttime** should be set to some date in the future, approximately one month after a software release date including the soft fork. This allows for some release delays, while preventing triggers as a result of parties running pre-release software.
4. **timeout** should be 1 year (31536000 seconds) after starttime.

A later deployment using the same bit is possible as long as the starttime is after the previous one's timeout or activation, but it is discouraged until necessary, and even then recommended to have a pause in between to detect buggy software.

Bit flag - Important but discussed earlier

- Little endian representation
- 29 bits for flagging - 2 bits for future upgrades to different mechanisms
- Miners ‘should’ continue setting the bit in LOCKED_IN so uptake is visible

Only 2 Valid Work Flows - LOCKED_IN or FAILED



What is the activation workflow?

Under *version bits*, a soft fork proposal goes through a workflow:

- `[DEFINED]` `->` `[STARTED]` `->` `[LOCKED_IN]` `->` `[ACTIVE]`

or

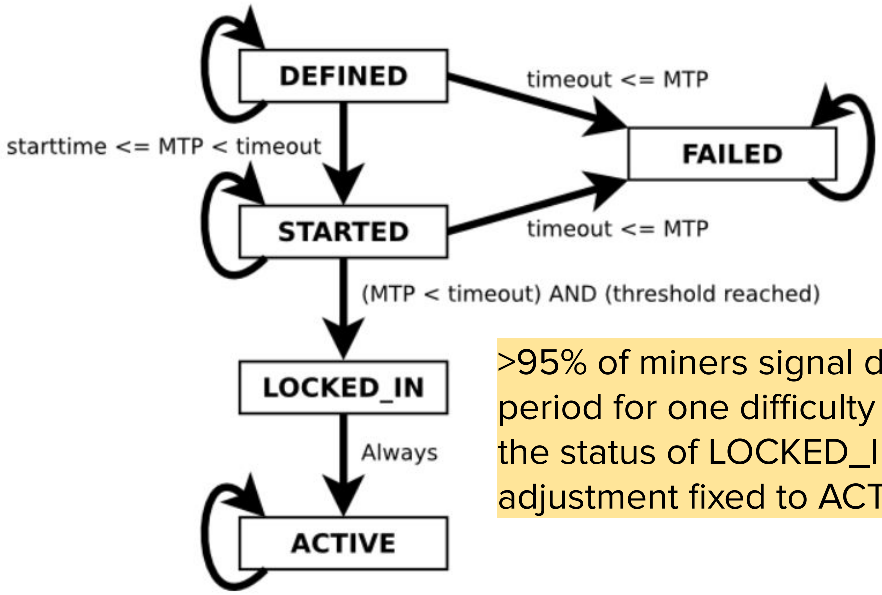
- `[DEFINED]` `->` `[STARTED]` `->` `[FAILED]`

BIP 9 Consensus Rules

New consensus rules

The new consensus rules for each soft fork are enforced for each block that has ACTIVE s

State transitions



>95% of miners signal during starting period for one difficulty adjustment to get the status of LOCKED_IN. 1 difficulty adjustment fixed to ACTIVE

ACTIVATED VIA. BIP9
- Segwit (BIT 1)
- Check Sequence Verify (BIT 0)

During the started phase threshold counts for the signaled version bits are tallied together to move into the locked in phase.

Warnings to Clients

BIP 9 also expresses the use of a waning mechanism which when the soft fork(s) are locked in the client should announce loudly to the user. Then warn even more LOUDLY to the user that the softfork is in the active status.

Finally: Support For Future Changes.

1. Modified Thresholds - The threshold of BIPs 34s 95% doesn’t have to be maintained forever. Open to change lower or higher
2. Conflicting Soft forks - These can be handled with the completing soft forks excluding one another
3. Multistage Soft forks - These are supported by handling the bits as integers.

THANK YOU TO EVERYONE

- NOTES:
  - I am human, there may be mistakes. Do not take this as the end all be all of this topic.
  - Please notify me of any mistakes right away
  - This is a technical analysis and does not cover the scope of actual implementation and consensus.
  - I hope everyone learned something.

Sources

1. https://github.com/bitcoin/bips/blob/master/bip-0009.mediawiki
2. https://bitcoincore.org/en/2016/06/08/version-bits-miners-faq
3. https://learnmeabitcoin.com/explorer/block/version/3fffe004
4. https://lists.linuxfoundation.org/pipermail/bitcoin-dev/2017-February/013643.html
5. https://github.com/bitcoin/bips/blob/master/bip-0148.mediawiki