```
BIP: 326
Layer: Applications
Title: Anti-fee-sniping in taproot transactions
Author: Chris Belcher <belcher@riseup.net>
Status: Draft
Type: Informational
Comments-Summary: No comments yet.
Comments-URI: https://github.com/bitcoin/bips/wiki/Comments:BIP-0326
Created: 2021-06-10
License: CC0-1.0
Post-History: 2021-6-10: https://lists.linuxfoundation.org/pipermail/bitcoin-dev/2021-June
```

Abstract

This document proposes a certain type of wallet behaviour which uses BIP341 taproot[1]. It provides a greater anonymity set for off-chain protocols which will make use of point-time-locked contracts (PTLCs) such as CoinSwap, Lightning and Discrete Log Contracts.

Motivation

With taproot recently added to bitcoin, and wallet software about to implement taproot wallets, we are in a unique position to improve the privacy of off-chain protocols if we act soon.

Taproot allows for point-time-locked contracts (PTLCs) as a more private replacement for hash-time-locked contracts (HTLCs). If an off-chain contract (for example a Lightning channel) is closed using a PTLC instead of an HTLC, then the blockchain will just see a regular taproot script instead of a hash value and preimage. However, if a contract is closed using the timelock path, then the blockchain will either see a OP_CHECKSEQUENCEVERIFY opcode or a nSequence value in the transaction, neither of which are very common today, and this would mark the closing transaction as something special and unusual.

This BIP proposes to improve the privacy and fungibility of off-chain protocols by having on-chain wallets like Bitcoin Core also set the nSequence field in their taproot transactions as in BIP68[5]. This would be in place of their regular nLockTime anti-fee-sniping protection. The end result is that, if an observer of the blockchain sees a taproot spend with an nSequence value, then that could be either: a regular spend from a wallet, or an off-chain settlement transaction spent with a timelock. The two cases would be indistinguishable, and this could greatly improve the privacy and fungibility of bitcoin. The community and wallet developers should act now to implement this so that the anonymity set of nSequence transactions starts to be built up as soon as taproot itself becomes adopted by wallets.

Background

Fee sniping

Fee sniping is a hypothetical outcome of bad incentives to bitcoin mining in the low-inflation future. For a large miner the value of the transactions in the best block and the mempool can be exceeded by the cost of deliberately attempting to mine two blocks to orphan the best block. However with anti-fee-sniping protection using nLockTime or nSequence the bad miner will soon run out of transactions that can be put in the first block, which means they now need to go in the second. Anti-fee-sniping adds to the incentive to move the blockchain forward.

The nLockTime field is being used this way today. It is implemented in Bitcoin Core[2] and Electrum[3], and adopted by approximately 20% of all recent transactions[4].

Absolute vs relative locktime

nLockTime is an absolute lock time, it allows the transaction to only be mined after a certain block height or unix time. The widespread adoption of it might have provided a good anonymity set for off-chain protocols. Unfortunately those protocols also commonly use relative lock times, because it allows contracts (for example Lightning payment channels or CoinSwaps) to remain open indefinitely as the countdown clock only starts ticking when the closing transaction is confirmed.

Absolute locktimes are also still used, so we should keep using nLockTime, but also often use nSequence.

Transaction pinning

Transaction pinning[8] is a method for making fee bumping prohibitively expensive by abusing node protections against attacks that can waste bandwidth, CPU, and memory. This can make fee management more difficult in multipart contract protocols (such as Lightning Network or CoinSwap). One possible way of solving the problem is to include a 1-block relative timelock '1 OP_CSV' to all spend paths, making it impossible to spend the unconfirmed UTXO. Such a 1-block locktime can also be created with an nSequence value of 1. Many on-chain transactions in bitcoin spend inputs that were created just one or two blocks ago, following this BIP such transactions with 'nSequence=1' would also provide cover traffic for off-chain transactions which disable transaction pinning.

Specifications

When wallets create transactions spending UTXOs protected by BIP341 taproot, they should set either an nLockTime value or nSequence values to discourage fee sniping, by allowing the transaction to only be mined in the next block after the tip, not the current block. This BIP suggests 50% probability for using nLockTime and 50% for nSequence. If nSequence is set it should apply to at least one of the inputs of the transaction, if it has multiple inputs. It is suggested that on-chain wallets pick an input randomly.

Wallets should also have a second random branch which sets the nLockTime or nSequence value even further back, so that transactions that are delayed after signing for whatever reason (e.g. high-latency mix networks) have better privacy. Existing behaviour is that with a probability of 10%, choose a random number between 0 and 99, and subtract it from the current block height. See the Bitcoin Core and Electrum source codes linked in the references for an example.

nSequence can only encode up to 65535 for the block distance[5] so if the UTXOs being spent have more than 65535 confirmations, then the wallet should use nLockTime instead.

Pseudocode

```
def apply_anti_fee_sniping_fields(transaction, rbf_set):
    # bip68 requires v=2
    transaction.version = 2
    # Initialize all nsequence to indicate the requested RBF state
    # nsequence can not be 2**32 - 1 in order for nlocktime to take effect
    for input in transaction.inputs:
        if rbf_set:
            input.nsequence = 2**32 - 3
        else:
            input.nsequence = 2**32 - 2
    # always set nlocktime if any of the transaction inputs have more
    # confirmations than 65535 or are not taproot inputs, or have
    # unconfirmed inputs
    # otherwise choose either nlocktime or nsequence with 50% probability
    if not rbf_set || any(map(lambda input: input.confirmations() > 65535
            || !input.is_taproot() || input.confirmations() == 0,
            transaction.inputs)) || randint(2) == 0:
        transaction.nlocktime = blockchain.height()
        if randint(10) == 0:
            transaction.nlocktime = max(0, transaction.nlocktime
            - randint(0, 99))
        # nsequence must be set in order for nlocktime to take effect
    else:
        transaction.nlocktime = 0
        input_index = randint(len(transaction.inputs))
        transaction.inputs[input_index].nsequence = transaction.inputs\
            [input_index].confirmations()
        if randint(10) == 0:
            transaction.inputs[input_index].nsequence = max(1,
                transaction.inputs[input_index].nsequence - randint(0, 99))
```

Compatibility

This BIP doesn't need any consensus changes. It can be adopted unilaterally and gradually by wallets. Although for greater privacy it would be good for software to adopt it as soon as possible. Ideally during the process of developers implementing their taproot wallets, so that when taproot starts to be used it will already include the nSequence code.

All wallet software already keeps track of how many confirmations its UTXOs have, so the information required to set the nSequence field is already available.

Acknowledgements

Originally suggested by David Harding[6] and mentioned to me by ZmnSCPxj.

Thanks to craigraw for suggesting a new value for input nsequence in the absolute locktime case[7].

Copyright

References

[1] https://github.com/bitcoin/bips/blob/master/bip-0341.mediawiki

[2] https://github.com/bitcoin/bitcoin/pull/2340

[3] https://github.com/spesmilo/electrum/blob/7e6d65ec11c0dccfc24478471c5951d3ae586937/electrum/wallet.py#L211-L224

[4] https://txstats.com/dashboard/db/blocks-statistics?panelId=4&fullscreen&orgId=1

[5] https://github.com/bitcoin/bips/blob/master/bip-0068.mediawiki

[6] https://lists.linuxfoundation.org/pipermail/lightning-dev/2020-January/002412.html

[7] https://github.com/sparrowwallet/sparrow/issues/161#issuecomment-925003231

[8] https://bitcoinops.org/en/topics/transaction-pinning/