Decentralised Mining Pool for Bitcoin (Draft 0.1)

Kulpreet Singh

February 2021

Abstract

Bitcoin p2pool's usage has steadily declined over the years, negatively impacting bitcoin's ability decentralisation. The primary problems with p2pool are twofold. First, the variance in earnings for miners increases with increasing hashrate participating in the pool. Secondly, payouts to miners require a linearly increasing blockspace with the increase in the number of miners on p2pool. Building a direted acyclic graph (DAG) of miner's shares and the use of payment channels are two proposals trying to alleviate these problems faced by p2pool. In this paper, we present a unified solution that uses a DAG to track miners shares and uses payments channels to reward miners. The shares calculation is verified by all participants of the pool, and the rewards are paid out by a hub. Using the payment channels construction, neither the hub nor the miners can cheat. We show that our approach is incentives compatible and reduces variance in earnings for miners. We also show how the hub maintains its anonymity and resists a DDoS attack by using I2P to communicate with the miners.

1 Motivation

P2Pool [4] bitcoin's decentralisation by allowing miners to select which transactions they mine. This avoids any potential transaction censorship by pool operators. However, the construction used by P2Pool faces a number of problems that eventually lead to miners abandoning the pool. The most often cited problmes are:

1. Large variance in earnings for miners.
2. Large number of dead on arrival shares and stale blocks.
3. Large block space requirement.

The first two problems are a direct consequence of the shares block rate limited to 30 seconds for the bitcoin p2pool. With only one block possible every thirty seconds, any increase of hashrate on P2Pool results in shares competing to be the next block in the p2pool chain. However, if the pool wants to increase the block rate frequency, it doesn't result in an increase in the throughput of the pool in terms of number of shares found. This is because a faster block rate lead to an increase the number of orphans. With miners not being rewarded for orphans, it leads to increase in the variance of the miner rewards. Ethereum's inclusive protocols [10] helps alleviate the problem for the Ethereum blockchain, where small pools can work with a reduced variance in their rewards as shown in the analysis by McElrath [11].

Knowing the challenges faced by P2Pool, we list the goals of a new decentralised mining pool as:

1. Reduced variance for miners with increasing pool hash rate.
2. Payouts for miners with constant size block space requirement.
3. Independent miners can build their own blocks.
4. Provide building blocks for a hash rate futures market. [1]

2 Current Proposals

TerraHash Coin [11], Jute [18] and [17] are some of the attempts to use a DAG for faster block times. However, they focus on changing the consensus layer of bitcoin. The ideas in these proposals allow for miners to produce shares that have conflicting transactions and then apply rules to find a set of transactions acceptable at various cuts of the DAG.

Instead we propose to build a DAG of miner shares to enable faster share times and use this DAG to calculate distribution of payouts between miners. We then propose using payment channels as defined by Belcher [6] to avoid using block space for making payouts to miners. Belcher's construction uses payment channels between federated hubs to pay miners after a block has been successfully mined. The payouts are made after a long enough period, similar to the 100 blocks requirements for spending from coinbase transactions. Miners register with hubs where bitcoin has been locked in to open payment channels to miners. The construction shows how both miners and hubs can't cheat and how the funders of the hub can earn a reward for funding the payment channels.

The two ideas of using a DAG and payment channels for rewards payouts together present a potential path for rebooting P2Pool. In the rest of the paper we present a modified version of TerraHash Coin and show how the various components work together.

3 Decentralised Bitcoin Mining

In this section we present a modified version of TerraHash Coin and show how to build a DAG of miner's shares and use that to determine the payout distribution

[1] We don't elaborat on this goal in this paper, but readers can look up the gist on TerraHash coins https://gist.github.com/kulpreet/19927c7188a4224ce2de43efb3c69370

between miners. We then show that the payout distribution rewards miners for the shares they find and broadcast to the p2p network in a timely manner.

3.1 A DAG of Shares

The braiding the blockchain proposal [11] shows how smaller more frequent blocks can form a directed acyclic graph (DAG) of blocks, with each block pointing to one or more one previous blocks. Blocks in TerraHash Coin can have transactions repeated in different blocks. The proposal describes how repeated and potentially some double spend transactions can be resolved to decide on the state of the ledger at any cut of the DAG.

The rewards that miners earn in the proposal is a coin native to the braid blockchain and is called TerraHash Coin. This coin can then be swapped for Bitcoin. The proposal doesn't yet define how this native coin will be swapped by bitcoin. Some of the suggestions under discussion include using atomic swaps, burning the TerraHash Coin, or using financial instruments like futures of the bitcoin's hash rate to swap TerraHash Coins for BTC.

We propose taking a modified approach, where the blocks of a DAG represent shares of the mining pool. The DAG is maintained by each participant of the p2p network as a replicated database. Shares broadcast by miners include a hash pointer to all previous shares received from other miners, this ensures the DAG is eventually consistent on all p2p participants.

3.1.1 Building Blocks

Each miner builds their own block, selecting transactions according to their own criteria. We call this block the WORK. The description of WORK is then disseminated to the p2p network of miners using the compact block specifications [7].

The miner then starts mining on WORK and generates SHAREs. Each SHARE is mined at a difficulty level chosen by the miner. This difficulty can be dynamically chosen by the miner after each SHARE, depending on miner's observation of the p2p network's hashrate. This dynamic adjustment allows miners to adjust the rate at which they produce SHAREs.

Figure 1 shows the relationship between WORK and its SHAREs. Each WORK created by a miner can result in multiple SHAREs and both the WORK and SHAREs are broadcast to the p2p network.

The nodes in the DAG are SHAREs mined at varied difficulty levels. Each SHARE that matches or exceeds the current bitcoin difficulty starts a new *epoch* for the p2p mining pool. Figure 2 shows l and r as the two valid bitcoin blocks that have been mined such that they meet bitcoin's difficulty at the time, and all the blocks between l and r are in the same *epoch*.

When a miner starts working on a SHARE it includes a reference to the most recent SHARE from all other miners that the miner has received valid shares from. Note, the miner also has access to WORK blocks from all participating miners. If a miner doesn't have the WORK block from another miner, then it rejects any SHAREs received from the other miner.

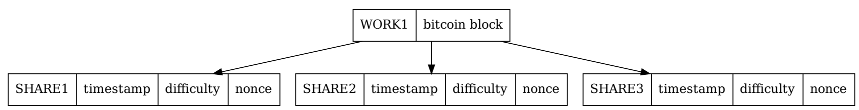

Figure 1: Each WORK generated and shared by a miner is then followed by the SHAREs the miner finds.

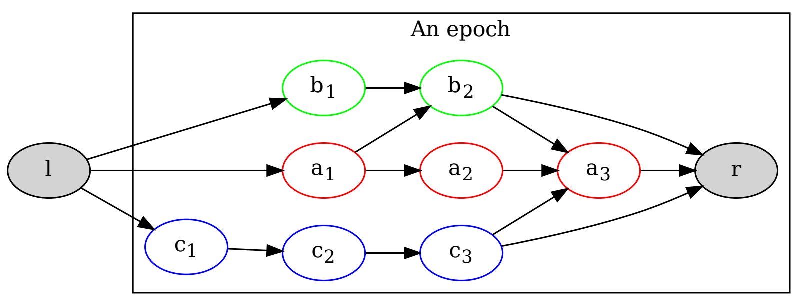

Figure 2: A epoch is defined as all the SHAREs mined between two bitcoin blocks. Here all the SHAREs between l and r are in the same *epoch*.

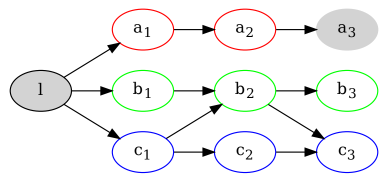

Figure 3: a discovers a share and the reward is not shared with any other miner.

In the next section we then describe how all peers compute their fair share of profits using the DAG of shares. We then show how our reward computation algorithm is incentives compatible as per [16].

3.2 Incentives Compatible Rewards

Each participating node, which includes the miners and the hub, maintains a local replica of the the SHAREs DAG. Each SHARE includes a reference to the shares the miner was aware of when the SHARE was found. The reason for doing so is simple. If a miner a doesn't include the SHAREs of miner b, it signals a failure of communication between the two miners. We introduce a rule that miner a in such a situation should stop including references to b's SHAREs.

then b has a clear signal to stop including the SHAREs of a, and as we will see a miner wants that their SHAREs are referenced by other miners as only then they will be rewarded for their work.

The incentive in lay terms is that all miners should honestly include the SHAREs discovered by other miners, as otherwise they will most likely be excluded by other miners and they will lose the opportunity to be rewarded for their work. We call this the degenerative case of "isolated miners" and argue that miners have no incentives to act in this manner. Figure 3 shows a DAG where all three miners a, b and c are working independently. In such a situation when the miner a discovers a share a_3 which is a valid bitcoin block but the reward is not shared with any other miner.

With the above understanding of why miners will co-operate, we now state the rules to calculate how the block reward should be divided between miners.

[TODO: Convert to algorithm?]

1. Traverse the DAG in reverse order from the SHARE that found a bitcoin block to the previous bitcoin block found and collect a set of shares.

2. From the above set of shares remove all shares that don't have a reverse path to the previous bitcoin block.

3. Distribute the reward between miners weighted by the sum of the difficultly of all SHAREs found by miners.

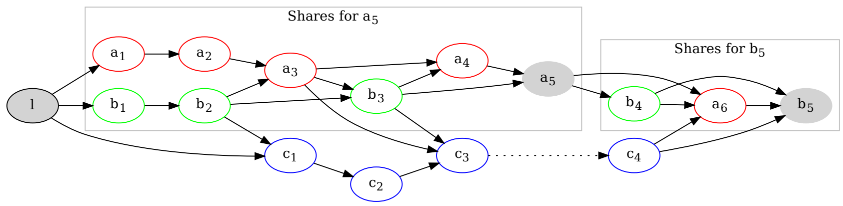

Figure 4: Two epochs in a DAG of shares mined by three mines — a, b and c. The shares in grey meet the bitcoin difficulty at the time they were mined.

As an example consider the p2p network of miners a, b and c with the DAG of shares as shown in Figure 4. In the DAG the set of shares that receive reward proportional to their difficulty are $\{a_1..a_5, b_1..b_3\}$. The shares $\{c_1..c_3\}$ do not receive any reward as they are not reachable from the bitcoin block, a_5, even if they are reachable from l.

For the second bitcoin block b_5 only the miners a and b receive rewards in proportion to the difficulties of their shares $\{b_4, b_5, a_6\}$. c doesn't receive any reward for c_4 as it is doesn't include a reference to the last found bitcoin block a_5.

Given the above rules, we show how they together provide an incentives compatible reward function as defined by [16]. We present an outline of proofs that will be be formalised in future work.

3.2.1 Incentive Compatibility

A reward function is incentive compatible when every miner's best response strategy reports full solutions immediately [16]. Where a "full solution" is a share that meets the bitcoin network's difficulty requirement.

Given the rules in Section 3.2, if a miner finds a bitcoin block the miner wants to get maximum reward possible based on all the shares it has found and therefore is incentivized to announce their SHARE as soon as it finds the block. The longer a miner doesn't announce the bitcoin block to the bitcoin network, the higher the probability that some other miner on the pool will find a different bitcoin block that doesn't reward their latest shares that haven't reached the other miner. Further still, the rewards calculation can be adjusted to give an extra reward to the miner that finds the block. This is similar to what p2pool does and Belcher also mentions in his proposal.

3.2.2 Proportional Payments

As per [16], we require that miners are paid in proportion to the amount of work they have performed.

Since rewards are calculated at the end of an epoch and a miner is incentivized to include hash pointers to the most recent shares seen from all miners, it directly follows that all miners are guaranteed payments for their shares that have reached the miner who found the block. It is also clear here that miners are incentivized to propagate their shares to the network as fast as they can.

3.2.3 Budget Balanced

Again, as per [16] we would like the pool operator to never incur a deficit. From the rules, since rewards are paid at the end of the epoch, a hub pays out rewards without losing or retaining any amount.

3.3 Payment Channels

We propose using Payment Channels for paying miners based on the work by Belcher [6]. The construction of the payments channels we present is similar to Belcher's construction, but there are a few changes and we highlight them before describing the details.

- We use the DAG based scheme to distribute rewards among miners, so we avoid the problem of estimating block rewards before miners have created their shares.
- We use a single hub, and prevent DDoS attacks by keeping the hub hidden using I2P [5, 9] for communication between the hub and miners.

The rest of the construction is the same as presented by Belcher — There is a one-way channel between the hub and each of the miners. The hub updates the state of each channel with appropriate reward after each block is found.

3.3.1 Coinbase

We use Belcher [6] construction where each miner builds a coinbase transaction that can be spent in one of the following three ways:

1. Co-operatively by Hub and Miner, or
2. By the Hub with a hash lock for pre-image `X`, or
3. By the Miner that found the bitcoin block, but after waiting for six months.

The scriptPubKey for the above conditions in the coinbase is shown in Table 1.

These conditions mean that the hub can not spend the coinbase without revealing the pre-image X. This pre-image is included in the construction of payment channels, as we will see in the next section. This use of pre-image in both the coinbase and the payment channel definition guarantees that miners

Coinbase	
`2 H M 2 CHECKMULTISIG`	(cb_1)
`OR`	
`hash(X) + Hub P2WPKH`	(cb_2)
`OR`	
`M and CHECKSEQUENCEVERIFY 6 months`	(cb_3)

Table 1: Coinbase transaction with hub and miner public keys.

get paid for their accumulated payouts if the Hub defects and spends through the cb_2 branch. We discuss how the miner and the hub don't gain by defecting in Section 3.5.

3.3.2 One-way Channels

One-Way payment channels between hub and all miners allow miners to receive payouts without consuming any blockspace in the bitcoin block they mine. The use of payment channels is what makes braidpool scale up without losing blockspace. Braidpool thus avoids losing the fees that can be earned from the blockspace.

Just like in the early versions of proposal by Belcher we use one-way payment channels. One-Way payment channels solve the problem of aggregating a miner's payouts requiring a single blockchain transaction for spending multiple payouts earned from their PoW shares.

We use one-way instead of bidirectional payment channels as an initial implementation for Braidpool. If required we can switch to bidirectional channels [15] allowing miners to spend their mining payouts over the lightning network. However, for now, we deliberately stay away from the complexity of making the hub a lightning node. If there is interest from miners to use the lightning network, we can build that out in future.

Each miner has a one-way payment channel with the hub using a two of two multisig with a time lock of six months. For each payout a miner receives over the payment channel, the hub will charge an agreed upon fees between the miner and the hub. Belcher's proposal recommends a 0.1% fees for the hub.

3.3.3 Payment Channel Transactions

The hub creates a funding channel locking an amount R of bitcoin. The miner then creates a refund transaction, spending the funding transaction and sending the R bitcoin back to the Hub. The refund transaction has a locktime of six months allowing the miner to accumulate their payout over the six month period. The protocol can be extended to allow each miner to agree upon a locktime with the hub. The trade-off will be between the fees charged by the hub and the length of the locktime.

Fund Transaction

Input	Output	
Hub's UTXO (Signed by the hub)	`2 H M 2 CHECKMULTISIG`	(f_1)
	`OR`	
	`2 H' M' 2 CHECKMULTISIG + Hash(X)`	(f_2)
	(R coins)	

Table 2: Fund transaction for payment channel between hub and miner.

Refund Transaction. Locktime 6 months

Input	Output
Fund Tx (Signed by miner)	`P2WPKH Hub's address` (R coins)

Table 3: Refund transaction signed by miner and held by hub.

The funding transaction includes an input from the hub and an output that can be spent in one of the two conditions shown in Table 2. `H` and `M` are the public keys for Hub and Miner, they are called the co-operative keys by Belcher. While `H'` and `M'` are alternative public keys for Hub and Miner, and are called the uncooperative keys in the Belcher proposal.

The hub doesn't broadcast the funding transaction, instead it waits for the miner to create refund transaction. This is the same as any other timelocked one-way channel construction, i.e. the Miner creates a refund transaction with a timelock, signs it and sends it to the hub. The refund transaction is shown in Table 3.

With the refund transaction, if the Miner stops responding, the Hub can get a refund in six months time. However, the hub can be attacked by sending requests to open new channels and locking up the hub's liquidity. Hub responds to a miner's channel open request only after the miner has contributed enough shares. The threshold number of shares required before opening the channel is a configuration parameter for the hub. The miner will still receive the payouts for the shares generated, it is only that the channel opening is delayed.

On receiving the refund transaction, the hub broadcasts the funding transaction. Once the funding transaction is confirmed, the hub can start sending payouts to the miner. These payouts are determined in proportion to the shares found by the miner and included in the DAG as described in Section 3.2.

The payment transactions are updates to the channel where each update increases the earnings of the miner. The payment transactions are signed by the hub using the non-cooperating key, `H'`. The Table 4 shows the structure of the payment transactions.

The hub updates the payment channel for each miner with payouts as determined by the incentives compatible rewards algorithm shown in Section 3.2.

Payment Transaction

Input	Output	

Fund Tx
(Signed by the hub using H')
	`2 H M 2 CHECKMULTISIG`
	`OR`
	`2 H' M' 2 CHECKMULTISIG + Hash(X)`
	(Hub: $R - earnings$; Miner: $earnings$)
		(p_1)
		
		(p_2)

Table 4: Payment channel update transaction sent from hub to the miner.

In the next section we present how the payouts are distributed, we then show how the hub and the miners are disincentivized to cheat.

3.4 Anonymous Payouts

Once a miner mines a share that also meets the currently bitcoin difficulty, the miner immediately broadcasts the block to the bitcoin network. The coinbase of this block as shown in Table 1 can now be spent by

Co-operative Branch: The miner and the hub by both signing the first branch co-operatively.

Hub Branch: The hub alone, by publishing the pre-image `X`.

Miner Branch: the miner alone, after waiting for six months.

The proposal by Belcher specifies a payout algorithm that requires the hub updates the state of all channels, i.e. make payment to all miners. Once it has done so, the miner signs the first branch of the coinbase transaction and hands the coinbase to the hub. The hub can now redeem the entire payout.

However, there is a small issue here about how the miner signing the coinbase can know if the hub has updated the states of all payment channels. One obvious solution might seem like the miner collects acknowledgements from all other miners that they have received the channel update. We still have to deal with the situation where the ACK from some miners never arrives? Or worse still, if a miner purposely doesn't send an ACK to stall the pool.

We propose that instead of requiring ACKs from all miners, we extend the P2P protocol such that all miners store channel updates for all miners and send them out again if required. This approach is similar to the use of *inv* and *getdata* messages in bitcoin. We elaborate further on this in Section 3.6, where we describe how the hub sends all payment updates to all miners through I2P gateways.

3.5 Defecting Does Not Pay

Using the above construction of the payment channel and the distributed payout algorithm, we now show that defecting by the hub or the miner doesn't pay.

3.5.1 Hub Defects

If the hub defects and pays itself from the coinbase it uses the cb_2 branch of the coinbase. In doing so, the hub has to reveal the pre-image `X`. With the pre-image available, all miners will use the p_2 branch of their payment channel transactions and close their channels and receive all payouts earned up to that block.

It is possible that the hub defects on the very first block mined and the miners lose their earnings for that single block. But that will end the pool before it could be useful.

The hub could defect after a few blocks have been mined. In such a case the miners will receive their fair share of earnings for all previous blocks, but it will again be the end of life for the pool.

Remember the hub charges fees to fund the payment channels between the itself and the miners. When the pool ends, the hub loses a profit making opportunity. We argue that the incentive to defect reduces as the size of the pool grows.

If the hub is co-opted, all it can do is deny miners a single block worth of payouts. Miners will immediately close their channels, collect their payouts and re-organise with a different hub.

3.5.2 Miner Defects

A miner that found the block could chose to not sign the co-operative branch of the coinbase. In such a case, the hub will wait for a timeout period much shorter than the locktime on the miner branch. This will close all channels and require that all these channels are opened again. Such an attack by the miner will hurt miners on the network who have not yet earned enough payouts to amortise the cost of closing the channel.

Say a miner starts participating in the pool, and after N blocks becomes the successful miner of a bitcoin block, but refuses to sign the co-operative branch of the coinbase allowing the hub to claim the coinbase reward. In such a case all miners have to close their channels and claim the rewards they have earned. Miners who recently joined the pool stand to lose the most because of the forced closure of channels.

However, the miner also loses any payouts for all the shares it mined since the last block was found by the pool. A malicious miner could contribute a large portion of the pool's hash rate and then refuse to sign the co-operative branch. This will disrupt the functioning of the pool and the censor could be willing to execute such an attack. Again, the only defence here is that the miners re-organise and start a different instance of the pool hiding their activity behind I2P tunnels. We elaborate on our use of I2P in Section 3.6.

3.5.3 Hub and Miner Collude

The hub and the miner could collude where they co-operate to spend the coinbase to themselves without requiring that the hub pays all other miners as per the reward schedule. The motivation for the miner is clear that it earns a big payout, however, such an action by the hub will end the pool and the stream of future profits for the hub.

3.5.4 DDoS on the Hub

The hub can be attacked using a distributed denial of service attack rendering it unable to process requests to open new channels and to distributed payouts to miners. Belcher in his proposal suggested the use of multiple hubs as a defence against such an attack. The proposal also points out that multiple hubs will reduce the liquidity required to open channels with miners.

According to Belcher's proposal, with multiple hubs available on the p2p network miners will open channels to all hubs and receive payouts from all of the hubs. The coinbase is split between hubs in such a way that if any hub defects, the other hubs can still spend the coinbase and split the block reward between them. In this way all miners receive payouts in proportion to the shares contributed by them.

Creating a larger coinbase for multiple hubs scales if we can use Taproot [14, 12, 13] once it is activated. The solution uses staggered timeouts for hubs to spend the coinbase in case of hubs defecting or coming under DDoS attacks. This option is worth exploring once Taproot is enabled and the pool will benefit from multiple hubs. This will be even more useful as with the multiple hubs construction reduces the liquidity requirements for all hubs. The only problem is that in case of hubs defecting, the staggered timeouts can result in a large wait for the payouts to be distributed.

We propose a different solution that requires a single hub. The advantage is a simpler channel construction and allow us to build the mining pool without waiting for Taproot activation. The single hub approach requires that all miners open I2P tunnels to gateways and publish the gateway's address in the shares they broadcast on the p2p network. The hub then can communicate with miners by opening new tunnels and remaining anonymous. We present this approach in the next section.

3.6 Anonymous Hub Miner Communication

The hub and the miner need to exchange messages at the time of channel creation as well updating the channel state when the hub makes a payout to the miners. The channel management messages are exchanged infrequently as compared to the SHAREs broadcast messages and don't have the same timeliness requirements as the shares broadcasts. We propose adoption a solution that doesn't compromise on the hub's anonymity and makes a trade-off with increased latency of the channel management messages.

We use I2P's tunnels to avoid compromising the anonymity of the hub I2P works by setting up separate inbound and outbound tunnels. According to the I2P documentation [2]:

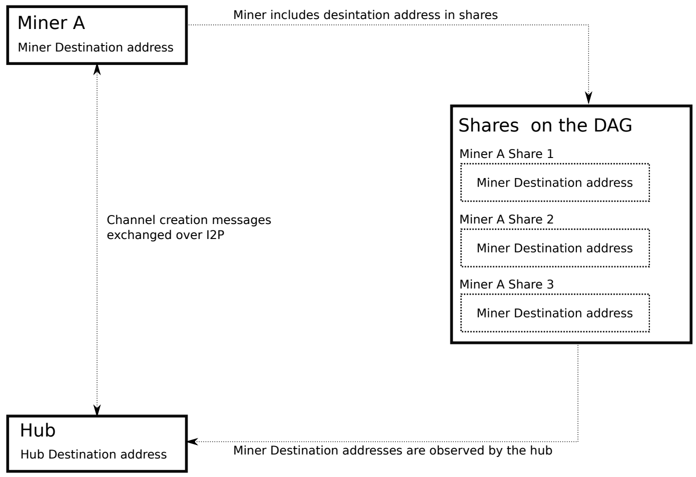

Figure 5: Discovery of miner I2P destination addresses enables communication for creating channels.

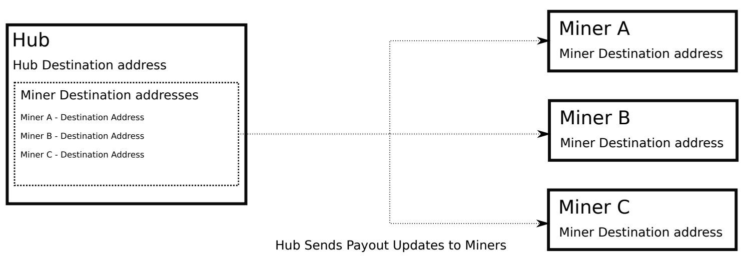

Figure 6: Hub sends one way updates to miners without revealing its destination address.

> A tunnel is a directed path through an explicitly selected list of routers. Layered encryption is used, so each of the routers can only decrypt a single layer. The decrypted information contains the IP of the next router, along with the encrypted information to be forwarded. Each tunnel has a starting point (the first router, also known as "gateway") and an end point. Messages can be sent only in one way. To send messages back, another tunnel is required.

There are two types of control messages exchanged between the hub and the miner: 1. the channel creation messages when the miner first joins the pool and 2. the payout messages whenever a block is found.

Miners will run an I2P node next to their mining controller and setup an incoming tunner. Once the tunnel is ready, miners include their I2P destination address in their share headers. The hub can then send messages to the miner through the I2P tunnel. By hiding their real I.P. address miners can increase their anonymity, and in turn help the hub maintain its anonymity too.

The hub participates in the shares broadcast p2p network and identifies a new miner along with the new miner's destination address. Once the miner has mined for a certain time period, the hub contacts the miner and provides its own destination address in the message [3]. Figure 6 shows how the "Miner A" announces its I2P destination address, enabling the hub to contact the miner for exchanging channel creation messages.

To prevent the tunnel endpoint and gateway from seeing the messages being exchanged, the hub and the miner use I2P's layered encryption, called garlic routing [1]. With the destination addresses of both the new miner and the hub known to each other the two can proceed to anonymously setup a payment channel as described in Section 3.3.

To send payouts as channel state updates, the hub again uses the I2P destination address of each miner and sends a one way message, with no reply destination included in the message. This allows the hub to remain anonymous and makes it relatively hard to DDoS the hub.

4 Future Work

The proposal presents an approach to enable decentralised mining for bitcoin. Apart from the work of describing the various components in detail, we also want to provide results from simulations, formalised proofs of rewards schemes and possible extensions to using multiple hubs.

Before we work on implementing they system, our next step is to simulate p2p mining network using ns-3 [8] and make informed decisions about how large a network a single hub can support. The observations we want to make are how large a p2p network can be sustained without an increase in work lost by miners. Each hub and p2p network can grow as long as miners are communicate WORK and SHARES with each other with bounded latency and can limit their lost work. With a simulation we want to find out the bounds of these.

We want to specify the p2p protocols and the message formats for both the SHAREs propagation and Channel management networks. By publishing the specifications separate from the source code, we aim to receive more feedback from the community. We want to use the model presented in [16] to provide proofs for how the rewards distribution is incentives compatible. We would like to build further on the multiple hubs construction described by Belcher once Taproot is activated on bitcoin.

5 Acknowledgements

Please to review! Many wow! ;)

References

[1] Garlic routing and "garlic" terminology. https://geti2p.net/en/docs/how/garlic-routing.

[2] I2p: A scalable framework for anonymous communication. https://geti2p.net/en/docs/how/tech-intro.

[3] I2p: Streaming library. https://geti2p.net/en/docs/api/streaming.

[4] P2pool. https://en.bitcoin.it/wiki/P2Pool.

[5] Astolfi, F., Kroese, J., and Van Oorschot, J. I2p - the invisible internet project. Web technology report, Media Technology, Leiden University, 2015.

[6] belcher. Payment channel payouts: An idea for improving p2pool scalability. https://bitcointalk.org/index.php?topic=2135429.0.

[7] Corallo, M. Compact block relay. https://github.com/bitcoin/bips/blob/master/bip-0152.mediawiki.

[8] Henderson, T. R., Lacage, M., Riley, G. F., Dowell, C., and Kopena, J. Network simulations with the ns-3 simulator. *SIGCOMM demonstration 14*, 14 (2008), 527.

[9] Hoang, N. P., Kintis, P., Antonakakis, M., and Polychronakis, M. An empirical study of the i2p anonymity network and its censorship resistance.

[10] Lewenberg, Y., Sompolinsky, Y., and Zohar, A. Inclusive block chain protocols. In *Financial Cryptography and Data Security* (Berlin, Heidelberg, 2015), R. Böhme and T. Okamoto, Eds., Springer Berlin Heidelberg, pp. 528–547.

[11] McElrath, B. Decentralized mining pools for bitcoin. https://www.youtube.com/watch?v=91WKy7RYHD4.

[12] Pieter Wuille, Jonas Nick, A. T. Bip 341: Taproot: Segwit version 1 spending rules. https://github.com/bitcoin/bips/blob/master/bip-0341.mediawiki.

[13] Pieter Wuille, Jonas Nick, A. T. Bip 342: Validation of taproot scripts. https://github.com/bitcoin/bips/blob/master/bip-0342.mediawiki.

[14] Pieter Wuille, Jonas Nick, T. R. Bip 340: Schnorr signatures for secp256k1. https://github.com/bitcoin/bips/blob/master/bip-0340.mediawiki.

[15] Poon, J., and Dryja, T. The bitcoin lightning network: Scalable off-chain instant payments, 2016.

[16] Schrijvers, O., Bonneau, J., Boneh, D., and Roughgarden, T. Incentive compatibility of bitcoin mining pool reward functions. In *Financial Cryptography and Data Security* (Berlin, Heidelberg, 2017), J. Grossklags and B. Preneel, Eds., Springer Berlin Heidelberg, pp. 477–498.

[17] Sompolinsky, Y., Lewenberg, Y., and Zohar, A. Spectre: A fast and scalable cryptocurrency protocol. Cryptology ePrint Archive, Report 2016/1159, 2016. `https://eprint.iacr.org/2016/1159`.

[18] Vorick, D. Jute: More scalable, more decentralized proof-of-work consensus. `https://github.com/Taek42/jute`.