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 to remain decentralised. The primary problems with p2pool were twofold. First, the variance in earnings for miners didn't reduce with the increase in hashrate participating in p2pool. Secondly, making payouts to miners required a linearly increasing blockspace with the increase in the number of miners on p2pool. Building a DAG of miner's shares and the use of payment channels are two proposals trying to alleviate these problems faced by p2pool. In this document, we present a unified solution that uses a directed acyclic graph to track miners shares and uses payments channels to reward miners. The shares calculation can be carried out by any node on the p2p, and the rewards are paid out by a pool operator. Using the payment channels construction neither the pool operator nor the miners can cheat each other. We show that our approach is incentives compatible and reduces variance in earnings for miners. We also propose a solution for trading the miner's proof of work for BTC in an open market.

1 Motivation

P2Pool [1] helped decentralise bitcoin by enabling miners to select which transactions they mined and thus avoid any potential censorship by pool operators. However, the construction used by P2Pool faced a number of problems that lead to miners abandoning the pool.

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

The first two problems are a direct consequence of the shares block rate limited to 30 seconds. With only one block possible every thirty seconds, any increase of hashrate on P2Pool resulted in shares competing to be the next block in the p2pool chain.

There is a clear tension in play here, increasing the block rate frequency doesn't scale the throughput of the pool in terms of number of shares found, as most of them are orphaned and the miners not being rewarded for those orphans. Ethereum's inclusive protocols [7] help 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 [10].

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

1. Lower variance for miners to enable the long tail of independent miners.
2. Independent miners that build their own blocks.
3. Payouts for miners with constant size block space requirement.
4. Provide building blocks for a hash rate futures market.

2 Current Proposals

TerraHash Coin [10], Jute [20] and [19] are some of the attempts to use a DAG for faster block times. However, these works focus on changing the consensus layer of bitcoin itself. The ideas in these proposals allowed 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 block times and using this DAG for calculating distribution of payouts between miners. We then propose using payment channels as defined by Belcher [2] to avoid using block space for making payouts to miners.

Apart from the work for increasing block rates using DAGs, Belcher [2] proposes a different idea to help decentralise mining deals with the problem of a paying miners from a decentralised mining pool. P2Pool uses the coinbase transaction of a block to pay out miners. Belcher shows how a scheme can be constructed using payment channels between federated hubs to pay miners after a block has been successfully mined. The payouts can be paid after a long enough period, similar to the 100 blocks requirements for spending from coinbase transactions. Miners can register with hubs where bitcoin has been locked in to open payment channels to miners. The construction presented by Belcher 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 ideas of the Hash Rate Futures and Payment Channels for Rewards Payouts together present a potential path for rebooting P2Pool. In the rest of the document we present a slightly modified version of TerraHash Coin and show how the various components can work together.

3 Decentralised Bitcoin Mining

In this section we present a modified version of TerraHash Coin and show how it can use payments channels with hubs to deliver a decentralised mining pool for bitcoin.

3.1 A DAG of Shares

The braiding the blockchain proposal [10] 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 slightly different approach, where the blocks of a DAG represent shares of the mining pool, use payment channels between miners and a hub for distributing payouts in proportion to the work done by the miners.

Each miner builds their own block, selecting transactions as they want, we call this block the WORK. The description of WORK is then disseminated to the p2p network of miners using the compact block specifications [4].

The miner then starts mining on WORK and generates SHARE. 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 what the miner observes on the p2p network. This dynamic adjustment allows miners to adjust the rate at which they produce SHAREs. [TODO: Build a model to recommend an emission rate of these shares.]

Figure 1 shows the relationship between WORK and its SHAREs. Each WORK created by a miner has multiple SHAREs and they are both broadcast on the p2p network.

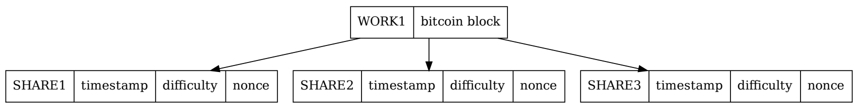

Figure 1: Each WORK generated and shared by a miner is then followed by the SHAREs the miner finds.

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 the block was mined, and all the blocks between l and r are in the same *epoch*.

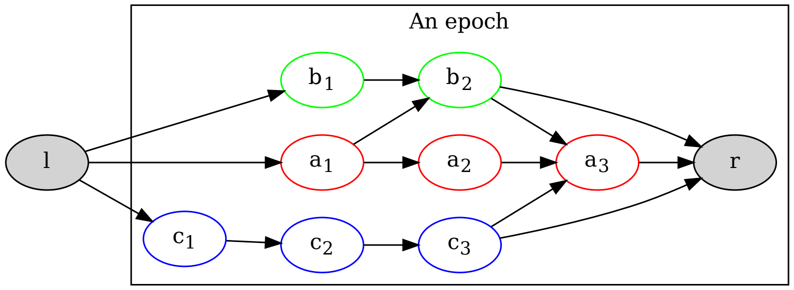

Figure 2: A epoch is defined as all the SHAREs mined between two bitcoin blocks. Here all the SHAREs between l and r are in the same *epoch*.

When a miner starts working on a SHARE it includes a reference to the highest known 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. This requires that miners is to the detriment of the other miners as we show in Section 3.2.

In the next section we then describe how all peers compute their fair share of profits using the DAG of shares. We show how our reward computation algorithm is incentives compatible [17].

3.2 Incentives Compatible Rewards

Each participating node, which includes the miners and the hub, is able to the see the DAG of SHAREs as broadcast on the network. Each SHARE includes a reference to the blocks 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, 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 and the reward is not shared with any other miner.

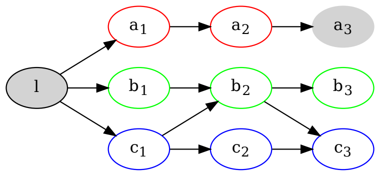

Figure 3: a discovers a share and the reward is not shared with any other miner.

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.

1. Traverse the DAG in reverse order from the SHARE that found the latest 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 difficulty of all SHAREs found by miners.

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 $\{a_i..a_5, b_1..b_3\}$. 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 $c4$ as it is doesn't include a reference to the last found bitcoin block a_5.

With the above rules, it should be easy to prove that the rule reward is an incentives compatible reward function as defined by [ref:incentives compatibility], and we present an outline of proofs that will be be formalised in future work.

Incentive Compatibility Given the rules above, 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 they find it.

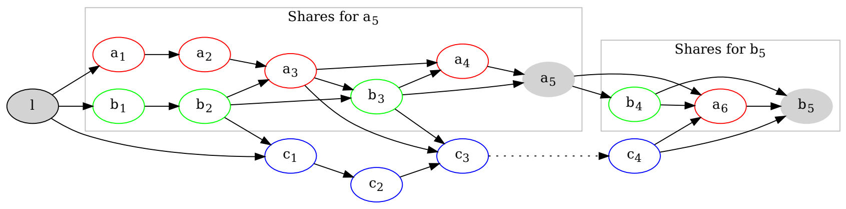

Figure 4: Two epochs in a DAG of shares mined by three mines — a, b and c. The shares in grey meet the bitcoin difficulty at the time they were mined.

Proportional Payments Since rewards are calculated at the end of an epoch, that is when all the valid shares found by all miners have been discovered, all miners are guaranteed payments for the shares that have reached the miner who found the block.

Budget Balanced Again, 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 [2]. The construction of the payments channels we present is similar to Belcher's construction, but there are a few changes and we highlight 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 using responder anonymity [15, 8, 16, 3] techniques developed for P2P networks.

3.3.1 Coinbase

We use Belcher [2] 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. The Hub with a hash lock for pre-image `X`, or
3. The Miner that found the bitcoin block, but after waiting for six months.

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

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

The scriptPubKey for the above conditions in the coinbase can be written as:

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 get paid for their accumulated payouts if the Hub defects.

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 mined. 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 2-way payment channels as an initial implementation for Braidpool. If required we can switch to 2-way channels [14] 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 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.

Fund Transaction

Input	**Output**

Hub's UTXO
(Signed by the hub)
	`2 H  M  2 CHECKMULTISIG`
	`OR`
	`2 H' M' 2 CHECKMULTISIG + Hash(X)`
	(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 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.

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 show 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 **??** shows the structure of the payment transactions.

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: $earnings$)

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

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, but for the purposes of discussion we assume there is a mechanism that allows a miner to be sure that the hub has updated the states of all payment channels.

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 `hash(X) + Hub P2WPKH` branch of the coinbase. In doing so, the hub has to reveal the pre-image `X`. With the pre-image available, all miners can use the `2 H' M' CHECKMULTISIG + Hash(X)` branch of their payment channel transactions and close their channels.

The hub could defect on the very first block mined and the miners would lose all their earnings. 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 would 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.

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 be 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. However, the miner also loses any payouts since the last block was found by the pool. A malicious miner could provide a large portion of the pool's hash rate and then refuse to sign the co-operative branches, this way it will make the pool defunct, but it will be an expensive attack. Probably only one the state can afford, and the only response to such an attack would be for the miners and the hub to go underground.

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 a stream of future profits for the hub.

3.5.4 DDoS on the Hub

The hub could be attacked using a distributed denial of service attack making 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 attack. The proposal also points out that multiple hubs will reduce the liquidity required to open channels with miners.

According to the Belcher proposal, with multiple hubs available on the p2p network, miners will open channels to all hubs and receive payouts from all the hubs. The coinbase is split between hubs in a way that if any hub defects, the other hubs can still spend the coinbase and split the block reward between them — paying the miners appropriately.

Creating a larger coinbase for multiple hubs scales if we can use Taproot [13, 11, 12] once it is activated. The solution uses staggered timeouts for hubs to spend the coinbase in case of defecting or hubs under DDoS attacks. We believe 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,

We propose a different solution that requires a single hub. The advantage is a simpler channel construction and the ability to release the pool without waiting for Taproot activation. The single hub handling payouts is made indistinguishable from other miners by using well known responder anonymity [15, 8, 16, 3] protocols for use in p2p networks.

3.6 P2P Network Protocol

The p2p protocol has to enable two types of communications where:

Shares broadcasts: Miners broadcast their shares to all other miners in the network.

Channel management: Channel creation and channel state update messages exchanged between the miners and the hub.

Even though the two message types seem to be broadcast and unicast communication, we propose using broadcast for both the communications.

In the first case, an epidemic or gossip based broadcast [5] algorithm like the one use in bitcoin and p2pool will meet our goals of making sure all miners receive all the updates to the DAG of SHAREs. There are a number of implementation of a gossip based protocols and we don't get into the details here.

The second case where the miners and the hub have to exchange messages seems to call for a one to one communication. However, we need to enable this communication without anyone able to find out which node is the hub. The hub has to be indistinguishable from other participants to prevent a DDoS attack on the hub. Our solution is to use gossip based broadcast of all communication between the hub and the miner using techniques to hide the source and destination of a message in a p2p network [15, 8, 16, 3, 9, 18]. Our solution makes a trade off between latency of Channel management messages and the hub's anonymity.

We use two different p2p networks for Shares broadcasts and for Channel management messages. The shares broadcasts network prioritises lower latency over anonymity, meanwhile the Channel management network sacrifices latency for hub's anonymity. Since the Channel management messages are far less frequent than Shares broadcast messages we think separating the networks and making the latency-anonymity trade-off is justified.

In this paper, we provide an overview of the approach we will take for the Channel management p2p network and the details will follow in a separate document. We next provide the high level approach we take for the channel management p2p network:

Epidemic Broadcast — All nodes forward all received with probability P_{fwd}, a system parameter.

Noise — All nodes periodically initiate a broadcast, the way a hub will do, so that no external observer can use traffic analysis to identify the hub.

Message Encryption — all messages are encrypted by the sender using the hub's public key. Hub encrypts the messages in response using the miner's public key included in the received message and the miner's SHAREs. The hub's public key is discovered out of band.

Pad Messages — All encrypted messages are padded to be the same size.

Anti-Entropy — Occasionally all nodes send a message to nodes they are connected to compare the list of channel update messages they have seen. If any messages are missing the initiating node uses *pull* based mechanism to request missing messages.

Use I2P — The channel management p2p network is built on top of the I2P [6] sublayer to further prevent any association between the miner's IP addresses and the source of Channel management messages. Recall that the SHAREs are broadcast on a separate faster p2p network using a different epidemic broadcast protocol.

The above high level approach will generate a lot of traffic on the channel management network and also results in high latency of message delivery. The seminal analysis presented in [5] points to around five messages forwarded by each node if all nodes forward a received message with 20% probability. Say we want all nodes to initiate one channel update message (dummy or genuine) every block interval, then we get 5000 message transmissions by each node every ten minutes given a 1000 node network.

There are proposed optimisations to the above described approach in literature [15, 8, 16, 3, 9, 18], however the focus of these optimisations is to enable anonymous transfer of large files. To reduce the bandwidth requirements they reduce the level of anonymity provided to the receiver of the requests. However, in our case we are not transferring large files, and we don't want to compromise of the level of anonymity afforded to the hub.

As a final note, in our initial implementation we will keep the hub publicly available on a well known IP address, so that all miners can directly communicate with the hub. As the pool grows, we plan to switch to the above described p2p network for channel management and hiding the hub in plain sight.

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.

4.1 Simulations

Before we work on implementing they system, our next step is to simulate p2p mining network using ns-3 [ref] 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.

4.2 Specifications

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.

4.3 Proofs

We want to use the model presented in [17] to provide proofs for how the rewards distribution is incentives compatible.

4.4 Multiple Hubs

We would like to build further on the multiple hubs construction described by Belcher once Taproot is activated on bitcoin.

References

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

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

[3] BENNETT, K., AND GROTHOFF, C. GAP - practical anonymous networking. vol. 2760, pp. 141–160.

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

[5] Demers, A., Greene, D., Houser, C., Irish, W., Larson, J., Shenker, S., Sturgis, H., Swinehart, D., and Terry, D. Epidemic algorithms for replicated database maintenance. *SIGOPS Oper. Syst. Rev. 22*, 1 (Jan. 1988), 8–32.

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

[7] 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.

[8] Liu, Y., Han, J., and Wang, J. Rumor riding: anonymizing unstructured peer-to-peer systems. *IEEE Transactions on Parallel and Distributed Systems 22*, 3 (2010), 464–475.

[9] Liu, Y., Han, J., and Wang, J. Rumor riding: Anonymizing unstructured peer-to-peer systems. *IEEE Transactions on Parallel and Distributed Systems 22*, 3 (2011), 464–475.

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

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

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

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

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

[15] Scarlata, V., Levine, B., and Shields, C. Responder anonymity and anonymous peer-to-peer file sharing.

[16] Scarlata, V., Levine, B., and Shields, C. Responder anonymity and anonymous peer-to-peer file sharing.

[17] 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.

[18] Sherwood, R., Bhattacharjee, B., and Srinivasan, A. P5: A protocol for scalable anonymous communication. *Journal of Computer Security 13*, 6 (2005), 839–876.

[19] 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`.

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