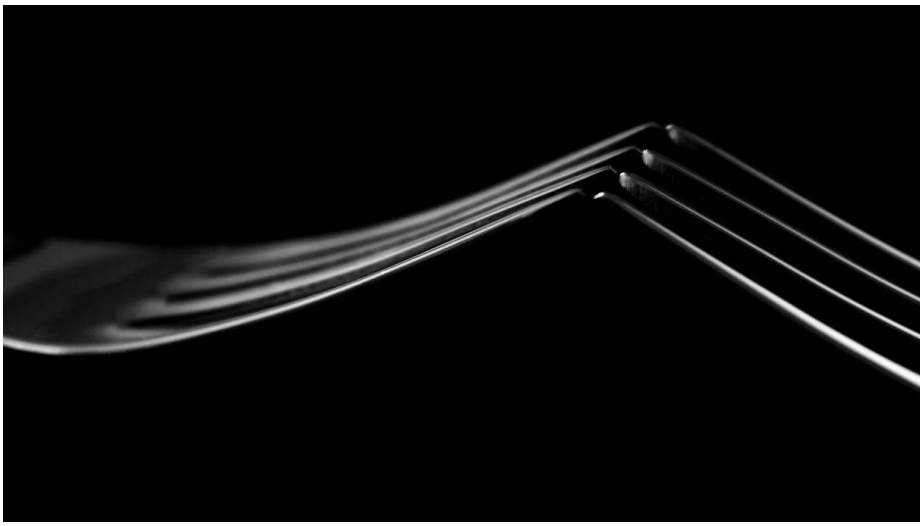
BitMEX Research

The Art of Making Softforks: Protection by Policy Rule

1 February 2018



(Source: gryb25)

Abstract

In this article, following on from our piece on the history of consensus forks, guest writer Dr. Johnson Lau explains the distinction between policy rules and consensus forks. He explains why it may be safer to introduce new softforks when the proposed rule is already covered by policy rules (non-standard behaviour), as this may mitigate or reduce some of the risks normally associated with changing the consensus rules.

BitMEX Research

Filtering out the hype with unbiased, evidence-based reports on the crypto-coin ecosystem.

BitMEX Research is also active on Twitter and Reddit.

research.bitmex.com

Previous reports:

The Lightning Network
(25/01/18)

Mining Incentives, Part 3: Short Term vs Long Term
(17/01/18)

A Complete History of Bitcoin's Consensus Forks
(28/12/17)

Bitcoin Gold: Investment Flow Data
(21/12/17)

Consensus rules and softforks

Consensus rules determine whether a transaction or a block is valid or not. Every user and miner on the Bitcoin network is expected to adhere to the same set of consensus rules, so they will all agree to a single ledger.

A softfork is an event when the majority of users and/or miners decide to adopt a stricter set of consensus rules, which makes some previously valid transactions/blocks invalid, but not the opposite. If the majority enforces the new rule set, any violating fork would (statistically) never catch up to the stricter fork in terms of total proof of work. The minority with the old rules set will always follow the longer and stricter fork, so everyone on the network would still agree to a single ledger.

Policy rules and consensus rules

While consensus rules are the only criteria for determining transaction validity, relaying or mining nodes may prefer some kinds of transactions over others. For example:

- As spam control, transactions with very low fees or "sand outputs" (outputs with very low value) are rejected.
- Some miners refused to include "on-chain casino" transactions, considering them spammy.
- Transactions with an unknown version are rejected (currently only version 1 and 2 are "known").
- Transactions with exotic scripts (i.e., not P2PKH, P2SH, v0 segwit, or a few other cases) and unknown NOPx codes (currently only OP_NOP2 and OP_NOP3 are known) are rejected.
- "Replace by fee" and "child pay for parent" are also policy rules, as they determine which transactions are preferred by miners.

By definition, policy rules *must* be at least as strict as consensus rules. Obviously, no miners would like to include invalid transactions in a block (which will lead to a loss of mining reward) or to relay them (which will get them banned by peers).

While policy rules could be stricter than consensus rules, it is important to note that policy rules do not determine the validity of transactions. Once a transaction is included in a valid block, all network nodes will accept it even if it violates some policy rules.

It is also important to note that policy rules are local, while consensus rules are universal. That means different network nodes might have different policy rules but they will still agree to the same blockchain ledger as long as they are running the same consensus rules.

Transactions that violate a policy rule are sometimes called “non-standard transactions”, distinguishing them from “invalid transactions”.

Policy rules and softforks

Ideally, all miners should have upgraded to the new, stricter rule sets on or before the activation of a softfork. Financially, they have a strong incentive to do this, as mining an invalid block (in terms of the new rules) would lead to significant monetary loss. However, in a decentralized system like Bitcoin, this is not guaranteed.

Although miners are expected to pay attention to any proposed rule changes and take timely action, miners who build invalid blockchain might lead to market disruption and monetary loss for ordinary users. Therefore, any well-planned softforks should bear this in mind and minimize the risks.

The trick is to make a softfork only if it is covered by existing, widely adopted policy rules. Miners with the policy rules who are unaware of the new consensus rules would refuse to include such transactions by default, so they would never include transactions that are invalid in terms of the new consensus rules. Some cases in Bitcoin history illustrate this.

A worker is adding a “Road Closed” sign to a route that is not being used due to an obstruction that existed before the sign was placed. The new traffic rules only prevent behaviour that was already “non-standard” and disruption is therefore minimal.

Case Study	Description
BIP65: Check lock-time verify	OP_NOP1 to OP_NOP10 originally had no meaning in the Bitcoin script language. While they are counted as one operation (there is a limitation of 201 operations in a script), practically, they are skipped during transaction validation. However, a policy rule has been included in Bitcoin Core since version 0.10 to reject OP_NOPx by default. BIP65 is a softfork introduced in Bitcoin Core 0.12 to redefine OP_NOP2 as OP_CHECKLOCKTIMEVERIFY (OP_CLTV). OP_CLTV checks if the top stack value is greater than the transaction's nLockTime field (along with a few more conditions). If any of the conditions are matched, the transaction is considered as invalid. Otherwise, OP_CLTV is skipped like OP_NOP2. New nodes would always enforce the new consensus rules after softfork activation. Yet even before the softfork was activated, the original OP_NOP2 policy rule was replaced by the OP_CLTV rules (which is okay, since OP_CLTV rules are stricter than the original OP_NOP2 consensus rules). Legacy mining nodes would not perform the nLockTime check. However, as long as they were running version 0.10 or above, the default OP_NOP2 policy rule would prevent them from including ANY transactions with OP_CLTV, valid or not. As a result, legacy mining nodes of 0.10 or above would never actively produce an invalid block with respect to the new OP_CLTV consensus rules.
BIP68: Relative lock-time using sequence numbers	nSequence is a field in Bitcoin transactions, which was essentially unused. The idea of BIP68 was to use the nSequence field for the purpose of relative lock-time, which is a very important building block of advanced transactions such as payment channels and the Lightning Network. However, the nSequence field has been ignored since the very first version of Bitcoin, and miners would accept any transaction with any nSequence value. There was no policy rule governing nSequence value, therefore a safe softfork could not be done as simply as OP_CLTV. The trick was to use the transaction-version field (nVersion). Since version 0.7, non-version-1 transactions are rejected by a policy rule. To leverage this, BIP68 requires that the new rules for nSequence are enforced *only* if the transaction version is 2 or above (or below 0, to be precise). Therefore, legacy mining nodes would not produce any BIP68-violating block, since they won't include any non-version-1 transactions by default. An attacker could not "turn off" BIP68 by simply changing the transaction version, since the version is covered by signature. This is also the only instance in which the transaction version is associated with consensus rules.
BIP141: Segregated witness	Segregated witness (segwit) is a softfork to fix transaction malleability by redefining a certain script pattern. In BIP141, the pattern is an output script (or P2SH redeemscript) which starts with a single OP_x (x = 0 to 16), followed by a canonical data push between 2 and 40 bytes. However, this is not what it was originally proposed. In the first draft, the witness-program pattern was a single push between 2 and 41 bytes.

	A policy has been implemented since v0.6 to reject transactions that spend exotic scripts (i.e. not P2PKH, P2SH, and a few more types). The first draft of the witness program was indeed non-standard in this regard. The problem is with the witness program when wrapped in P2SH. Before v0.10, the policy rules would also reject any exotic P2SH scripts. This rule was greatly relaxed in v0.10, and the original witness-program design was not covered. A few alternative proposals were considered: • A new transaction nVersion (like BIP68) does not work. If the new consensus rule is "segwit rules are enforced only if nVersion is larger than 2", an attacker could steal all money stored in segwit outputs by changing the nVersion (since the nVersion is covered only by the segwit signature, which is not checked when nVersion is 2 or below). • An OP_NOPx might be used to label a witness program. However, this would make all witness programs 1 byte bigger, and occupy the limited OP_NOPx space. The final version made use of the so-called "clean stack" policy rule from BIP62. Although BIP62 is now withdrawn, its rules are still enforced as policy. "Clean stack" requires that script evaluation must end with one and only one stack item. The final witness-program design, however, leaves two items on the stack. This is valid by consensus but violates "clean stack" policy.
Failing example: BIP16 and pay-to-script hash (P2SH)	BIP16 was the first planned softfork on Bitcoin. It was activated when 55% of hash power signalled readiness (compared with the 80% to 95% currently in use). Before P2SH was introduced, there was no policy rule for checking the form of spending output. As a result, a significant number of miners kept creating invalid blocks, occasionally long chains, months after softfork activation.
Failing example: Segregated witness on Litecoin	Not long after the Bitcoin segwit implementation was finalized, Litecoin started to integrate the segwit code. However, while segwit was released in Bitcoin Core 0.13.1, the last Litecoin version at that time was 0.10.4, which did not include the "clean stack" rule. Litecoin developers tried to fix the problem by adding an extra consensus rule to segwit that required the block version to be at least 0x20000000, hoping that would force miners to upgrade. It turned out that all miners upgraded right before the activation (with the last large miner upgrading a few hours before), and no fork was created due to the lack of "clean stack" in the last release. Should a large mining pool have failed to upgrade at the last minute, the extra-block version rule would have provided little or no protection. This will be discussed in a future article.

Policy protection is not a panacea

At this point, a reader might find that the policy-protection trick described above would only prevent un-upgraded miners from actively making the first invalid block after softfork activation. However, should such an invalid block be somehow created, un-upgraded miners would still accept it and extend such a blockchain if it had more proof of work. So, this is a way to only reduce but not eliminate the chance of an accidental chain split at softfork activation. This issue is also particularly problematic if a significant number of miners are using different full-node implementations, which might not have the same policy rules.

Dr. Johnson Lau, Bitcoin Protocol Developer

Disclaimer

Transacting on BitMEX is not offered or available to any resident of (I) the United States of America, (ii) Cuba, Crimea and Sevastopol, Iran, Syria, North Korea, Sudan, or any other sanctioned jurisdiction, or (iii) any jurisdiction where the services offered by BitMEX are restricted.

This material should not be the basis for making investment decisions, nor be construed as a recommendation to engage in investment transactions and is not related to the provision of advisory services regarding investment, tax, legal, financial, accounting, consulting or any other related services, nor is a recommendation being provided to buy, sell or purchase any good or product.

Any views expressed are the personal views of the authors of the report. BitMEX (or any affiliated entity) has not been involved in producing this report and the views contained in this report may differ from the views or opinions of BitMEX.

The information and data herein have been obtained from sources we believe to be reliable. Such information has not been verified and we make no representation or warranty as to its accuracy, completeness or correctness. Any opinions or estimates herein reflect the judgment of the authors of the report at the date of this communication and are subject to change at any time without notice. BitMEX will not be liable whatsoever for any direct or consequential loss arising from the use of this publication/communication or its contents.