```
BIP: 301
Layer: Consensus (soft fork)
Title: Blind Merged Mining (Consensus layer)
Author: Paul Sztorc <truthcoin@gmail.com>
        CryptAxe <cryptaxe@gmail.com>
Comments-Summary: No comments yet.
Comments-URI: https://github.com/bitcoin/bips/wiki/Comments:BIP-0301
Status: Draft
Type: Standards Track
Created: 2019-07-23
License: BSD-2-Clause
```

Abstract

Blind Merged Mining (BMM) allows miners to mine a Sidechain/Altcoin, without running its node software (ie, without "looking" at it, hence "blind").

Instead, a separate sidechain user runs their node and constructs the block, paying himself the transaction fees. He then uses an equivalent amount of money to "buy" the right to find this block, from the conventional layer1 Sha256d miners.

Motivation

"Merged-Mining" (MM) allows miners to reuse their hashing work to secure other chains (for example, as in Namecoin).

However, traditional MM has two drawbacks:

1. Miners must run a full node of the other chain(s). (Thus, they must run "non-Bitcoin" software which may be buggy.)
2. Miners are paid on the other chain, in Alt-currency. (Miners who MM Namecoin, will earn NMC.)

Notation and Example

Note: We use notation side:* and main:* in front of otherwise-ambiguous words (such as "block", "node", or "chain"), to sort the mainchain version from its sidechain counterpart. We name all sidechain users "Simon", and name all mainchain miners "Mary".

Example: imagine that a sidechain block contains 20,000 txns, each paying a $0.10 fee; therefore, the block is worth $2000 of fee-revenue. As usual: the sidechain's coinbase txn will pay this $2000 to someone (in this case, "Simon"). Under Bip301, Simon does no hashing, but instead makes one layer1 txn paying $1999 to the layer1 miners ("Mary").

Upon finding a sidechain block worth $2000...
Item
Runs a sidechain node?
How much hashing?
Coins collected, on Layer2
Coins paid out, on Layer1
Coins rec'd, on Layer1
d(Net Worth)

Bip301 makes this specialization-of-labor trustless on layer1. If Mary takes Simon's money, then she must let Simon control the side:block.

Specification

Bip300 consists of two messages: "BMM Accept" and "BMM Request". These govern something called "h*".

So we will discuss:

1. h* -- The sidechain's hashMerkleRoot, and why it matters.
2. "BMM Accept" -- How h* enters a main:coinbase. When Mary "accepts" a BMM Request, Mary is *endorsing a side:block*.
3. "BMM Request" -- Simon offering money to Mary, if (and only if) she will Endorse a specific h*. When Simon broadcasts a BMM Request, Simon is *attempting a side:block*.

h*

h* ("h star") is the sidechain's Merkle Root hash.

In Bip301, a sidechain's coinbase txn acts as a header (it contains the hash of the previous side:block, and previous main:block). Thus, the MerkleRoot contains everything important.

Note: in Bip301 sidechains, "headers" and "block hashes" do not have significant consensus meaning and are in the design mainly to help with IBD. (In the mainchain, in contrast, headers and block hashes determine the difficulty adjustments and cumulative PoW.)

Above: h* is located in the main:coinbase. h* contains all side:txns, including the side:coinbase. The side:coinbase contains many "header-like" fields, such as the hash of the previous side:block.

Mary controls the main:coinbase, so she may select any h*. Her selection will determine which side:block is "found".

BMM Accept

To "Accept" the BMM proposal (and to accept Simon's money), Mary must endorse Simon's block.

```
For each side:block Mary wishes to endorse, Mary places the following into a main:coinbase 
    1-byte - OP_RETURN (0x6a)
    4-bytes - Message header (0xD1617368)
    32-bytes - h* (obtained from Simon)
```

Code details here.

If these OP_RETURN outputs are not present, then no Requests were accepted. (And, Mary would get no money from Requests.)

It is possible for Mary and Simon to be the same person.They would trust each other completely, so the BMM process would stop here. There would only be Accepts; Requests would be unnecessary.

When Simon and Mary are different people, Simon will need to use BMM Requests.

BMM Request

Simon will use BMM Requests to buy the right to find a sidechain block, from Mary.

```
For each side:block that Simon wants to attempt, he broadcasts a txn containing the followi
        3-bytes - Message header (0x00bf00)
        32-bytes  - h* (side:MerkleRoot)
        1-byte  - nSidechain (sidechain ID number)
        4-bytes - prevMainHeaderBytes (the last four bytes of the previous main:block)
```

We make use of the extended serialization format. (SegWit used ESF to position scriptWitness data within txns; we use it here to position the five fields above.)

The Message header identifies this txn as a BMM transaction. h* is chosen by Simon to correspond to his side:block. nSidechain is the number assigned to the sidechain when it was created. preSideBlockRef allows Simon to build on any preexisting side:block (allowing him to bypass one or more invalid blocks, details below). prevMainHeaderBytes are the last four bytes of the previous main:block (details below).

This txn is invalid if it fails any of the following checks:

1. Each "BMM Request", must match one corresponding "BMM Accept" (previous section).
2. Only one BMM Request is allowed in each main:block, per sidechain. In other words, if 700 users broadcast BMM Requests for sidechain #4, then the main:miner singles out one BMM Request to include.

3. The 4-bytes of prevMainHeaderBytes must match the last four bytes of the previous main:blockheader. Thus, Simon's txns are only valid for the current block, in the block history that he knows about (and therefore, the current sidechain history that he knows about).

Most BMM Request txns will never make it into a block. Simon will make many BMM Requests, but only one will be accepted. Since only one BMM Request can become a bona fide transaction, Simon may feel comfortable making multiple offers all day long. This means Mary has many offers to choose from, and can choose the one which pays her the most.

This BIP allows BMM Requests to take place over Lightning. One method is here. (BMM Accepts cannot be over LN, since they reside in main:coinbase txns.)

Backward compatibility

As a soft fork, older software will continue to operate without modification. To enforce BMM trustlessly, nodes must watch "pairs" of transactions, and subject them to extra rules. Non-upgraded nodes will notice that this activity is present in the blockchain, but they will not understand any of it.

Much like P2SH or a new OP Code, these old users can never be directly affected by the fork, as they will have no expectations of receiving payments of this kind. (As a matter of fact, the only people receiving BTC here, all happen to be miners. So there is less reason than ever to expect compatibility problems.)

As with all previous soft forks, non-upgraded users are indirectly affected, in that they are no longer performing full validation.

Deployment

This BIP will be deployed via UASF-style block height activation. Block height TBD.

Reference Implementation

See: https://github.com/drivechain-project/mainchain

Also, for interest, see an example sidechain here: https://github.com/drivechain-project/sidechains/tree/testchain

References

- http://www.drivechain.info/literature/index.html
- http://www.truthcoin.info/blog/blind-merged-mining/
- http://www.truthcoin.info/images/bmm-outline.txt

Thanks

Thanks to everyone who contributed to the discussion, especially: ZmnSCPxj, Adam Back, Peter Todd, Dan Anderson, Sergio Demian Lerner, Matt Corallo, Sjors Provoost, Tier Nolan, Erik Aronesty, Jason Dreyzehner, Joe Miyamoto, Chris Stewart, Ben Goldhaber.

Copyright