Tortoise and Hares Consensus: the Meshcash Framework for Incentive-Compatible, Scalable Cryptocurrencies

Iddo Bentov   Pavel Hubáček   Tal Moran   Asaf Nadler

Abstract

We propose Meshcash, a new framework for cryptocurrency protocols that combines a novel, proof-of-work based, permissionless byzantine consensus protocol (the tortoise) that guarantees eventual consensus and irreversibility, with a possibly-faulty but quick consensus protocol (the hare). The construction is modular, allowing any suitable "hare" protocol to be plugged in. The combined protocol enjoys best of both worlds properties: consensus is quick if the hare protocol succeeds, but guaranteed even if it is faulty. Unlike most existing proof-of-work based consensus protocols, our tortoise protocol does not rely on leader-election (e.g., the single miner who managed to extend the longest chain). Rather, we use ideas from asynchronous byzantine agreement protocols to gradually converge to a consensus.

Meshcash, is designed to be *race-free*: there is no "race" to generate the next block, hence honestly-generated blocks are always rewarded. This property, which we define formally as a game-theoretic notion, turns out to be useful in analyzing rational miners' behavior: we prove (using a generalization of the blockchain mining games of Kiayias et al.) that race-free blockchain protocols are incentive-compatible and satisfy linearity of rewards (i.e., a party receives rewards proportional to its computational power).

Because Meshcash can tolerate a high block rate regardless of network propagation delays (which will only affect latency), it allows us to lower both the variance and the expected time between blocks for honest miners; together with linearity of rewards, this makes pooled mining far less attractive. Moreover, race-free protocols scale more easily (in terms of transaction rate). This is because the race-free property implies that the network propagation delays are not a factor in terms of rewards, which removes the main impediment to accommodating a larger volume of transactions.

We formally prove that all of our guarantees hold in the asynchronous communication model of Pass, Seeman and shelat, and against a constant fraction of byzantine (malicious) miners; not just rational ones.

Contents

1 Introduction

For a currency to be effective, it should satisfy several conditions:

- Limited Supply: The supply of new coins should be limited.
- No double spending: The total amount of money expended by a party cannot be more than the amount received by that party. That is, Alice should not be able to pay Bob and Charlie with the same coin.
- Liveness: If Alice wishes to pay Bob and has enough funds to do so, then she will be able to.
- Consensus on history: All the participants must have an agreed upon history of the transactions in the currency. That is, if Charlie believes that Alice paid Bob, then Dana and Eve should not believe a contradicting claim.
- Irreversibility: Once a transaction is in the agreed history, it should stay there. That is, if everyone agrees that Alice paid Bob today, they should also believe this tomorrow.

These requirements apply to all currencies, but are particularly problematic for digital currencies. The "traditional" solution (e.g., as employed by the financial industry) is to rely on a central authority in order to guarantee these properties (e.g., a government or a bank). The main innovation of Bitcoin [31, 12, 32] is a method to *distribute* the trust requirements; instead of having to trust entirely in a single authority, the underlying assumptions are about larger groups. In Bitcoin, the assumption is that the majority of the active participants behaves honestly, where majority is measured by computational power.

As described by Nakamoto [31], the core mechanism for achieving the above desiderata is by constructing a *distributed timestamping server.* This allows all participants to agree on the transaction history, and in particular on the order in which transactions occurred (ensuring that if a party tries to spend the same coin twice, everyone will agree which of the two transactions is valid). Loosely speaking, participants "vote" for the history they consider valid, and the majority "wins". In an ideal, democratic cryptocurrency, every participant would be allowed the same number of votes. However, this seems impossible to enforce in the current Internet; since there is no mechanism for identity verification, a malicious party can create many fake identities. Instead of using a "one-person-one-vote" rule, Bitcoin enforces a "one-resource-one-vote" rule, with the resource in question being computational work. That is, participants generate Proofs of Work, with each proof corresponding to a "vote". To reduce communication, Bitcoin actually uses a lottery system, where each Proof of Work (PoW) corresponds to a lottery ticket, and only the winning ticket gets to vote.

The decentralized nature of the Bitcoin protocol stems from the fact that it allows anyone who contributes computational power to participate. In this sense, Bitcoin is regarded to be a *permissionless* consensus protocol (cf. [3]).

Indeed, the Bitcoin network has been running without major setbacks since its launch in January 2009. Yet, a significant volume of academic work has been done to analyze future obstacles that Bitcoin may face: centralization of the mining power in the hands of few large data centers [28, 9], scalability barrier w.r.t. high volume of commerce [7, 40, 20] or selfish mining [11, 38].

1.1 Our Contributions

In this work, we describe our novel *Meshcash* framework, which aims to either solve or mitigate the aforementioned risks that Bitcoin faces. The fundamental idea behind Meshcash is a novel consensus protocol that is not based on leader-election. Instead of a race to be chosen as the "leader" of the next round, in which only one party can be the winner (e.g., generate the next block in Bitcoin), we use ideas from the byzantine agreement literature to achieve a consensus on all generated blocks. In particular, this means that one miner's success does not prevent the success of another.

The construction is modular, combining our long-term "tortoise" consensus protocol (that guarantees eventual consensus and irreversibility very robustly) with a short-term "hare" protocol that can achieve consensus quickly but is not robust (or irreversible). This allows us to get the best of both worlds: fast, irreversible consensus if the hare protocol succeeds, and long-term irreversible consensus under more extreme conditions.

Unlike most alternative cryptocurrency protocols, we prove our security guarantees with regards to *malicious* adversaries and in an asynchronous communication model. This means that our protocol is robust even against non-rational adversaries (as long as they do not have too large a fraction of the computation power). At the same time, we can still show that the protocol is incentive-compatible (under some simplifying assumptions).

Our protocol replaces the single chain of blocks (in which only one block can be next) with a *mesh*—a layered directed acyclic graph (DAG) which allows multiple blocks to coexist in parallel, while the rewards are still shared proportionally to the work performed. This offers mitigating factors for the risks that Bitcoin faces:

- **Greatly reduced incentives for pool mining.** This risk stems from the simple fact that the expected time and variance of solving blocks is too high for a hobbyist miner. For example, if there are $100,000$ miners with equal hashrate, and the Bitcoin difficulty dictates it takes 10 minutes on average to solve a block, then each miner will need to wait for 1 million minutes (slightly less than 2 years) on average before solving a block. This would obviously be unacceptable from the point of view of the individual miners, as they have running expenses and their mining equipment may fail before they are ever rewarded. Therefore, Bitcoin miners have a strong incentive to combine their resources into centralized pools. This is unhealthy for decentralization, because pools tend to increase in size over time. As a remedy against the centralization pressure, many more blocks would get created per unit of time in Meshcash (e.g., we can easily support 200 blocks in every 10-minute period), and hence solo-mining or participating in small pools is more feasible compared to Bitcoin.

- **Improved scalability.** One of the main barriers to scalability is the effect of larger block sizes on the network propagation delay. By removing the "race" aspect of mining, the propagation delay becomes much less relevant, allowing the system to support larger block sizes. (See Appendix A for more details.)

- **Incentive-compatible verification.** When a Bitcoin miner verifies and includes certain transactions in a block that she creates, she collects the transaction fees as her reward. Other miners should also verify those transactions and thereby ensure that the chain that they try to extend is valid, even though they do not collect any rewards for those transactions. Thus, rational miners can do a cost-benefit analysis, and may decide to skip the verification of transactions in prior blocks [23]. Indeed, this behavior appears to be widespread among Bitcoin miners, as some miners lost a significant amount of funds due to the BIP66 softfork [25]. In Meshcash this risk is mitigated because miners do not engage in tight races against one another, therefore they have plenty of time to verify the transactions that reside in the blocks that they endorse. Thus, it is less risky to have transactions with complex scripts in Meshcash relative to blockchain protocols, though a quantification of this claim requires analysis that we do not provide in this work.

- **Incentive-compatible propagation.** A rational Bitcoin miner may decline to retransmit transactions that were sent to her, thereby increasing the likelihood that she will collect more fees when she eventually solves a block [2]. Such a behavior damages the performance of the Bitcoin system from the point of view of its users, as transactions would become confirmed at a slower pace overall. In Meshcash, the transaction fees are divided among all miners who created blocks in the recent layers, and hence an individual miner does not gain by keeping transactions secret.

- **Resistance to bribe attacks.** In Bitcoin, rational and malicious parties may benefit from offering bribes to other miners, by sending in-band messages in an anonymous fashion [4, 21]. A rational miner may fork a high-value block by collecting only some of its transactions, thereby enabling the next miners to pick up the rest of the transactions and earn extra fees. A malicious adversary may even put a "poisonous" transaction tx_0 in the honest chain and then offer high fees for blocks that include another transaction that conflicts with tx_0, and thus incentivize rational miners to work on a fork. Similarly to the rational miner, this adversary may also mine blocks that do not include all the transactions that honest miners collected in their chain, and in effect incentivize rational miners to work on her fork (without offering overt bribes). In Meshcash, the fees are shared among all the blocks of a layer, and conflicting transactions do not invalidate blocks that reference them (cf. Section 3.4.4), hence these kinds of bribe strategies are ineffective.
- **Resistance to forking.** An important property of our protocol (and one that, to the best of our knowledge, is not satisfied by any other cryptocurrency) is that forking the mesh is hard even for an attacker with a constant fraction of the computational power. This makes it much easier to argue about rational behavior—honest miners know that with high probability their work will not go to waste. In particular, it makes the standard selfish-mining attacks moot.

Informally, our Meshcash protocol achieves the following guarantees.

Theorem 1.1 (informal)**.** *If the hare protocol achieves fast consensus in the presence of adversaries controlling $q < 1/3$ fraction of the computational power then the combined tortoise/hare protocol achieves consensus and irreversibility against the same adversary.*

For the formal statement see Corollary 4.18.

Theorem 1.2 (informal)**.** *When the adversary controls $q < \frac{1}{15}$ fraction of the computational power, then the tortoise protocol achieves consensus and irreversibility irrespective of the initial conditions and the properties of the hare protocol.*

For the formal statement see Theorem 4.12.

Note that because the tortoise protocol guarantees consensus *irrespective of the initial conditions*, it is extremely robust to temporary violations of our security assumptions. For example, if the adversary normally has low computational power, but might be able to generate short "spikes", the tortoise protocol guarantees that the system will recover. Moreover, we expect the security of the protocol in practice to be much better than our worst-case analysis shows—our analysis is optimized for readability and asymptotic results rather than reducing the constants.

2 Related Works

The idea of replacing the blockchain with a DAG is not new; to the best of our knowledge the earliest consideration of it was in [24], though the treatment there was rather abstract.

2.1 Leader-Election-Based Protocols

Protocols based on leader election have an inherent asymmetry: loosely speaking, consensus is achieved by selecting some "special" party (the leader) in each round of the protocol. In Bitcoin and several other permissionless protocols, the leader is the first party that successfully to successfully solve a proof-of-work. Due to the possibility that more than one party will be special in a round, these protocols all imply some sort of "race". We note that this is a property of the consensus protocol, not the reward mechanism; thus, in theory, a leader-election-based protocol can still be completely race-free. We classify the protocols into those in which the chosen leader receives all of the reward (these have an inherent race), and those in which leaders have less power and the rewards are shared more equally between the parties.

2.1.1 Winner-takes-all

GHOST The GHOST protocol of Zohar and Sompolinsky [40] modifies the Bitcoin best-chain rule to take into account valid blocks that are not on the longest chain. This means that less honest mining power goes to waste, and therefore GHOST can support a shorter interval between blocks without sacrificing security (cf. [16] for a formal analysis). The main motivation behind GHOST is to increase the transaction throughput. While this is also a secondary motivation for Meshcash, our main design goal is to make the cryptocurreny race-free, so that incentive-compatibility is easier to achieve (and to prove). These concerns are not addressed in the GHOST protocol (e.g., GHOST is vulnerable to selfish mining attacks). With regard to frequent payouts to small miners, GHOST improves upon Bitcoin since it allows an increase in the block generation rate, but the growth rate of the main chain (i.e., the payout rate) is still bounded as a function of the network propagation time. In fact, the main chain in GHOST grows at a slower pace than what the propagation time would imply, due to the commonplace orphaned blocks. For example, if blocks are generated every 15 seconds on average and network propagation time is 20 seconds, then the effective interval between blocks in the main chain in GHOST is at least 55 seconds (this bound is implied by the tightness of [40, Lemma 6]). That is, the payout rate is only about 10 times faster than Bitcoin. In Meshcash, on the other hand, we can easily support (say) 200 times the Bitcoin payout rate by using a layer width of 200. It should also be noted that GHOST exhibits the nonlinearity of rewards phenomenon, implying that a fast chain growth is likely to result in centralization instead of decentralization(see Appendix A).

Bitcoin-NG The Bitcoin-NG protocol of Eyal et al. [10] seeks to improve upon Bitcoin by offering faster confirmation time and better throughput. The incentive-compatibility aspects of the Bitcoin-NG protocol are not analyzed in [10], and Bitcoin-NG may indeed suffer from the same kind of problems that Bitcoin can potentially have. E.g., when a rational Bitcoin-NG miner sees that the microblocks before the most recent leader L have collected a relatively high amount of fees, she may opt to fork L by extending one of the intermediate microblocks that preceded L, and thus offer more fees so that the next leader will prefer her fork. Moreover, due to the chain structure, the scalability improvements of Bitcoin-NG require sending microblocks of a smaller size as the transaction volume increases, otherwise the competing miners will be discouraged from collecting too many transactions. This becomes impractical for microblock sizes that are too small (the reported experiments in Bitcoin-NG were done with a transaction volume that is at most 5 times greater than that of Bitcoin).

Blockchain as Bootstrap for Byzantine Agreement Hybrid Consensus [34], Solidus [1], SCP [22], and Byzcoin [17] are protocols that harness the consensus guarantees of a PoW based blockchain in order to select a committee of parties who would then use a Byzantine agreement protocol (PBFT [6]) to achieve faster transaction irreversibility and higher throughput, in the permissionless setting. Thus, the objective of these protocols is a mirror image of our hare component in Section 3, i.e., we use an optional off-chain ABA protocol for a single bit (much simpler than PBFT) in order to help the PoW based mesh achieve consensus in the first place. These four protocols guarantee security only if 2/3 of every committee remains honest for at least τ time. If this new assumption is violated, the committee can reverse history and create unresolvable conflicts for a long period. Moreover, there is no definite cost for the adversary in attempting a collusion attack, since a failure would not involve depletion of physical scarce resources. In comparison, Meshcash relies only on a standard hashpower majority assumption for security and is more robust against temporary adversarial "power spikes", as even a completely corrupt ABA committee can—at worst—slow down consensus.

2.2 Reward-sharing

Inclusive Blockchains The work of Lewenberg, Sompolinsky and Zohar [20] presents a cryptocurrency protocol that is based upon a DAG instead of a chain, but unlike Meshcash it does not provide a mechanism for cooperative sharing of the block rewards. In fact, [20] explicitly states that it does not mitigate selfish mining attacks. Furthermore, [20] is still, in some sense, a blockchain protocol; there is always a main chain, but parties will incorporate non-conflicting transactions from off-chain blocks that do not appear on the main chain.

DagCoin and Braidcoin DagCoin [18] is another DAG-based proposal that focuses on possible improvements over Bitcoin in terms of faster security confirmations and greater throughput. Braidcoin [26] is a more recent DAG-based cryptocurrency construction, that aims to support more frequent blocks and thus faster payouts to small miners. However, neither [18] nor [26] give a security analysis of their protocols.

FruitChain The FruitChain protocol of Pass and Shi [33] is a blockchain-based scheme in which miners who create blocks are not rewarded, but the blocks incorporate "fruits" and miners earn rewards for the fruits that they create. Similarly to Meshcash, FruitChain attains a protocol that is ϵ-incentive-compatible, by employing a scheme that shares the rewards among those who contributed to the recent ledger history. However, the analysis of [33] sidesteps the potential freeloading risk by using a model in which verification complexity has a negligible cost, and assuming that an honest majority follows protocol without having a clear rationale to do so. Additionally, [33] does not claim to improve upon the scalability aspects of Bitcoin, and FruitChain may potentially inherit the throughput limitations that chain-based protocols have.

2.3 Leaderless Protocols

Blockchain-free Ledger A recent work by Boyen, Carr and Haines [5] presents a cooperative DAG-based cryptocurrency protocol, though it only considers security against a rational adversary. While it is not based in leader-election, this protocol is not race-free—conflicts between honest parties may occur due to network delays, in which case some of the honest parties will not receive a reward.

SPECTRE The SPECTRE protocol of Sompolinsky, Lewenberg, and Zohar [39] is a DAG-based construction, and hence it bears a resemblance to Meshcash in the way that each block commits to a list of earlier blocks. However, SPECTRE only guarantees consensus on blocks that were honestly generated, and makes no guarantees *at all* for maliciously-generated blocks. This relaxed guarantee is enough to construct a crypto-currency supporting basic monetary transactions, but is insufficient for more advanced uses (such as complex scripts), as they may require consensus on a total ordering of the blocks in order to evaluate even honest transactions.

Meshcash, on the other hand, supports and in fact mitigates the potential risks that are associated with complex scripts, due to its race-freeness. A major design goal of SPECTRE is fast transaction irreversibility. While transactions are confirmed faster in Meshcash than in Bitcoin, this is not an aspect that Meshcash excels in. Another main goal that SPECTRE seeks to accomplish is high scalability. However, the SPECTRE protocol is not of a cooperative nature, and is therefore more likely to exhibit the nonlinear rewards phenomenon when the transaction volume increases.

3 Protocol Description

3.1 Informal Overview

Meshcash is a modular protocol combining a fixed "tortoise" protocol to guarantee long-term consensus and irreversibility with an interchangeable "hare" protocol that can aid in getting quick consensus. In terms of execution, miners in the combined Meshcash protocol behave similarly to Bitcoin. As in Bitcoin, each Meshcash miner performs PoW computations to generate blocks. Abstractly, the main distinction between Bitcoin and Meshcash is the structure of the "blockchain". In Bitcoin, each block points to one previous block, forming a chain. The consensus rule is that the longest (heaviest) chain is the "correct" one. In Meshcash, the structure is instead a layered DAG; each block belongs to a layer (it contains a $layer_{id}$ field that declares the layer number) and points blocks previous layers. The consensus rule is slightly more complex—for blocks that are in the recent past, Meshcash delegates consensus to the hare protocol (i.e., deciding which blocks are part of the "correct" history depends on the hare protocol rules). For blocks in the far past, we use the tortoise protocol rule, which takes a weighted "vote" of all subsequent blocks to decide whether or not a block is part of the canonical history.

The mining protocol is actually much simpler, and avoids much of the consensus rule complexity: essentially, miners generate blocks pointing to every valid block they see in the previous layers (the number of previous layers to include depends on the hare protocol). A detailed description of the protocol is specified in Section 3.4.

3.2 Modeling Generic BlockDAG Protocols

We generalize the model of Pass, Seeman and shelat [32] from block*chains* to block*DAGs* (this can also be seen as a generalization of the Kiayas and Panagiotakos model [16] from trees to DAGs, but using the asynchronous communication model of Pass et al.)

In the spirit of [32], we define a blockDAG protocol to be an interactive protocol where each participant has a local state that contains a list of messages received (and the time at which each message was received). Based on this state, each participant constructs a graph of *blocks* with directed edges between them. In a blockDAG protocol, the graph is an arbitrary DAG (in [32] only chains are permissible, while [16] also allows trees).

For timing purposes, we identify each block with one of the messages in the state. Each block is uniquely labeled by (a collision-resistant hash of) its contents: these include the labels of all blocks to which it points, as well as an arbitrary "content string" (which is interpreted in a protocol-specific way; e.g., it can contain a list of transactions).

3.2.1 Validity.

A blockDAG protocol (like a blockchain protocol) defines a "validation function" for blocks; given a state, the validation function labels every block in the graph as either valid or invalid. We separate the validation function into two types of validity (both are required for a block to be considered valid):

- Syntactic Validity: this can be computed based only on contents of block and its view (the subset of the DAG that is reachable from that block).

- Contextual Validity: everything that is not syntactic.

For example, the hash proof validity in Bitcoin is syntactic, while the chain selection rule is contextual (it depends on whether a longer chain that does not include the block exists in the state, something that cannot be determined from the block's view by itself).

A blockDAG protocol's validation function may allow valid blocks to have invalid blocks in their view (this is not allowed in existing blockchain protocols). We define the *valid DAG* to be the DAG restricted to valid blocks (note that this subgraph may not be connected).

3.2.2 Total Order on Blocks.

Unlike a chain, a DAG does not guarantee a unique topological ordering of its blocks. Thus, the "T last blocks" in the DAG may not be well-defined. To generalize the Pass et al. definitions, we will require the blockDAG protocol additionally to define a unique total order on its blocks that is consistent with the topological ordering of the DAG (that is, for every pair of blocks A and B such that $A < B$, there does not exist a directed path in the DAG from A to B).

3.2.3 Security Properties.

Our basic security properties are exactly as in [32]:

- *consistency*: with overwhelming probability (in T), at any point, the valid DAGs of two honest players can differ only in the last T blocks;

- *future self-consistence*: with overwhelming probability (in T), at any two points in time $r < s$ the valid DAGs of any honest user at r and s differ only in the last T blocks (as they appear at time r);

- *g-chain-growth*: with overwhelming probability (in T), at any point in the execution, the valid DAG of honest players grew by at least T blocks in the last g rounds, where g is called the chain-growth of the protocol;

- *μ-chain quality*: with overwhelming probability (in T), for any T consecutive blocks in any valid DAG held by some honest player, the fraction of blocks that were "contributed by honest players" is at least μ.

3.3 Weak Common Coins

Our protocols rely on a *weak common coin* as a black box (this is used both in the ABA protocol of Moustfaoui et al. [29] and to guarantee eventual consensus of the tortoise protocol when the hare protocol does not achieve consensus). A weak common coin with parameter $p_{\text{coin}} \leq \frac{1}{2}$ is a protocol with the following guarantees (probabilities are over the coins of the honest parties and the adversary):

1. The probability that all honest parties have the same output and it is 0 is at least p_{coin}.

2. The probability that all honest parties have the same output and it is 1 is at least p_{coin}.

3. Before the beginning of the protocol, for every honest player P, the adversary cannot guess the output of P with probability more than $1 - p_{\text{coin}}$.

Note that this definition does not prevent the adversary from completely controlling the output of all honest parties with probability $1 - 2p_{\text{coin}}$. Moreover, the results of the coin toss are not required to be verifiable (i.e., an honest party might not be able to prove to another party that its output was correct), and honest parties might not know whether they are in consensus on their output or not.

3.3.1 Implementing a Weak Common Coin Based on Proof-of-Work.

We can implement a weak common coin based on any proof-of-work protocol (assuming the PoW hashes are the output of a random oracle) in which block publication can be modeled as a Poisson process with rate proportional to computational power. Assume blocks are generated as a Poisson process with rate λ for an interval of length T, and that the network propagation delay δ is defined as in Section 3.4.4. Our protocol is as follows, for party P starting at time t:

1. Wait until time $t + T$. Denote S_P the set of *fresh*[1] syntactically valid blocks received in the interval $[t, t + T]$.

2. Sort the blocks in S_P by their hash values. P outputs the LSB of the minimum block in S_P (according to the sort order).

Lemma 3.1 gives the resulting parameters for the weak coin.

Lemma 3.1. *If the rate of blocks generated by the adversary is at most $q\lambda$, then for every constant $c > 1$ our protocol generates a weak common coin with parameter $p_{coin} \geq c^{-2} \cdot \frac{1-q}{2} \cdot \frac{T-2\delta}{T+2\delta} - \boldsymbol{neg}(\lambda)$.*

Proof. Denote S_P^- the set of blocks received by P in the interval $[t + \delta, t + T - \delta]$ and S_P^+ the set of blocks received by P in the interval $[t - \delta, t + T + \delta]$.

By our communication assumption, honest nodes will agree on the time at which they received every message up to a difference of δ. Thus, for every pair of honest parties $P.P'$, we must have $S_P^- \subseteq S_{P'}$ and $S_P \subseteq S_{P'}^+$.

This means that if A is the minimum block of S_P^+ and $A \in S_P^-$, then for every honest P' it must be the minimum block for $S_{P'}$ (if $S_{P'}$ contains a lower-weight block, then it would also be in S_P^+, contradicting the minimality of A). Let $S_P^* \subseteq S_P^-$ be the subset of blocks that were *honestly generated.*

If the minimum block of S_P^+ is in S_P^*, then the LSB (and hence the coin) is uniformly random, even conditioned on the set of published blocks (formally, we assume the hashes are the output of a random oracle; we can ignore the LSB in the sort order to make it fully independent). Moreover, since $S_P^* \subseteq S_P^-$, all honest parties agree on the coin. Let $2p_{\text{coin}}$ be the probability that this occurs; then the protocol is a weak common coin with parameter p_{coin}.

To lower-bound p_{coin}, note that since the hashes are the output of a random oracle, we can think of the sorting as a random shuffle of all the blocks in S_P^+. So $2p_{\text{coin}} \geq \mathbb{E}\left[|S_P^*|/|S_P^+|\right]$.[2]

By our block rate assumption $|S_P^+|$ has a Poisson distribution with parameter $\lambda^+ = \lambda \cdot \frac{T+2\delta}{T}$, while $|S_P^*|$ is Poisson with parameter $\lambda^* = \lambda \cdot (1 - q) \cdot \frac{T-2\delta}{T}$. The ratio of the two parameters is

$$\frac{\lambda \cdot (1-q) \cdot \frac{T-2\delta}{T}}{\lambda \cdot \frac{T+2\delta}{T}} = (1-q) \cdot \frac{T - 2\delta}{T + 2\delta}$$

[1] To ensure blocks are not pre-generated by the adversary, we require blocks to include a "freshness indicator". This can be, for example, pointers to all blocks received before time t, or a special transaction that is generated just before time t.

[2] formally, we condition on $|S_P^+| > 0$, but this event occurs with overwhelming probability so for clarity we omit it.

For all $c > 1$, let $p_{c-}(\lambda) = \Pr_{X \sim \mathrm{Pois}(\lambda)}[X \leq \lambda/c]$ and $p_{c+}(\lambda) = \Pr_{X \sim \mathrm{Pois}(\lambda)}[X \geq c \cdot \lambda]$. Note that for every constant c, both $p_{c-}(\lambda)$ and $p_{c+}(\lambda)$ are negligible in λ. Then

$$\begin{aligned}
\mathbb{E}\left[|S_P^*|/|S_P^+|\right] &\geq \Pr\left[|S_P^*|/|S_P^+| \geq c^{-2}\frac{\lambda^*}{\lambda^+}\right] c^{-2}(1-q) \cdot \frac{T-2\delta}{T+2\delta} \\
&\geq \left(1 - p_{c-}(\lambda^*) - p_{c+}(\lambda^+)\right) c^{-2}(1-q) \cdot \frac{T-2\delta}{T+2\delta} \\
&\geq c^{-2}(1-q) \cdot \frac{T-2\delta}{T+2\delta} - \mathbf{neg}(\lambda) \ .
\end{aligned}$$

□

3.4 Modular (Tortoise) Protocol

3.4.1 Overview.

Our high-level construction combines a “tortoise” protocol (that we describe below) with an arbitrary “hare” protocol. The long-term “tortoise” protocol guarantees consistency, regardless of the hare protocol’s properties. For every block X on whose validity honest parties agree at the end of the hare protocol, the tortoise protocol will ensure that all honest parties will continue to agree (and will not change their opinion in the future).

1. If the hare protocol is race-free (i.e., honest blocks are always accepted as valid), the composed protocol will also be race-free.

2. If the hare protocol guarantees agreement on *all* blocks with probability p, the composed protocol will guarantee irreversible consensus by the end of the hare protocol with probability $p - \varepsilon$ (for some negligible ε).

Our tortoise protocol has the property that the opinion of a node about the validity of any block in its view depends only on the view itself (and not on additional private context). Thus, given the node’s view, we can “simulate” its computation and reconstruct its opinion about any block. To make this possible, when mining on a block, the block’s content will include “view” edges to every other block in the node’s view that cannot be reached by existing edges (i.e., edges to the “heads” of the blockDAG).

Layers. The DAG defined by our tortoise protocol uses the notion of *layers* to partition blocks into sets. Layers are sequentially numbered, and we require each block to include its corresponding layer number as part of its content (i.e., this number is included in the PoW that gives the block its “name”). We also require honest parties to maintain weak consensus about a *layer counter*, signifying the current layer number. By *weak* consensus, we mean that if any honest party increments its own layer counter at time t (according to its own local clock), every other honest party will increment its counter to the same value by time $t + \delta$ (according to the first party’s local clock).

Note that the event that triggers the layer counter increment can be simply the passage of time, but it can also depend on each party’s view (since we assume δ-bounded delays in the network, this will still guarantee weak consensus). In our case, the layer counter will increment whenever the node can see sufficiently many syntactically valid blocks in the current layer.

Block Weights. For the purpose of computing the “weight” of a set of blocks, we sum up the weights of each block in the set. The weight of a block is the expected amount of work (e.g., number of hash evaluations) required to generate a block of the same difficulty. That is, if the threshold for block generation is p for layer i, the weight of all blocks in layer i is exactly $1/p$.

3.4.2 Hare Protocol Requirements.

For simplicity, we will assume the hare protocol uses the same notion of layers and layer counter (this is indeed the case for the protocols we propose in this paper). We have two main requirements from the hare protocol:

- Preventing pre-generation of blocks. To guarantee convergence, we need a bound on the number of syntactically valid blocks the adversary can "pre-generate" (i.e., blocks whose layer number is in the future). The tortoise protocol inherits its syntactic-validity rules from the hare protocol, so these rules must ensure the bound. For simplicity, in this paper we assume all hare protocols require a *minimum out-degree* rule: blocks in layer i are syntactically valid only if they have at least $T_{\min}$ syntactically valid layer $i-1$ blocks in their view.

- Limited $[t, s]$-consistency: ideally, we want that for every block X claiming to be in layer i, all honest nodes whose layer counter is in the interval $[i+t, i+s]$ are in consensus about the validity of X, and this consensus does not change in the interval $[\texttt{start}_{i+t}, \texttt{start}_{i+s+1})$.

3.4.3 Tortoise Protocol Description.

Let Π be a qualifying hare protocol with output interval $[t, s]$. To construct the corresponding tortoise protocol, we modify the validation function and graph construction for Π. Let i be the current value of the layer counter:

- *Add* view *and* voting *edges.* In addition to the edges specified by Π, every block generated by an honest player with DAG G will now include special edges:

  - *view edges*: an edge to every "head" block in G (i.e., blocks that have in-degree 0). These edges ensure that the view of a block generated by an honest party includes the entire view of that party.
  - *voting edges*: an edge to every *valid* block in layer $i-s$ through $i-t$, where validity is computed using Π's validation function (note that if validity can be inferred from the DAG itself, then explicit voting edges are not necessary).

  Π will ignore the extra edges.

- *Add coin bits.* Every block adds a coin to its header. The value of the coin is determined by a weak common coin protocol (cf. Section 3.3) that starts at time $\texttt{start}_i + \delta$. The Coin bit is set to 1 if the coin was 1 and 0 otherwise;

- *Add "before coin" bit.* This bit indicates whether the block was generated before the coin protocol ended (this is used to "abstain" from voting in cases where the coin bit matters).

- *Add "early block" bit.* Every block adds a additional bit to its header to specify whether the block was generated within less than δ time after the start of the layer. (this is used to abstain from voting in cases where the late blocks from the previous layer might make a difference.)

- *"Block Voting".* For every block A in layer $i' < i-s$, the validity of A will be determined by a simple "election". Every syntactically valid block Y claiming to be in layers $i'+1, \ldots, i-1$ is given a "vote", in the set $\{-1, 0, 1\}$, as long as Y was received by time $\texttt{start}_i + \delta$. Block A will be considered valid if the weighted sum of the votes for A is positive, where we use the block weights derived from their PoW. The algorithm to determine how a block X votes is recursive. Consider a block X in layer $j > i'$. The basic voting rules depend on the distance (in layers) between X and A:

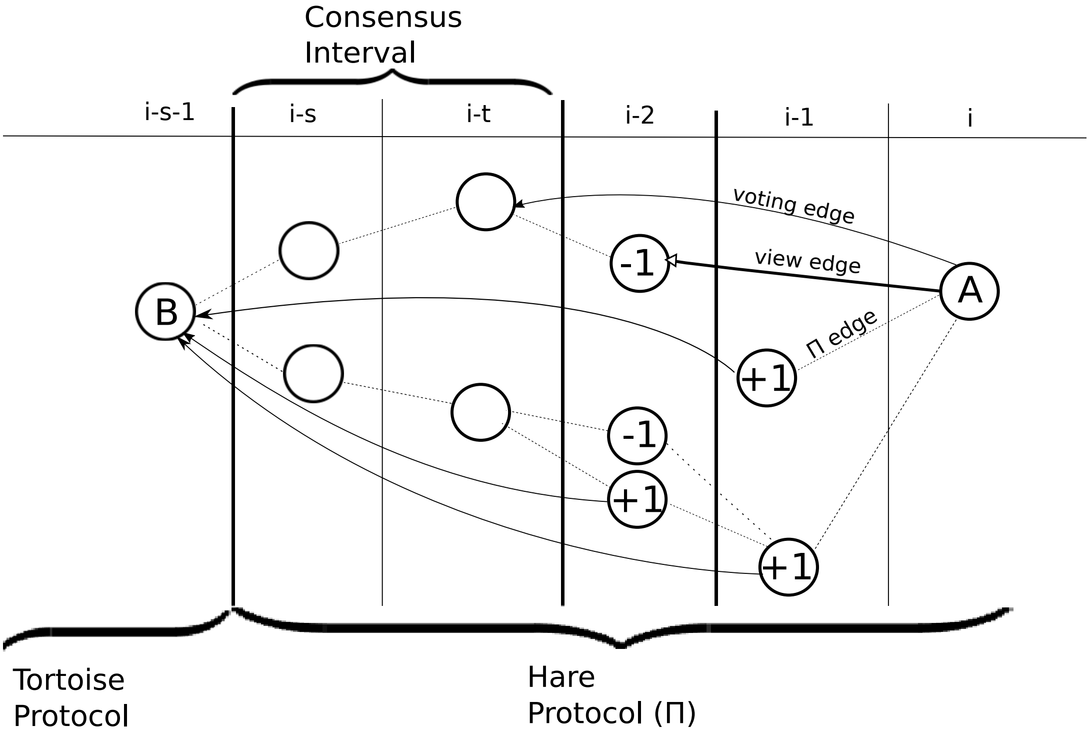

Figure 3.1: In *block voting*, the validity of any block is determined by election. In the figure, block A considers block B as valid because the sum of block votes in its view in favor of B is positive

- If $j < i' + t$, then X votes 0 (it is neutral with regards to A since it was generated before the hare protocol guaranteed consensus about A).
- If $j \in \{i' + t, \dots, i' + s\}$, then X votes 1 if X has a voting edge to A and -1 if it does not.
- Otherwise ($j > i' + s$), consider all the blocks in X's view and sum their weighted votes; X votes 1 if the sum is positive, -1 otherwise.

To ensure convergence even if the hare protocol fails to achieve consensus between honest nodes, we also add a randomized voting rule when there is not a clear majority in either direction:

- If $j > i' + s$ and the weighted sum of votes for the blocks in X's view is in the range $[-\theta, \theta]$, X votes 1 if the coin bit is set to 1 and -1 otherwise (unless "before coin" bit is set to 1 in which case it votes 0).

3.4.4 Protocol Parameters.

The Meshcash protocol has several tunable parameters which are set globally and remain fixed once the protocol has started:

1. T_{min}: The minimum number of blocks in a layer. A higher number increases the robustness of the protocol and decreases the interval between rewards for miners, at the cost of increased communication overhead.

2. ℓ: The average length of a layer interval. This is kept approximately constant by adjusting the difficulty parameter according to estimates of the total hash rate (similarly to Bitcoin's mechanism for difficulty adjustment).

3. δ: A bound on the combined network propagation time/local clock differences between honest nodes. If any honest node sees a block at time t according to its local clock, *all* honest nodes are guaranteed to receive the block by time $t+\delta$ according to *their own* local clocks. (Note that the actual network propagation delay is not tunable, but the bound we use is.)

4. t^{coin}: Bound on the time required to execute the weak coin protocol (in multiples of δ).

5. θ: A threshold for using the randomized block-voting protocol. If the vote margin (sum of the votes) for a block is less than θ (in absolute value) nodes will vote according to the output of the weak coin (instead of using the majority). A larger θ will guarantee convergence faster from arbitrary initial conditions, but will make it easier for the adversary to invalidate honest blocks (if the honest block has less than θ margin it might be invalidated by coin flip).

3.4.5 Protocol Concepts.

Syntactic Validity of Blocks. Syntactic validity of a block can be decided purely based on the block's visible mesh (i.e., it does not depend on "context", such as the time it was received or other blocks in the same or future layers). There are several conditions for a block to be syntactically valid in layer i:

1. It is not pointing to a syntactically invalid block.

2. The block has a valid proof-of-work for layer i's difficulty level (note that the difficulty level is a function of the view).

3. Its view contains at least $T_{\min}$ syntactically-valid blocks in layer $i-1$.

4. Transactions included in the block are valid according to the block view.

Layer Boundaries. Honest nodes maintain a *layer counter* which determines the serial number of the "current layer". The layer counter is initialized to 1 on receiving the genesis layer. The layer counter is incremented to $i+1$ when at least $T_{\min}$ syntactically-valid blocks were received in layer i.

The *honest start* for layer i, denoted $\texttt{start}_i$, is the time at which the first honest node increments its counter to i.

Transactions in Blocks. Each block contains a list of transactions. Unlike Bitcoin, however, the same transaction may appear in multiple blocks; what matters for the purpose of considering the transaction valid is whether it is included in a *layer.* We consider a transaction L to be included in layer i (according to block A) if i is the first layer in which L appears in at least one of the valid blocks of A's view. There is a fixed cap on the number of transactions that can be included in a block. The cap can be easily adjusted proportionally to the transaction publication rate in the system, and can award a bonus. (cf. Section 5.4). Every miner selects transactions to include in a block according to protocol 1.

Conflicting Transactions and Validity. There are no conflicts between blocks due to the transactions included in them. However, the transactions themselves may conflict (e.g., double spending). If two conflicting transactions are included in the same layer, the first one (according to the total ordering of blocks) is considered valid and the second invalid.

Protocol 1 Honest Mining Algorithm (transaction selection)

Let i^* be the latest layer about whose contents we are confident.
Let n be the cap for the number of transactions that can be included in a block.
Add a transaction L to the currently mined block up to the cap n if:

- L is not included in a layer $j \leq i^*$ in our view.
- There does not exist a transaction L' conflicting with L that is included in a layer $j \leq i^*$ in our view.

After reaching the cap, swap a transaction L with a transaction L' appearing in the currently mined block if L is not included in any published block up to the current layer and L' is.

Distributing Rewards We stress that the system for distributing the block rewards is completely independent of the security properties of Meshcash. In particular, we can choose any method for allocating the rewards that maintains the race-free property while ensuring incentive compatibility of Meshcash. We propose such as scheme in Section 8.4.

3.5 Security Proof Overview for Tortoise Protocol

In this section we give an informal overview of our security proof and intuitions.

3.5.1 Irreversibility.

At a high level, we can view the tortoise protocol as a voting process: every new block "votes" for or against all previous blocks. The irreversibility of the tortoise protocol stems from the fact that once consensus is reached, all honest users will vote in the same direction; this causes the margin of votes (the difference between positive and negative votes) to increase linearly with time. Similarly to the Bitcoin race analysis, an adversary can only reverse history by generating enough votes to overturn the current consensus. However, since the adversary generates blocks at a lower rate than the honest parties, the probability that this can be done decreases exponentially with time.

The main wrinkle here is that the adversary might keep a "reserve" of unpublished blocks and then publish them at a later date to reverse what seems like a consensus with large margin. However, in order to reverse the honest users' consensus about a block A, the adversary's reserve must contain "future" blocks (whose layer id is greater than that of block A)—since only future blocks have a "vote" regarding A. We show this cannot happen by bounding the adversary's ability to keep a large reserve of "future" blocks. In Corollary 4.10, we show that, irrespective of the initial conditions, there will be a layer in which the adversary's future reserve reaches a steady-state (see Definition 4.7). Additionally, we use the fact that with overwhelming probability no layer is "too long" (see Lemma 4.6) to prove that once in a steady state, the probability that the adversary leaves it is negligible; since in order to generate enough "future" blocks, the adversary needs a long layer-interval (see Corollary 4.11).

Finally, in Theorem 4.13 we formalize the race-analysis and show that the vote margin will grow linearly with the number of layers.

3.5.2 Pure Tortoise Consensus.

The harder part of the proof is to show that consensus will always (eventually) be achieved, even under active attack. Intuitively, the difficulty of guaranteeing consensus is due to the adversary's ability to "play" with network latency. By sending blocks near the "edge" of a layer, some honest parties would consider the block valid, while others would not. The voting scheme does not help in this instance, since the honest parties now disagree on the votes themselves (each vote is a block). Further complicating the analysis is that the adversary can generate and maintain a "reserve" of valid blocks (for the current or future layers) that can be used strategically by the adversary to cause disagreements among the honest miners on the contents of the layers.

Our main technical theorem, Theorem 4.12, shows that for any initial reserve of blocks (here we do not care about whether they are in the future or the past), the tortoise protocol will eventually arrive at consensus. We do this by a case analysis on the adversary's strategy, showing that the adversary has to "spend" her reserve in order to keep honest parties from agreement. Since the adversary's ability to generate new blocks is limited, either the honest parties will reach consensus, or the adversary will exhaust her reserve (in which case the honest parties will also reach consensus).

At a lower level, to show that the adversary must spend blocks from its reserve, we consider basically the following cases:

Case 1: There is already a large vote margin. In this case, the adversary has to spend at least that much blocks from her reserve to prevent consensus.

Case 2: The vote margin is small. In this case, some honest parties will use a coin-flip to choose how they vote, while others might see a large enough margin that they vote disregarding the coin. If the adversary spends too few blocks, we show that all parties that disregard the coin will vote in the same direction, so if the adversary does not guess the outcome of the coin correctly, all honest parties will agree.

3.5.3 Hare & Tortoise Consensus.

Theorem 4.12 does not use the hare protocol at all. Our bounds on the adversary's ability to pre-generate blocks (i.e., the bound on the future reserve) allow us to show that once we reach the future reserve steady-state, any hare protocol assuring limited consistency combined with a tortoise protocol will guarantee consistency and future self-consistency (see Corollary 4.14). The idea here is straightforward—once we have achieved consensus in the hare protocol, the honest parties all vote in the same direction;

3.5.4 Race-Freeness.

To show that honestly-generated blocks are always in the consensus, we need to lower-bound the number of honest blocks in every layer (since honest blocks are "guaranteed" to vote for other honest blocks). We can do this when the adversary is in a future reserve steady-state, by showing that no layer is too short (since the adversary can only shorten a layer by "dumping" blocks from its future reserve), which implies that the honest parties have enough time to generate blocks in every layer (see Lemma 4.15).

3.6 Hare Protocols

We propose several different hare protocols. The first is almost trivial, but guarantees consensus only when all participants are honest. The others are more complex, and use an off-chain asynchronous byzantine agreement protocol (ABA) to achieve a more robust consensus, even under malicious attack.

3.6.1 Simple Hare Protocol.

The simple hare protocol does not define additional edges (beyond the view and vote edges from the tortoise protocol). Its contextual validity rule is: "a block A claiming to be in layer i is valid iff it was received in the interval $(\texttt{start}_i - \delta, \texttt{start}_{i+1} + \delta)$".

Lemma 3.2. *The simple protocol has* $[1, 1]$*-limited consistency for honestly-generated blocks*

Proof. If all parties are honest, layer-i blocks are always mined in the interval $(\texttt{start}_i, \texttt{start}_{i+1})$, so every honest party will receive them in the interval $(\texttt{start}_i - \delta, \texttt{start}_{i+1} + \delta)$ and consider them valid. □

Note that while there is guaranteed consensus about honestly-generated blocks within a single layer, an adversary can create disagreement by publishing blocks near the layer-interval boundary, so that some honest parties receive them before the boundary and some after. In this case, we have to rely on the tortoise protocol to achieve consensus.

3.6.2 Byzantine-Agreement-Based Hare Protocols.

Overview. The basic idea behind the ABA-based hare protocols is to use the blocks published in a layer to select a committee that will then run an off-chain "traditional" byzantine agreement protocol to achieve consensus about the validity of each block in the layer. The off-chain protocol can be performed using direct links, so can be quite fast; at the end of the protocol, the committee will append digital signatures (using public keys provided in their published blocks) testifying to each block's validity. The contextual validity rule is now "a block A claiming to be in layer i is valid iff it has valid signatures from a majority of the layer-i committee members".

There are two main difficulties to overcome in using this paradigm: the first is that we do not have consensus about the committee members; standard ABA protocols assume there is agreement on the participants. We solve this problem by using an identity-free ABA protocol (we use the protocol of Mostfaoui et al [29]) and a bound on the total number of participants. Every party will interact with the committee members it recognizes, and consider any others as faulty. The properties of the ABA protocol guarantee that security still holds as long as the number of honest parties is 2/3 of the *maximal* number of participants.

The second problem is that the ABA protocol uses a "weak random coin": this is usually implemented via a cryptographic protocol that requires trusted setup, which we do not have. However, there are some constructions that do not require setup. Alternatively, we can implement the weak-random coin using the blockchain itself (unfortunately, our coin implementation has a minimum time per coin flip, so each round of the ABA protocol has a minimum time as well. However, the ABA protocol we chose needs a very small number of rounds in expectation, so with high probability the total time will still be less than one layer interval. We describe several ABA-Based Hare protocols in Section 6 (which trade increasing complexity for better security assumptions).

3.7 Communication/Storage Optimizations.

Note that the transmitted and stored transaction data does not need to grow linearly with the number of blocks, because the miners can store each transaction once, regardless of how many blocks it appears in; the blocks themselves need only contain a transaction id (computed as a collision-resistant hash of the transaction data).

In Bitcoin, the average size of a transaction is about 525 bytes[3] , while the transaction id hash is 32 bytes, so this reduces the data size needed by a factor of ≈ 16. Notice that this optimization reduces both the communication complexity and the computational complexity of computing the hashes of blocks. Furthermore, miners can refer to transactions in their local storage according to their index among all the transactions that they maintain, so the local storage complexity (not communication complexity) can be 64 times smaller than the worst case if we assume that each index is 8 bytes.

[3] The average size of a block is about 900 kilobytes and the average number of transactions in a block is about 1700, see also `https://tradeblock.com/blog/analysis-of-bitcoin-transaction-size-trends` and `http://p2sh.info/dashboard/db/non-standard-outputs-statistics`.

Note that in the current Bitcoin blockchain, transactions are the dominating factor in terms of storage. By using this optimization, the communication overhead for a 100-fold increase in block rate (compared to Bitcoin) will be about 6 times greater, while the storage overhead factor can be less than 2.

3.7.1 Efficient Verification Algorithm.

Since all blocks should gain an overwhelming voting margin after $O(1)$ layers (except when the network is under an active attack), it is expected that irreversibility will take hold quickly. Therefore, miners can maintain a separate structure only for tentative past blocks, i.e., old data in the history can be pruned and nodes will record that the more recent blocks are valid without actually storing the older blocks. However, in order to make sure that the adversary cannot exploit the efficient verification algorithm, honest nodes need to be able detect that they pruned too much old data and thus request missing data from peers that do not prune (similarly to archival nodes versus nodes that maintain only the UTXO set in Bitcoin).

4 Proof of Security

4.1 Notation

For the proofs and analysis we introduce some additional notation.

Tortoise Protocol Variables. We define the following random variables:

- Y_j: the number of maliciously-generated blocks in the interval $[\texttt{start}_j + \delta, \texttt{start}_{j+1} + \delta]$
- Z_j^{coin}: the number of non-abstaining blocks in layer j. this is at least the number of honestly-generated blocks in the interval $[\texttt{start}_j + t^{\text{coin}}\delta, \texttt{start}_{j+1}]$ (for t^{coin} definition, see Section 3.4.4)

Tortoise Protocol Variables Bounds. We also define the bounds on these variables, such that the following are guaranteed with overwhelming probability (in $T_{\min}$):

- ℓ_ε^*: an upper bound on the duration of a layer. $\forall j : \texttt{start}_{j+1} - \texttt{start}_j \le \ell_\varepsilon^*$. We choose $\ell_\varepsilon^* = (1+\varepsilon) \cdot \frac{\ell}{1-q} + 2\delta$. We parameterize by ε; for every constant ε, Lemma 4.6 guarantees that the probability of exceeding the bound is negligible. Note that using the assumption section above, $\ell_\varepsilon^* < (1+\varepsilon)\frac{14}{10}\ell$
- Y_ε^*: an upper bound on the number of maliciously-generated blocks $\forall j : Y_j \le Y_\varepsilon^*$ except with probability $\Pr_{X \sim \text{Pois}\left(q \cdot \frac{\ell_\varepsilon^*}{\ell} \cdot T_{\min}\right)} [X \ge Y_\varepsilon^*]$ (for this to be negligible, we take $Y_\varepsilon^* \ge (1+\varepsilon)^2 \cdot q \cdot (\frac{1}{1-q} + \frac{\delta}{\ell}) \cdot T_{\min} \ge (1+\varepsilon) q \cdot \frac{\ell_\varepsilon^*}{\ell} \cdot T_{\min}$ for some constant ε).
- ℓ_ε^-: a lower bound on the duration of a layer in which the adversary is in the future-reserve steady-state. $\forall j$ s.t. the adversary is in its future-reserve steady-state, $\texttt{start}_{j+1} - \texttt{start}_j \ge \ell_\varepsilon^-$ (we choose $\ell_\varepsilon^- \le (1-\varepsilon) \cdot \frac{(T_{\min} - F^*)}{T_{\min}} \ell$ and apply Lemma 4.15 here).
- Z_ε^*: a bound on the number of honestly-generated blocks in interval $[\texttt{start}_j + \delta, \texttt{start}_{j+1}]$. $\forall j$ s.t. the adversary is in its future-reserve steady-state, $Z_j^{\text{coin}} \ge Z_\varepsilon^*$ except with probability $\Pr_{X \sim \text{Pois}\left((1-q)\frac{\ell_\varepsilon^- - t^{\text{coin}}\delta}{\ell} \cdot T_{\min}\right)} [X \le Z_\varepsilon^*]$ (for this to be negligible, we take $Z_\varepsilon^* \le (1-\varepsilon)(1-q)\frac{\ell_\varepsilon^- - t^{\text{coin}}\delta}{\ell} T_{\min}$ for some constant ε).

Tortoise Protocol Assumptions. For our analysis, we assume the following:

- $q < 1/15$: a bound on the computational power of the adversary (the hash-rate of the adversary is at most a q fraction of the entire network's hash rate). This is required to guarantee convergence by the tortoise protocol when the plugged-in hare protocol fails to reach consensus.

- $\ell > 4\delta$: a lower bound on the expected layer time w.r.t. the network delay. Even if assuming a large network delay time (e.g. $\delta < 30$ seconds) resulting in $\ell > 2$ minutes, it still allows a faster confirmation time compared to Bitcoin.

Definition 4.1 (Adversary's Reserve)**.** The adversary's *reserve* at time τ is the set of syntactically valid blocks generated by the adversary that have not been published up to time τ. The adversary's reserve for a block A claiming to be in layer i is the subset of blocks in its reserve whose vote for A is not neutral (i.e., blocks in layers after $i + t$). For $j > i$, we denote $R_j(i)$ the adversary's reserve for blocks in layer i at time $\texttt{start}_j + \delta$ (not including blocks that were published at time $\texttt{start}_j + \delta$).

Definition 4.2 (Adversary's *Future* Reserve)**.** The adversary's *future reserve* at time $\texttt{start}_i$ is the subset of blocks in its reserve that claim to be in future layers (i or greater). We denote F_i the adversary's future reserve at time $\texttt{start}_i$.

Definition 4.3 (Adversary's Late-Published Blocks)**.** The adversary's late-published blocks, Δ_{j-1}, is the number of blocks published by the adversary from its reserve at time $\texttt{start}_j + \delta$.

For any layer $i < j$, if the honest parties on layer j agree on a block A claiming to be in layer i, the maximal number of blocks that can counter-vote their agreement on A is Δ_{j-1}. We assume w.l.o.g. that all of the adversary's blocks are late-published.

Definition 4.4 (Visible Confirmation Margin)**.** The i^{th} (visible) *confirmation margin* for a block A claiming to be in layer $j < i$ is the sum of the votes for A from all syntactically valid blocks in layers $i' \geq j$ that were received by time $\texttt{start}_i + \delta$. We denote $M_i^{(P)}(A)$ the i^{th} confirmation margin for block A as seen by party P (we may omit P when it is clear from the context).

Informally the *confirmation margin* for a block is how confident we are about the block's validity (or invalidity).

Definition 4.5 (Actual Confirmation Margin)**.** Let $P_i^{\max}$ denote P such that $|M_i^{(P)}(A)|$ is maximized and $M_i^{(\max)}(A) = M_i^{(P_i^{\max})}(A)$. The i^{th} *actual confirmation margin* for a block A claiming to be in layer $j < i$ is $M_i^{\dagger}(A) = M_i^{(\max)}(A) - \Delta_{i-1} - |R_i(j)|$.

If the actual confirmation margin is large enough (greater than θ), it means all honest parties agree on the validity of A, and the adversary does not have enough blocks in its reserve to change this opinion. This is used to bound any future attempts by the adversary to change the validity of block A (see Theorem 4.13).

Lemma 4.6 states that with overwhelming probability in $T_{\min}$, we can bound the length of any layer time interval by $\ell_\varepsilon^* = (1+\varepsilon) \cdot \frac{\ell}{1-q} + 2\delta$.

Lemma 4.6. *For any layer i and $\varepsilon > 0$,*

$$\Pr[\texttt{start}_{i+1} - \texttt{start}_i > \ell_\varepsilon^*] \leq \left(2^{\frac{(1+\varepsilon)-\ln(1+\varepsilon)-1}{\ln 2}}\right)^{-T_{min}}$$

(see Section 4.1 for ℓ_ε^* definition)

Proof. By definition, $\left[\texttt{start}_i, \texttt{start}_{i+1}\right]$ describes the time interval between any honest party knowing $T_{\min}$ blocks on layer $i-1$ and any honest party knowing $T_{\min}$ blocks on layer i.

The honest parties must start generating layer i blocks before $\texttt{start}_i + \delta$. Since the adversary cannot delay any honestly-generated block by more than δ, the best strategy it has for increasing the layer interval is to refrain from publishing blocks. The difficulty adjustment for the protocol ensures that the expected time for the entire network (including the adversary) to generate $T_{\min}$ blocks is ℓ. Since the adversary controls at most a q fraction of the hash power, the honest parties by themselves will generate at expected rate $(1-q)T_{\min}$ per ℓ, or $T_{\min}$ in an interval of length $\frac{\ell}{1-q}$. These $T_{\min}$ blocks will be seen by the entire network at most δ time after generated. Thus, the probability that any interval length is longer than $\ell^*_\varepsilon = (1+\varepsilon)\cdot\frac{\ell}{1-q} + 2\cdot\delta$ can be tail-bounded. According to Poisson tail bounds (cf. Appendix B), for all $c > 1$,

$$\Pr_{X \sim \mathrm{Pois}(c\lambda)}\left[X \le \lambda\right] \le 2^{-\lambda\frac{c-\ln(c)-1}{\ln 2}} = \left(2^{\frac{c-\ln(c)-1}{\ln 2}}\right)^{-\lambda}.$$

In particular, for $c = (1+\varepsilon), \lambda = T_{\min}$

$$\Pr_{X \sim \mathrm{Pois}((1+\varepsilon)T_{\min})}\left[X \le T_{\min}\right] \le \left(2^{\frac{(1+\varepsilon)-\ln(1+\varepsilon)-1}{\ln 2}}\right)^{-T_{\min}}.$$

□

4.2 Bounding the Layer Interval and the Size of the Future Reserve

Our tortoise consensus proof requires a bound on the size of the future reserve (otherwise the adversary effectively has unlimited computational power, since publishing blocks from the future reserve is "free" for the adversary).

Definition 4.7 (Future reserve steady-state)**.** We say the adversary is in a *future reserve steady-state* at time $\texttt{start}_i$ if $|F_i| \le F^* = 3\cdot\frac{q}{1-q}\cdot T_{\min}$.

Lemma 4.8. *For every $\varepsilon > 0$ and $i > 0$, if the adversary controls at most q-fraction of the computational power, then*

$$\Pr\left[|F_{i+1}| > (1+\varepsilon)\frac{\ell^*_\varepsilon}{\ell}\cdot qT_{min} \;\middle|\; F_i \cap F_{i+1} = \emptyset\right] < 2\cdot\left(2^{\frac{(1+\varepsilon)-\ln(1+\varepsilon)-1}{\ln 2}}\right)^{-\min(\frac{1}{q},\frac{1+\varepsilon}{1-q}+\frac{2\delta}{\ell})\cdot qT_{min}}$$

Proof. If $F_i \cap F_{i+1} = \emptyset$, the adversary cannot have any blocks of layer $i+1$ in its reserve at time $\texttt{start}_i$ (otherwise they would be in $F_i \cap F_{i+1}$). Thus, F_{i+1} can only contain blocks that were generated in the interval $[\texttt{start}_i, \texttt{start}_{i+1}]$.

We will bound F_{i+1} by bounding the length of the layer interval, and then using a Poisson tail bound to bound the number of blocks the adversary can generate in that interval. Let Bad_{long} be the event that $\texttt{start}_{i+1} - \texttt{start}_i > \ell^*_\varepsilon$ (i.e., the layer-interval is "too long"). By Lemma 4.6, $\Pr\left[Bad_{long}\right] < \mathbf{neg}(T_{\min})$.

Let Bad_{fast} be the event that the adversary generated "too many" (more than $(1+\varepsilon)\cdot\frac{\ell^*_\varepsilon}{\ell}\cdot qT_{\min}$) blocks' in the interval $[\texttt{start}_i, \texttt{start}_i + \ell^*_\varepsilon]$. Denote X the number of blocks generated by the adversary in the interval . Since the adversary's block generation is bounded by a Poisson process with parameter $\frac{\ell^*_\varepsilon}{\ell}\cdot qT_{\min}$ for an interval of length ℓ^*_ε,

$$\begin{aligned}
\Pr\left[Bad_{fast}\right] &= \Pr_{X \sim \mathrm{Pois}\left(\frac{\ell^*_\varepsilon}{\ell}\cdot qT_{\min}\right)}\left[X > (1+\varepsilon)\cdot\frac{\ell^*_\varepsilon}{\ell}\cdot qT_{\min}\right] \\
&\le \left(2^{\frac{1+\varepsilon}{\ln 2}(\frac{1}{1+\varepsilon}+\ln(1+\varepsilon)-1)}\right)^{-\frac{\ell^*_\varepsilon}{\ell}\cdot qT_{\min}} \\
&= \left(2^{\frac{1+\varepsilon}{\ln 2}(\frac{1}{1+\varepsilon}+\ln(1+\varepsilon)-1)}\right)^{-\left(\frac{1+\varepsilon}{1-q}+\frac{2\delta}{\ell}\right)\cdot qT_{\min}}
\end{aligned}$$

(where the final identity follows from the definition $\ell^*_\varepsilon = (1+\varepsilon)\cdot\frac{\ell}{1-q} + 2\cdot\delta$).

Since $|F_{i+1}|$ is a subset of adversary blocks created during $[\mathtt{start}_i, \mathtt{start}_{i+1}]$, we must have $|F_{i+1}| < |X| < (1+\varepsilon)\frac{\ell^*_\varepsilon}{\ell}\cdot qT_{\min}$ unless either Bad_{long} or Bad_{fast} occurred.

Thus, by the union bound,

$$\begin{aligned}
\Pr\left[|F_{i+1}| > (1+\varepsilon)\frac{\ell^*_\varepsilon}{\ell}\cdot qT_{\min} \;\middle|\; F_i \cap F_{i+1} = \emptyset\right] &\le \Pr\left[Bad_{long}\right] + \Pr\left[Bad_{fast}\right] \\
&\le \left(2^{\frac{(1+\varepsilon)-\ln(1+\varepsilon)-1}{\ln 2}}\right)^{-T_{\min}} \\
&\quad + \left(2^{\frac{(1+\varepsilon)+\ln(1+\varepsilon)-1}{\ln 2}}\right)^{-\left(\frac{1+\varepsilon}{1-q}+\frac{2\delta}{\ell}\right)\cdot qT_{\min}} \\
&\le 2\cdot\left(2^{\frac{(1+\varepsilon)-\ln(1+\varepsilon)-1}{\ln 2}}\right)^{-\min\left(\frac{1}{q},\frac{1+\varepsilon}{1-q}+\frac{2\delta}{\ell}\right)\cdot qT_{\min}}
\end{aligned}$$

which is negligible in $T_{\min}$

□

Theorem 4.9. *For every initial future reserve F_{i_0}, the adversary will have no far-future blocks after $|F_{i_0}|\cdot\frac{1-q}{(1-2q)T_{min}}$ layers (in expectation).*

Proof. First, note that since every syntactically valid block in layer j must point to $T_{\min}$ syntactically valid blocks in layer $j-1$, if F_i includes any blocks from layer $j \ge i+1$ then it must include at least $T_{\min}$ blocks in *each* of the layers $i,\ldots,j-1$. Thus, if the adversary has "far-future" blocks in F_i, its future reserve will shrink by at least $T_{\min}$ blocks at every layer boundary (since F_j does not contain any blocks in layers less than j).

Even if the adversary uses all of its hash power to increase its reserve, the expected layer-interval length will be at most $\frac{\ell}{1-q}$; thus, the adversary's expected block-generation per layer-interval is bounded by $\frac{q}{1-q}\cdot T_{\min} < T_{\min}$ (this is a very loose bound that ignores difficulty adjustment). If the adversary has far-future blocks in F_i, the size of F_i will decrease, in expectation, by at least $T_{\min} - \frac{q}{1-q}\cdot T_{\min} = \frac{1-2q}{1-q}\cdot T_{\min}$ at every layer-interval, so the adversary will be left with no far-future blocks in expectation, within $|F_{i_0}|\cdot\frac{1-q}{(1-2q)T_{\min}}$ layers. □

Corollary 4.10 states that for any initial reserve, the adversary's future reserve will eventually shrink to the steady-state. The total number of layers until the adversary future-reserve is bounded depends only on the size of its initial reserve and fraction of the total hash power).

Corollary 4.10. *For every initial future reserve F_{i_0}, with probability 1 there will be some $i^* > i_0$ such that $|F_{i^*}| < F^*$.*

Proof. By Theorem 4.9, (in expectation) the adversary will have no far-future blocks after $|F_{i_0}|\cdot\frac{1-q}{(1-2q)T_{\min}}$ layers. Let i' be the layer in which the adversary no longer has far-future blocks. By Lemma 4.8 with overwhelming probability:

$$|F_{i'+1}| \le (1+\varepsilon)\frac{\ell^*_\varepsilon}{\ell}\cdot qT_{\min}$$

By definition, for any $\varepsilon > 0$, $\ell^*_\varepsilon = (1+\varepsilon)\cdot\frac{\ell}{1-q} + 2\cdot\delta$. Therefore,

$$\begin{aligned}
|F_{i'+1}| &\le (1+\varepsilon)\left((1+\varepsilon)\cdot\frac{1}{1-q} + \frac{2\delta}{\ell}\right)\cdot qT_{\min} \\
&\le \left((1+\varepsilon)^2 + \frac{(1+\varepsilon)\cdot(1-q)\cdot 2\delta}{\ell}\right)\cdot\left(\frac{q}{1-q}\right)T_{\min} \\
&\le \left((1+\varepsilon)^2 + (1+\varepsilon)\right)\cdot\left(\frac{q}{1-q}\right)T_{\min}
\end{aligned}$$

where the last bound is by Section 4.1 ($\delta < 4\ell$ and $q < \frac{1}{15}$). By setting $\varepsilon = 0.25$ we get:

$$|F_{i'+1}| < 3\left(\frac{q}{1-q}\right)T_{\min} = F^*$$

□

Corollary 4.10 proves that whatever the initial conditions, the adversary will eventually reach steady state. Corollary 4.11 proves that once the adversary reaches steady-state, it will stay there except with negligible probability:

Corollary 4.11. *For any layer i, if $|F_i| < F^*$, then*

$$\Pr\left[|F_{i+1}| > F^*\right] < \left(2^{\frac{(1+\varepsilon)-\ln(1+\varepsilon)-1}{\ln 2}}+1\right)^{-\left(\frac{1+\varepsilon}{1-q}+\frac{2\delta}{\ell}\right)\cdot T_{min}}$$

Proof. If $|F_i| < F^* < T_{\min}$, the adversary cannot have far-future blocks (at least $T_{\min}$ layer i blocks are required to generate layer $i+1$ blocks). Since having no far-future blocks, by Lemma 4.8

$$|F_{i+1}| \leq (1+\varepsilon)\frac{\ell^*_\varepsilon}{\ell} \cdot qT_{\min}$$

except with negligible probability. Using the same analysis as in Corollary 4.10, when $\varepsilon = 0.25$

$$|F_{i+1}| < 3\left(\frac{q}{1-q}\right)T_{\min} = F^*$$

. □

4.3 Consistency and Future Self-Consistence (Irreversibility)

Theorem 4.12. *If*

$$q \leq \min\left(\frac{\ell}{\ell^*_\varepsilon} \cdot \frac{2\theta}{T_{min}} \cdot p_{coin};\ 1 - \frac{\ell}{\ell^-_\varepsilon - t^{coin}\delta} \cdot \frac{4\theta}{T_{min}}\right)$$

then for every initial reserve $R_{i_0}(A)$ and every initial margin $M^{(\max)}_{i_0+1}(A)$, and every $k > 0$ there exists $i^* > i_0$ such that with probability 1 it holds that $M^\dagger_{i^*}(A) > \theta + k$.

Since the proof of Theorem 4.12 is involved, we defer its presentation to Section 5. We use the following parameters' values:

1. By Section 4.1, $\ell^*_\varepsilon < (1+\varepsilon)\frac{14}{10}\ell$. Thus, by setting $\varepsilon = 1/14$ we achieve
$$\frac{\ell^*_\varepsilon}{\ell} \leq \frac{2}{3}\ .$$
2. By setting $p_{\text{coin}} = \frac{1}{6}$ and given our definition in Section 4.1, we get $T > 6\delta$ as a lower bound on time for the weak coin protocol to terminate. Given $q \leq \frac{1}{15}$, we can assume $\frac{\theta}{T_{\min}} \leq \frac{1}{12}$, i.e. honest blocks will go to vote for a block A if there's at least 1/12 of blocks disagreeing on its validity. If $\ell > 24\delta$ (roughly 10 minutes), then by expectation 3/4 of honest blocks will see the weak coin's value after termination. Thus, we set:
$$\frac{2\theta}{T_{\min}} \cdot p_{\text{coin}} \leq \frac{2}{3} \cdot \frac{1}{6} = \frac{1}{9}$$

Therefore:

$$q \leq \frac{1}{15} < \min\left(\frac{2}{3} \cdot \frac{1}{9},\ 1 - \frac{4}{6}\right) = \min\left(\frac{2}{27}, \frac{1}{3}\right) \ .$$

Theorem 4.13 shows that if the actual margin for a block is positive at any point (by Theorem 4.12, this must eventually happen), then with high probability the block's validity is irreversible.

Theorem 4.13. *For* i, $i' > i$, $k > 1$ *and every block* A *claiming to be in layer* i*, if* $M^{\dagger}_{i'}(A) > \theta + k$ *then*

$$\Pr\left[\exists j > i' : \boldsymbol{sign}\left(M_j(A)\right) \neq \boldsymbol{sign}\left(M_{i'}(A)\right)\right] < \left(\frac{q}{1-q}\right)^k \ .$$

Proof. Since $M^{\dagger}_{i'}(A) > \theta + k$, all honest parties agree on A's validity at time $\texttt{start}_{i'} + \delta$, since for any honest P, $|M^{(P)}_{i'}(A) - M^{(\max)}_{i'}(A)| \leq X_{i'} + \Delta_{i-1}$. Moreover, this would be true even if the adversary published all the blocks in its reserve. Thus, we can reduce the probability of the adversary changing the validity of A to the Bitcoin analysis (cf. [37]): both the adversary and the honest nodes generate blocks, with the honest nodes generating at rate $(1-q)$ and the adversary at rate q.

Denoting a_z the probability that the adversary will *ever* catch up when the honest party has a head start of z blocks, we have

$$a_z = \min(q/(1-q), 1)^{\max(z+1,0)} \ .$$

In our case, $z \geq k$ and $q < 1-q$ so $a_z = \left(\frac{q}{1-q}\right)^k$. □

By Corollary 4.10, regardless of starting conditions, the adversary will eventually reach a steady-state in terms of its future reserve, in which the number of blocks it can pre-generate is bounded. When this occurs:

Corollary 4.14. *If* Π *satisfies* $[t,s]$*-limited-consistency, and the adversary is in its future-reserve steady-state, then the combined tortoise/*Π *protocol satisfies consistency and future self-consistence.*

Proof. First, since the all honest nodes agree on every block in layers $(i-s,\ldots,i-t)$, all honestly-generated blocks in layer i will vote the same way for these blocks. With overwhelming probability (in $T_{\min}$), the actual margin (the difference between the number of honest blocks generated in these layers and the number of blocks generated by the adversary, together with its future reserve), will be more than θ, by Theorem 4.13, this guarantees irreversibility (self-consistency). □

Lemma 4.15 states that no layer is "too short". This will be used later to assure that in each layer there is an honest majority if the adversary is in a future-reserve steady-state.

Lemma 4.15. *For* $\ell^-_\varepsilon \leq (1-\varepsilon) \cdot \frac{(T_{min}-F^*)}{T_{min}}\ell$*, if the* $|F_i| < F^*$ *and* $q \leq 1/10$*, then for every* $\varepsilon > 0$

$$\Pr\left[\texttt{start}_{i+1} - \texttt{start}_i < \ell^-_\varepsilon\right] \leq \left(2^{\frac{\varepsilon - \ln(1+\varepsilon)}{\ln 2}}\right)^{-\frac{2}{3}T_{min}} \ .$$

Proof. Let L_i be actual length of layer i i.e.

$$L_i = \texttt{start}_{i+1} - \texttt{start}_i \ .$$

We wish to represent the expected layer time w.r.t F_i, the adversary future-reserve. L_i is at least the time it takes to generate and publish $T_{\min} - F_i$ blocks so for any i

$$\mathbb{E}\left[L_i\right] \leq \ell\frac{T_{\min} - F_i}{T_{\min}} \ .$$

According to its definition, $\texttt{start}_{i+1}$ is set to the time in which the first honest node first sees $T_{\min}$ blocks on layer i. Thus, for any L', the probability of layer i being shorter than L' is the probability of generating more than $T_{\min} - F_i$ blocks within time L'. Since the block generation process is a Poisson process with expectation $T_{\min}$ in time ℓ, the number of blocks generated during an interval of length L' is distributed Poisson with parameter $T_{\min} \cdot \frac{L'}{\ell}$. Thus, the probability that $L_i < L'$ can be bounded by the probability that a Poisson process with parameter $T_{\min} \cdot \frac{L'}{\ell}$ generates more than $T_{\min} - F_i$ events. Taking L' to be $(1-\varepsilon)\ell\frac{T_{\min}-F_i}{T_{\min}}$ we get:

$$\begin{aligned}\Pr\left[L_i \leq (1-\varepsilon)\ell\frac{T_{\min}-F_i}{T_{\min}}\right] &= \Pr_{X\sim\text{Pois}((1-\varepsilon)\cdot(T_{\min}-F_i))}\left[X \geq T_{\min} - F_i\right] \\ &\leq \left(2^{\frac{\varepsilon-\ln(1+\varepsilon)}{\ln 2}}\right)^{F_i - T_{\min}}\end{aligned}$$

and if the adversary is on future-reserve steady-state on $\texttt{start}_i$ then $F_i \leq F^*$, thus

$$\begin{aligned}\Pr\left[\texttt{start}_{i+1} - \texttt{start}_i < (1-\varepsilon)\cdot\frac{(T_{\min}-F^*)}{T_{\min}}\ell\right] &\leq \left(2^{\frac{\varepsilon-\ln(1+\varepsilon)}{\ln 2}}\right)^{F^*-T_{\min}} \\ &\leq \left(2^{\frac{\varepsilon-\ln(1+\varepsilon)}{\ln 2}}\right)^{-\frac{2}{3}T_{\min}}\end{aligned}$$

where the last inequality is since $F^* = 3\left(\frac{q}{1-q}\right)T_{\min} \leq \frac{1}{3}T_{\min}$ if $q < 1/10$ □

Lemma 4.16. *Except with negligible probability in T_{min}, for every layer i if the adversary is in future-reserve steady-state the difference between honest blocks and adversarial blocks in layer i is at least $T_{min} - 2F^*$*

Proof. For any layer i, let L_i be defined as the actual layer i time that is:

$$L_i = \texttt{start}_{i+1} - \texttt{start}_i \ .$$

Let X_i be the number of honest blocks in layer i created in $[\texttt{start}_i, \texttt{start}_{i+1}]$. As all honest nodes follow the protocol, the expected value of X_i w.r.t. L_i is:

$$E[X_i] = (1-q)T_{\min} \cdot \frac{\mathbb{E}\left[L_i\right]}{\ell} \ .$$

Let Y_i be the number of adversary blocks in layer i created in $[\texttt{start}_i, \texttt{start}_{i+1}]$. For any layer i, by $\texttt{start}_{i+1}$, the adversary is expected to have at most:

1. $\mathbb{E}\left[|F_i|\right]$ (her future-reserve blocks); and
2. $qT_{\min} \cdot \frac{\mathbb{E}[L_i]}{\ell}$ blocks created during L_i

that is

$$E[Y_i] \leq \mathbb{E}\left[|F_i|\right] + qT_{\min} \cdot \frac{\mathbb{E}\left[L_i\right]}{\ell} \ .$$

Due to linearity of expectation, the expected difference between honest and adversary blocks for any layer i is:

$$\mathbb{E}\left[X_i - Y_i\right] = \mathbb{E}\left[X_i\right] - \mathbb{E}\left[Y_i\right] \geq T_{\min} \cdot \frac{\mathbb{E}\left[L_i\right]}{\ell} - \mathbb{E}\left[|F_i|\right]$$

which is monotonic decreasing for $\mathbb{E}\left[L_i\right]$, thus a shorter $\mathbb{E}\left[L_i\right]$ allows a smaller expected difference between honest and adversary blocks.

For any layer i,

$$\mathbb{E}\left[L_i\right] > \frac{T_{\text{min}} - F^*}{T_{\text{min}}}\ell.$$

That is the expected number of honest blocks for any layer i has the following lower-bound:

$$E[X_i] \geq (1-q)(T_{\text{min}} - F^*)$$

and if the adversary is in future-reserve steady-state, for any layer i it follows $F_i \leq F^*$

$$E[Y_i] \leq F^* + q(T_{\text{min}} - F^*) \ .$$

According to Lemma 4.15, $L_i > (1+\varepsilon)\frac{T_{\text{min}}-F^*}{T_{\text{min}}}\ell$ with overwhelming probability so:

1. Denote $\lambda_X = (1-q)(T_{\text{min}} - F^*)$ per $\frac{T_{\text{min}}-F^*}{T_{\text{min}}}\ell$ as the rate of honest blocks generation. By applying a Poisson tail bound,

$$\forall i: \Pr_{X\sim\text{Pois}(\lambda_X)}\left[X_i \leq (1-\varepsilon)\lambda_X\right] \leq \left(2^{\frac{\varepsilon - \ln(1+\varepsilon)}{\ln 2}}\right)^{-\lambda_X}.$$

2. Denote $\lambda_Y = F^* + q(T_{\text{min}} - F^*)$ per $\frac{T_{\text{min}}-F^*}{T_{\text{min}}}\ell$ as the rate of adversary blocks generation. By applying a Poisson tail bound,

$$\forall i: \Pr_{X\sim\text{Pois}(\lambda_Y)}\left[Y_i \geq (1+\varepsilon)\lambda_Y\right] \leq \left(2^{\frac{\varepsilon - \ln(1+\varepsilon)}{\ln 2}}\right)^{-\lambda_Y}.$$

Therefore, except with negligible probability in T_{min} for any layer i:

$$\begin{aligned} X_i - Y_i &> T_{\text{min}} \cdot \frac{L_i}{\ell} - F^* \\ &> (T_{\text{min}} - 2 \cdot F^*) \\ &= T_{\text{min}} - 2 \cdot 3\frac{q}{1-q}T_{\text{min}} \end{aligned}$$

since $q \leq \frac{1}{10}$:

$$\begin{aligned} &\geq T_{\text{min}} - 2 \cdot 3 \cdot \frac{1/10}{9/10}T_{\text{min}} \\ &= \frac{T_{\text{min}}}{3} \end{aligned}$$

□

Lemma 4.17. *With overwhelming probability in T_{min}, if*

$$M_i^{\dagger}(A) > \theta + k$$

then for all $i' > i$,

$$M_{i'}^{\dagger}(A) > \theta + k + (i' - i) \cdot \frac{\theta}{3} \ .$$

Proof. By Lemma 4.16, with overwhelming probability (in T_{min}) any layer in which the adversary is in a future-reserve steady-state will have at least $\frac{T_{\text{min}}}{3} \geq \frac{\theta}{3}$ more honest blocks than adversary blocks (since trivially $\theta \leq T_{\text{min}}$). Also, by Corollary 4.11 if the adversary is in steady-state when the actual margin exceeds $\theta + k$, she will remain with overwhelming probability. Thus except with negligible probability in each layer, the margin grows by at least the number of honest blocks minus the number of malicious blocks (which is at least $\theta/3$). □

Corollary 4.14 and Lemma 4.17 imply consistency:

Corollary 4.18 (Consistency)**.** *If $|F_i| < F^*$, $Z^*_\varepsilon - (F^* + Y^*_\varepsilon) > \theta$ and the requirements for the hare protocol's $[t, s]$-limited-consistency are met, then for every block A generated in the interval $[\texttt{start}_i, \texttt{start}_{i+1})$, the probability that there exist two honest nodes who disagree on the validity of A at time $\texttt{start}_{i+t}$ is negligible in t.*

Proof. If the hare protocol satisfies $[t, s]$-limited-consistency, there will be consensus on every block in the range of the tortoise protocol with probability that is negligible in the size of the margin. By Lemma 4.17, the size of the margin is linear in the number of layers, hence the disagreement is negligible in the number of layers. □

5 Proof of Theorem 4.12

5.1 Proof Overview

In order to bound the actual confirmation margin, it will be useful to look at the quantities

$$R^*_i(j) = 2|R_i(j)| - \Delta_{i-1}$$

and

$$M^*_i(A) = |M^{(\max)}_i(A)| - R^*_i(j) \ .$$

If for any positive α, β we'll be able to bound $M^*_i(A)$ s.t.

$$M^*_i(A) = \sum_{j=0}^{i} \left(M^*_{j+1}(A) - M^*_j(A)\right) \geq i \cdot \alpha - \beta \ ,$$

then for any c exists a layer i_c in which $M^*_{i_c}(A) > c$. We prove the following by exhaustive case analysis over the value of Δ_{i-1} and $|M^{(\max)}_i(A)|$ in each layer between j to i.

We then show using a followup case analysis that by fixing $c = \theta + k + 4Y^*_\varepsilon$ either the reserve decreases or by Definition 4.5 :

$$\begin{aligned} M^\dagger_i(A) &= |M^{(\max)}_i(A)| - |R_i(j)| - \Delta_{i-1} \\ &= M^*_i(A) + |R_i(j)| - 2\Delta_{i-1} \\ &\geq \theta + k \ . \end{aligned}$$

As the reserve is discrete, it must follow that after at most $|R_{i_c}(A)|$ layers, $M^\dagger_i(A) \geq \theta + k$.

5.2 Bounding the quantity $M^*_i(A)$

We will use the identity:

$$\begin{aligned} R^*_{i+1}(j) - R^*_i(j) &= 2(|R_{i+1}(j)| - |R_i(j)|) + \Delta_{i-1} - \Delta_i \\ &= 2(Y_i - \Delta_{i-1}) + \Delta_{i-1} - \Delta_i \\ &= 2Y_i - \Delta_{i-1} - \Delta i \ . \end{aligned}$$

For any $j < i$, instead of bounding the actual margin, we will bound

$$\begin{aligned} M^*_{i+1}(A) - M^*_i(A) &= |M^{(\max)}_{i+1}(A)| - |M^{(\max)}_i(A)| - (R^*_{i+1}(j) - R^*_i(j)) \\ &= |M^{(\max)}_{i+1}(A)| - |M^{(\max)}_i(A)| - 2Y_i + \Delta_i + \Delta_{i-1} \ . \end{aligned}$$

We partition the layers according to two things: Δ_{i-1} (see Definition 4.3), the number of blocks from $R_{i-1}(A)$ published by the adversary at time $\texttt{start}_i + \delta$ and $M^{(\max)}_i(A)$ (see Definition 4.4). As we show below, depending on the relation between these values and θ (see Section 3.4.4), we can bound the difference $M^*_{i+1}(A) - M^*_i(A)$. Below is an exhaustive case analysis:

Case 1: $\Delta_{i-1} < |M_i^{(\max)}(A)| - \theta$. Honest blocks on $\texttt{start}_i + \delta$ may disagree only on Δ_{i-1}. Therefore, in this case for every honest party P it holds that

$$|M_i^{(\max)}(A)| - |M_i^{(P)}(A)| \leq \Delta_{i-1} \ ,$$

implying that $|M_i^{(P)}(A)| > \theta$ for every honest P. Thus, the vote of every honestly generated block after the weak coin protocol completes is $\mathbf{sign}\left(M_i^{(\max)}(A)\right)$. Thus, for every non-abstaining honest P (of which there are, by definition, Z_i^{coin}) and in particular for $P_{i+1}^{\max}$:

$$|M_{i+1}^{(P_{i+1}^{\max})}(A)| \geq |M_{i+1}^{(P_i^{\max})}(A)| \geq |M_i^{(P_i^{\max})}(A)| + Z_i^{\text{coin}} - \Delta_i \ .$$

In this case

$$\begin{aligned} M_{i+1}^*(A) - M_i^*(A) &\geq Z_i^{\text{coin}} - \Delta_i - 2Y_i + \Delta_i + \Delta_{i-1} \\ &\geq Z_i^{\text{coin}} - 2Y_i \end{aligned}$$

Case 2: $|M_i^{(\max)}(A)| < \theta$. In this case, for every honest party P generating a block after the weak coin protocol completes it follows:

$$|M_i^{(P)}(A)| \leq |M_i^{(\max)}(A)| < \theta$$

that is every honest party will vote according to the result of the coin.

Case 2.1: (with probability at least p_{coin}): There is consensus on the weak coin for layer i and it agrees with $\mathbf{sign}\left(M_i^{(P_i^{\max})}(A)\right)$ so all honest parties will vote the same way on A. Thus,

$$|M_{i+1}^{(P_{i+1}^{\max})}(A)| \geq |M_{i+1}^{(P_i^{\max})}(A)| \geq |M_i^{(P_i^{\max})}(A)| + Z_i^{\text{coin}} - \Delta_i \ .$$

Therefore

$$|M_{i+1}^{(P_{i+1}^{\max})}(A)| - |M_i^{(P_i^{\max})}(A)| \geq Z_i^{\text{coin}} - \Delta_i \ .$$

In this case

$$\begin{aligned} M_{i+1}^*(A) - M_i^*(A) &\geq Z_i^{\text{coin}} - \Delta_i - 2Y_i + \Delta_i + \Delta_{i-1} \\ &\geq Z_i^{\text{coin}} - 2Y_i \end{aligned}$$

Case 2.2: (with probability at most $1 - p_{\text{coin}}$): The adversary selects $M_{i+1}^{(\max)}(A)$ arbitrarily. In this case

$$\begin{aligned} M_{i+1}^*(A) - M_i^*(A) &\geq -|M_i^{(\max)}(A)| - 2Y_i \\ &\geq -\theta - 2Y_i \end{aligned}$$

Case 3: $|M_i^{(\max)}(A)| - \theta \leq \Delta_{i-1} < 2\theta$ (note that, together with the negation of case 2, this implies $\theta \leq |M_i^{(\max)}(A)| < 3\theta$). In this case, every honest P must be on the "same side" of the $[\theta, -\theta]$ interval; that is, for every P it holds that

$$|M_i^{(P)}(A)| > \theta \Rightarrow \mathbf{sign}\left(M_i^{(P)}(A)\right) = \mathbf{sign}\left(M_i^{(\max)}(A)\right) \ .$$

Case 3.1: (with probability at least p_{coin}): There is consensus on the weak coin for layer i and it is equal to $\mathbf{sign}\left(M_i^{(\max)}(A)\right)$, so all non-abstaining honest parties will vote the same way on A. Thus,

$$\begin{aligned}|M_{i+1}^{(P_{i+1}^{\max})}(A)| &\geq |M_{i+1}^{(P_i^{\max})}(A)| \\ &\geq |M_i^{(P_i^{\max})}(A)| + Z_i^{\text{coin}} - \Delta_i\end{aligned}$$

In this case

$$\begin{aligned}M_{i+1}^*(A) - M_i^*(A) &\geq Z_i^{\text{coin}} - \Delta_i - 2Y_i + \Delta_i + \Delta_{i-1} \\ &\geq Z_i^{\text{coin}} - 2Y_i\end{aligned}$$

Case 3.2: (with probability at most $1 - p_{\text{coin}}$): The adversary selects $M_{i+1}^{(\max)}(A)$ arbitrarily. In this case

$$\begin{aligned}M_{i+1}^*(A) - M_i^*(A) &\geq -|M_i^{(\max)}(A)| - 2Y_i \\ &\geq -3\theta - 2Y_i\end{aligned}$$

Case 4: $\Delta_{i-1} \geq \max\left\{|M_i^{(\max)}(A)| - \theta, 2\theta\right\}$. The adversary chooses how each honest party will vote, so can fix $M_{i+1}^{(\max)}(A)$ arbitrarily. In this case,

$$\begin{aligned}M_{i+1}^*(A) - M_i^*(A) &\geq -|M_i^{(\max)}(A)| - 2Y_i + \Delta_i + \Delta_{i-1} \\ &\geq -|M_i^{(\max)}(A)| - 2Y_i + \max\left\{|M_i^{(\max)}(A)| - \theta, 2\theta\right\} \\ &= -\theta - 2Y_i\end{aligned}$$

For $j \in \{1, 2.1, 2.2, 3.1, 3.2, 4\}$, let C_j denote the set of rounds that fall under case j. To bound $M_i^*(A)$, we sum the $\left(M_{j+1}^*(A) - M_j^*(A)\right)$ differences from the layer in which block A claims to be (w.l.o.g set to 0) up to current layer i.:

$$\begin{aligned}M_i^*(A) &= \sum_{j=0}^{i} \left(M_{j+1}^*(A) - M_j^*(A)\right) \\ &\geq \sum_{j \in C_1} \left(Z_j^{\text{coin}} - 2Y_j\right) + \sum_{j \in C_{2.1}} \left(Z_j^{\text{coin}} - 2Y_j\right) + \sum_{j \in C_{2.2}} (-\theta - 2Y_j) \\ &\quad + \sum_{j \in C_{3.1}} \left(Z_j^{\text{coin}} - 2Y_j\right) + \sum_{j \in C_{3.2}} (-3\theta - 2Y_j) + \sum_{j \in C_4} (-\theta - 2Y_j)\end{aligned}$$

extracting $2Y_j$ from all the sums:

$$\begin{aligned}&= \sum_{j \in C_1} Z_j^{\text{coin}} + \sum_{j \in C_{2.1}} Z_j^{\text{coin}} + \sum_{j \in C_{2.2}} -\theta \\ &\quad + \sum_{j \in C_{3.1}} Z_j^{\text{coin}} + \sum_{j \in C_{3.2}} -3\theta + \sum_{j \in C_4} -\theta \\ &\quad - 2 \cdot \sum_{j=0}^{i} Y_j\end{aligned}$$

using our bounds on Z^{coin}:

$$\begin{aligned}&\geq (|C_1| + |C_{2.1}| + |C_{3.1}|) \cdot Z^*_\varepsilon\\ &\quad + (|C_{2.2}| + |C_{3.2}|) \cdot (-3\theta) + (|C_4|) \cdot (-\theta) - 2 \cdot \sum_{j=0}^{i} Y_j\end{aligned}$$

At this part, we wish to upper-bound the negative elements of the sum. Since $\sum_{j=0}^{i} Y_j$ is a sum of independent $Y_j \sim \mathit{Pois}(\lambda)$, then $\sum_{j=0}^{i} Y_j \sim \mathit{Pois}(i \cdot \lambda)$ and can be bounded using Poisson tail bounds s.t.

$$\Pr\left[\sum_{j=0}^{i} Y_j > (1+\varepsilon_Y) \cdot i \cdot q \cdot \frac{\ell^*_\varepsilon}{\ell} \cdot T_{\min}\right] \leq \left(2^{\frac{\varepsilon_Y - \ln(1+\varepsilon_Y)}{\ln 2}}\right)^{-i \cdot q \cdot \frac{\ell^*_\varepsilon}{\ell} \cdot T_{\min}}$$

The success of the weak coin protocol is independent for all layers (see Section 3.3). Therefore, either case 2.2 or case 3.2 occur on any layer with probability at most $1 - p_{\text{coin}}$. Using Chernoff, we bound the number of layers in which the weak coin protocol fails for all $\varepsilon_{p_{\text{coin}}} > 0$:

$$\Pr\left[|C_{2.2}| + |C_{3.2}| \geq i \cdot ((1 - p_{\text{coin}}) \cdot (1 + \varepsilon_{p_{\text{coin}}}))\right] \leq \exp\left(-\frac{{\varepsilon_{p_{\text{coin}}}}^2}{2 + \varepsilon_{p_{\text{coin}}}} \cdot (1 - p_{\text{coin}})\right).$$

As for case 4, in each round where it occurs the adversary "spends" at least 2θ blocks. Given $R_{i_0}(A)$, the adversary initial reserve and Y^*_ε, the bound for maliciously generated blocks in each round,

$$|C_4| \leq \left(R_{i_0}(A) + \sum_{j=0}^{i} Y_j\right) \cdot \frac{1}{2\theta}$$

Since $i = |C_1| + |C_{2.1}| + |C_{2.2}| + |C_{3.1}| + |C_{3.2}| + |C_4|$,

$$\begin{aligned}|C_1| + |C_{2.1}| + |C_{3.1}| &\geq i \cdot (1 + \varepsilon_{p_{\text{coin}}}) \cdot p_{\text{coin}} - \left(R_{i_0}(A) + \sum_{j=0}^{i} Y_j\right) \cdot \frac{1}{2\theta}\\ &\geq i \cdot p_{\text{coin}} - \left(R_{i_0}(A) + \sum_{j=0}^{i} Y_j\right) \cdot \frac{1}{2\theta}\end{aligned}$$

Putting it all together,

$$\begin{aligned}M^*_i(A) &\geq \left(i \cdot p_{\text{coin}} - \left(R_{i_0}(A) + \sum_{j=0}^{i} Y_j\right) \cdot \frac{1}{2\theta}\right) \cdot Z^*_\varepsilon - 2 \cdot \sum_{j=0}^{i} Y_j\\ &\geq \left(i \cdot p_{\text{coin}} - \frac{(Z^*_\varepsilon - 4\theta) \cdot \sum_{j=0}^{i} Y_j}{Z^*_\varepsilon \cdot 2\theta}\right) \cdot Z^*_\varepsilon - \frac{R_{i_0}(A) \cdot Z^*_\varepsilon}{2\theta}\\ &\geq \left(i \cdot p_{\text{coin}} - \frac{(Z^*_\varepsilon - 4\theta)}{Z^*_\varepsilon} \cdot \frac{\sum_{j=0}^{i} Y_j}{2\theta}\right) \cdot Z^*_\varepsilon - \frac{R_{i_0}(A) \cdot Z^*_\varepsilon}{2\theta}\end{aligned}$$

if $q < 1 - \frac{\ell}{\ell^-_\varepsilon - t^{\text{coin}}\delta} \cdot \frac{4\theta}{T_{\min}}$, except with negligible probability $Z^*_\varepsilon > 4\theta$:

$$\geq \left(i \cdot p_{\text{coin}} - \frac{\sum_{j=0}^{i} Y_j}{2\theta}\right) \cdot Z^*_\varepsilon - \frac{R_{i_0}(A) \cdot Z^*_\varepsilon}{2\theta}$$

with overwhelming probability $\sum_{j=0}^{i} Y_j \leq (1+\varepsilon_Y) \cdot i \cdot q \cdot \frac{\ell^*_\varepsilon}{\ell} \cdot T_{\min}$:

$$\geq i \cdot \left(p_{\text{coin}} - \frac{(1+\varepsilon_Y) \cdot qT_{\min}}{2\theta} \cdot \frac{\ell^*_\varepsilon}{\ell} \right) \cdot Z^*_\varepsilon - \frac{R_{i_0}(A) \cdot Z^*_\varepsilon}{2\theta}$$

Note that if

$$q \leq \left(\frac{2\theta}{T_{\min}} \cdot \frac{\ell}{\ell^*_\varepsilon} \cdot p_{\text{coin}} \right)$$

the expression in the parenthesis is a positive constant thus for any constant c

$$\exists i_c \; s.t. \; \forall i > i_c : M^*_i(A) > c \; .$$

5.3 Bounding the Actual Confirmation Margin

By fixing $c = \theta + k + 4Y^*_\varepsilon$, for any $i >= i_c$,

1. if $\Delta_{i-1} < 2Y^*_\varepsilon$ then

   $$M^\dagger_i(A) \geq M^*_i(A) + |R_i(j)| - 2\Delta_{i-1} \geq \theta + k$$

   and we are done;

2. otherwise, for every Δ_{i-1} s.t. $\Delta_{i-1} > 2Y^*_\varepsilon$, w.h.p. $|R_{i+1}(A)| < |R_i(A)|$ (adversary had to use blocks from her reserve).

As there are at most $|R_{i_c}(A)|$ consecutive rounds at which the last case applies, there must exists a layer $i^* > i_c$ s.t.

$$M^\dagger_{i^*}(A) \geq \theta + k \; .$$

5.4 Countermeasures against Freeloading

Per protocol 1, it is possible to give a small bonus to miners who reach the cap in the blocks that they creates. The bonus can be awarded by increasing the proportional amount (out of the total amount of transaction fees that were collected in the layer) that blocks that reach the cap receive, at the expense of blocks that are below the cap.

The rationale for the small bonus is due to possible freeloading behaviors:

- Without a bonus, rational miner may opt to forgo the work of collecting and verifying transactions. This concern has merit because verification of a large amount of transaction data is costly, and a single syntactically invalid transaction would cause the entire block to be invalidated.

- If the bonus is large, rational miners may duplicate transactions that previous blocks collected (albeit at the risk that an adversarial block included a poisonous invalid transaction on purpose, in order to invalidate other blocks in the layer).

As a middle ground, it is thus sensible to set the cap to a small amount relative to the average size of transaction data that is collected by an entire layer, and set the bonus to be small relative to the average block reward. This way, each individual miner will reach the cap by performing a relatively minor amount of computations, and the benefit in duplicating the transactions of prior blocks is less likely to be worth the risk of encountering a poisonous transaction. As our proposed reward allocation is smooth (cf. Section 8.4), transactions that already appeared in the previous layer should also count towards the cap, and hence miners are not disincentivized from creating blocks at the beginning of a layer.

For honest miners, the rules for selecting transactions to include in a block can involve both randomness and prioritization according to freshness. As a concrete example, let us consider the simple rule that picks transactions at random. Suppose that in a typical layer there are n blocks that accumulate m transactions in total. Let β denote the cap, and assume that our objective is that the n blocks will cover all of the m transactions. Let us simplify by assuming that the m transactions are available at the start of the layer (note that the required cap size will be significantly smaller with a rule that picks at random only among the transactions that have not yet been included in blocks). Since transactions are picked randomly, it is equivalent to assume that there are βn blocks such that each blocks collects a single transaction. Thus, this is the classical coupon collector's problem, implying that $\beta n < m(\ln(m) + 0.578)$ is enough for covering the m transactions in expectation. Additionally, a standard tail bound entails that $\Pr\left[\{\text{cover}\} > d \cdot m \ln(m)\right] \leq m^{d-1}$, hence $d = 2$ is already more than enough practice (since a layer that failed to cover all the available transactions is of no particular significance). To substitute concrete numbers: if $m = 10^6, n = 200, d = 2$, then a cap of $\beta \approx \frac{14}{100} \cdot 10^6$ will cover all the transactions of the layer with more than $1 - 10^{-6}$ probability, and a cap of $\beta \approx \frac{7}{100} \cdot 10^6$ will cover the m transactions in expectation. Note that $m = 10^6$ represents a transaction volume that is ≈ 500 greater than that of Bitcoin (cf. Section 3.7), i.e., it is the equivalent of a 500 megabytes (instead of 1 megabyte) Bitcoin block.

6 Byzantine-Agreement-Based Hare Protocol

Our first ABA-based hare protocol uses a binary byzantine agreement protocol that must satisfy some specific properties:

1. *Asynchronous Communication*: the protocol should allow for messages to be arbitrarily delayed (up to our communication latency bound)
2. *Name-independence*: The protocol should not rely on consistent names for parties. That is, it should allow each party to use its own labels for the other parties. Note that a party can identify where a message originated; it just might not agree with other parties on the originators name.

Formally, we will describe the ABA protocol as an interactive program P implementing the following interface:

- The party running P maintains an "event queue", initialized to $(\mathbf{init}, n)$.
- It repeats the following loop until P outputs a bit b and halts:
    1. Invoke P with the next event from the event queue (if there are no events, wait until one occurs).
    2. At every invocation of P, it may perform any of the following actions (possibly multiple actions sequentially):
        - Send a message to another party $j \in [n]$ (the parties are referenced by index).
        - Request a random coin bit i. In this case P will sleep until the bit i becomes available.
        - Output a bit b and halt (this should be the consensus bit)
    3. When a message m arrives from party j, add the event $(\mathbf{msg}, j, m)$ to the event queue (and invoke P if it was waiting)

6.1 The ABA Protocol's Environment

The ABA protocol may make the following assumptions about its environment:

- *Reliable, ordered, point-to-point channels.* For all $i \in [n]$, and every message m sent to P by party i (according to P's indexing), P it will (eventually) be invoked exactly once with $(\mathbf{msg}, i, m)$. Messages from a single party will be received in the order they were sent by that party. Moreover, only a message m sent by i will cause P to be invoked with $(\mathbf{msg}, i, m)$ (i.e., the network does not duplicate, create or reorder, modify or delete messages).

- *Weak common coin (with parameter p_{coin}).* The coin gives the following guarantees:

  1. No corrupt party will receive coin i until at least one honest party requested coin i.
  2. For all i, with probability at least p_{coin}, all honest parties will receive 0 in response to a request for coin i.
  3. For all i, with probability at least p_{coin}, all honest parties will receive 1 in response to a request for coin i.

6.1.1 Implementing the Environment.

We will choose m^* to be a bound on the number of blocks in a layer that satisfies, for every layer i, two properties:

1. The probability that more than m^* syntactically-valid layer-i blocks that were published before $\texttt{start}_i + \delta$ is negligible.
2. The probability that the number of honestly-generated blocks in layer i is less than $\frac{5}{6}m^*$ is negligible.

Note that if one of the two properties does not hold, the ABA protocol might fail, but the Tortoise protocol will still guarantee eventual consensus.

Our hare protocol implements the environment expected by the ABA protocol in the following way:

- *Participating parties.* For the hare protocol to work, every miner must add the following information to their blocks:

  - A verification key for a public-key signature scheme (this will be used to make our communication channels *reliable*).
  - (Optional) An address that allows direct point-to-point communication (e.g., an IP address). This can be used to make the point-to-point channels much more efficient. Alternatively, the parties can use the underlying gossip network to communicate.

  Every party that generated a block in layer i participates in the ABA protocol for layer i (in fact, the parties will run multiple ABA protocols in parallel—one for each block generated in layer i).

  Let party P be one of the parties that generated a block in layer i. At time $\texttt{start}_{i+1} + \delta$, P makes a list of all the (syntactically-valid) blocks it received in layer i. Let m be the actual number of blocks received. It orders them arbitrarily from 1 to m, then adds "placeholder" parties indexed $m+1, \ldots, m^*$. It then initializes an ABA protocol with m^* parties. If it receives messages from parties that were not in the list, it treats them as faulty parties (and ignores the messages).

- *Point-to-point messages.* If m is the k^{th} message P sent to party j (who has verification key vk_j), P signs the tuple (vk_P, vk_j, k, m) with P's signature key, and sends it to the address specified in j's block. When receiving a message of the form (vk_i, vk_P, k, m), P verifies that the signature is valid, and that vk_i appears in one of the blocks generated in block i. If there is some $k' < k$ such that a message of the form (vk_i, vk_P, k', m) has not yet been received (i.e., messages were reordered), the message is held until all previous messages have been delivered. Otherwise, this message is delivered to the queue (as the event $(\mathbf{msg}, j, m)$). Note the bound on network latency guarantees that all messages from honest parties will eventually be delivered.

- *Weak coin.* We can implement the weak coin as described in Section 3.3, based on the blockmesh itself. To satisfy the assumption that no corrupt party will receive coin i until at least one honest party requested coin i, we have to assume that the latency of communications in the ABA protocol is small enough that the coins are requested by honest parties before they actually become available (this is reasonable, since they can use actual point-to-point connections).

  Alternatively, we can use an off-chain coin based on unique signatures, such as the idea suggested by Micali [27]. Micali's construction gives a weak coin with $p_{\text{coin}} = 1/3$ that does not require proof-of-work; briefly, the idea is that to generate coin i, for all $j \in [m]$, party j will compute and publish $\sigma_j = \text{Sign}_{sk_j}(vk_j||i)$. Each party sorts $\{H(\sigma_j)\}_{j=1}^{m}$ for all valid messages, and takes the LSB of the first to be the coin value (this is similar to what we do with the proof-of-work-based coin). If $2m^*/3$ of the parties generating blocks are honest, there will be agreement on the value w.p. 2/3 (and hence $p_{\text{coin}} = 1/3$).

Because the ABA protocol does not rely on consistent naming, the fact that different parties see a different set of blocks will not matter—the honestly generated blocks will be in the intersection of all the honest parties' lists, and so every honest party will be considered a participant by every other honest party. By our assumptions about m^*, there are more than $2m^*/3$, so even if we count the parties $m+1, \ldots, m^*$ as faulty the honest parties are still a 2/3 majority.

6.2 Using the ABA Protocol

We require the following guarantees from the ABA protocol:

- *Consensus*: All honest parties should agree on a bit b for every execution of the ABA protocol. If all honest parties have the same input b', then $b = b'$.
- *Termination*: The ABA protocol should terminate in a small (constant) number of rounds (we count rounds as the number of weak coin flips) except with negligible probability.

Our ABA-hare protocol executes an instance of the ABA protocol for every block generated in layer i.

Block Propagation. For party P, denote $G = (vk_1, \ldots, vk_m)$ the ids of the parties(blocks) that P received before $\texttt{start}_i + \delta$.

```
Initialize $S_P = \{vk_1, \ldots, vk_m\}$ // the set of blocks to validate
for all $j \in [m]$ do // Update round 1
    Send $S_P$ to $j$
    Receive $S_j$ from $j$
    Update $S_P \leftarrow S_P \cup S_j$
end for
```

ABA Execution.

```
for all $vk_j \in S_P$ do // In parallel
    if $vk_j \in G$ then // We received the block before $\texttt{start}_i + \delta$
        $b' = 1$
    else
```

```
            $b' = 0$
        end if
        Execute the ABA protocol with input $b'$. Let $b$ be the output of the protocol.
        if $b = 1$ then  // Consensus on the block being included
            Sign $vk_j$ using $sk_P$ and publish the signature to the blockmesh.
        end if
    end for
```

ABA Hare Protocol Consensus. The hare protocol specifies that a block in layer i is valid iff it has $\frac{5}{6}m^*$ signatures from parties that generated a layer-i block.

Claim 6.1. *Every honestly-generated block is considered valid by the ABA hare protocol.*

Proof. Every honestly-generated block will be received by every honest party before $\texttt{start}_i + \delta$. Thus, all honest parties will participate in the ABA for confirmation of this block, and they will all agree that the block is valid. By the consensus guarantee of the ABA protocol, the consensus outcome will be that the block is valid, so all honest parties will sign it. Since the number of honest parties is more than $\frac{5}{6}m^*$, the hare protocol will consider this block valid. □

Claim 6.2. *Every block that was not received by any honest party before* $\texttt{start}_i + \delta$ *will be considered invalid by the ABA hare protocol.*

Proof. No honest party will initiate an ABA protocol to validate such blocks. However, the malicious parties may send such a block to a subset of the honest parties in the Block Propagation phase. If the block is sent to less than $\frac{2}{3}m^*$ honest parties, even the outcome of the consensus is that the block is valid, there will not be enough signatures to make it valid (since the malicious parties have at less than $\frac{1}{6}m^*$ blocks). If the block is sent to more than $\frac{2}{3}m^*$ honest parties, the honest parties will have a 2/3 majority in the ABA protocol and they all agree that the block is invalid, hence the consensus result will be that the block is invalid. □

Lemma 6.3. *The hare protocol will reach consensus on every block.*

Proof. By Claim 6.2, there will be consensus on every block that was not received by any honest party before $\texttt{start}_i + \delta$. If a block was received by at least one honest party, then all honest parties will receive the block in the Block Propagation phase, so they will all participate in the ABA protocol for that block. Since there is a 2/3 majority of honest blocks, the ABA protocol will reach consensus on that block. If the consensus is valid, all honest parties will sign the block, so it will be considered valid by the hare protocol. Otherwise, no honest party will sign the block, so it will be considered invalid by the hare protocol. □

6.3 Improving the Hare-Protocol Parameters using Multivalued ABA

In order to guarantee consensus using the binary ABA protocol, we need the number of honestly-generated blocks to be at least $\frac{5}{6}$ of the maximum number of blocks generated in a layer interval. This implies that the fraction of the hashpower controlled by the adversary cannot be more than $q < 1/6$. If a 1/2 majority (rather than 2/3) suffices for the ABA protocol, we can relax the assumption to $q < 1/4$ (this might be possible with cryptographic assumptions). However, if we are willing to use a more "heavy-duty" protocol that guarantees consensus on strings rather than bits, we can use it to improve parameters. Due to space restrictions, we describe the detailed algorithm in Section 7.

6.4 Additional Potential Improvements

Our ABA and Multivalued-ABA Hare protocols solve the "participant consensus" problem (i.e., agreeing on who is participating in the protocols) by using a preset bound m^* on the number of blocks in a layer. However, the adversary can influence the number of blocks in a layer (e.g., by withholding generated blocks until honest parties already generate the minimum number). This leads to restrictions on the adversary's hashpower that are not optimal (e.g., we cannot support an adversary with 1/3 the hashpower).

However, it may be possible to use the underlying ABA protocols more efficiently: instead of using a bound on the number of parties, each honest party will just run the ABA protocol with the parties it sees. Note that if all the honest parties see the same *number* of blocks, even if disagree on which blocks, this will already work with our existing analysis. The reason is that all honest parties see each other—the only blocks they can disagree on are the malicious blocks—but the ABA protocols already allow arbitrary malicious behaviour for corrupt parties, so we can treat two different malicious parties as a single party sending different messages to the honest parties.

Unfortunately, the honest parties may not agree on the number of blocks. However, for specific ABA protocols this can still be secure. In particular, a close look at the analysis of the ABA protocols of Mostéfaoui et al. [29, 30] shows that they use the total number of parties for achieving properties such as "if a message m was received from $t + 1$ different parties, it must have been sent by at least one honest party" and "if a message was received from $n - 2$ different parties, it must have been sent by at least $t + 1$ honest parties"—these statements still hold in our setting even when there is disagreement on n and t (as long all the honest parties see each other, and in the local execution of each honest party there is a 2/3 honest majority).

7 Improving the Hare-Protocol Parameters using Multivalued ABA: Details

We will require a multivalued ABA protocol with the same properties as the binary ABA protocol (asynchronous, name-independent), that is also *intrusion-tolerant*. That is, it guarantees that if all honest parties have v as their input, the consensus value output by honest parties at the end of the protocol must be v. We note that Mostéfaoui and Raynal show how to construct a multivalued ABA protocol with the desired propeties based on any binary ABA protocol [30]. Their multivalued protocol has a constant number of rounds and is secure when less than 1/3 of the parties are malicious.

Let ABA(v) denote an execution of the multivalued ABA protocol with input v, and BBA(b) denote an execution of the binary ABA with input b.

For party P, denote $G = (vk_1, \ldots, vk_m)$ the ids of the parties(blocks) that P received before $\texttt{start}_i + \delta$. Our Multivalued ABA Hare protocol can be described in two parts. A main loop, that runs until agreement is reached to terminate, and a message-processing procedure that handles incoming messages (the protocol is asynchronous, so does not wait for messages to arrive unless such a message is guaranteed to arrive). Think of the process as single-threaded (i.e., if a message arrives, it finishes executing the current line in the main loop, executes the ProcessMessage procedure and then returns to the next line in the main loop).

```
procedure MainLoop
    Initialize $S_P \leftarrow G$, sorted lexicographically
    Initialize $Valid \leftarrow \emptyset$.
    loop // Main Loop
        Let $vk$ be the first (smallest) element in $S_P$
        Let $vk^* \leftarrow \mathrm{ABA}(vk)$
        if $vk^* \neq \bot$ then
```

```
 8:             Let $Valid \leftarrow Valid \cup \{vk^*\}$  // Add $vk^*$ to valid block set
 9:             $S_P = S_P \setminus \{vk^*\}$  // Remove $vk^*$ from $S_P$
10:             **if** $vk < vk^*$ **then**
11:                 Send $vk$ to all parties
12:             **end if**
13:         **else** // Some party proposed to terminate
14:             Set $b = 1$ iff $S_P \neq \emptyset$.
15:             Let $b^* \leftarrow \mathrm{BBA}(b)$
16:             **if** $b^* = 0$ **then**  // At least one honest party agreed to terminate
17:                 **return** $Valid$ and terminate
18:             **else** // At least one honest party does not want to terminate
19:                 **if** $S_P \neq \emptyset$ **then**
20:                     Send the first element in $S_P$ to all parties
21:                 **else**
22:                     Wait until $S_P \neq \emptyset$
23:                 **end if**
24:             **end if**
25:         **end if**
26:     **end loop**
27: **end procedure**
28: **procedure** ProcessMessage($vk$)
    // Called whenever a message $vk$ is received if the party has not yet terminated
29:     **if** $vk \notin Valid$ **then**
30:         Set $S_P = S_P \cup vk$
31:     **end if**
32: **end procedure**
```

Lemma 7.1. *An honest party running the protocol will always terminate*

Proof. Let $T_r = bigcup_i S_i$ be the set of blocks in the union all honest parties' sets S_i at the start of iteration r of the main loop . Let Q_r be the set of blocks that are *not* in $I \cup Valid$ at the start of iteration r of the main loop. Note that $|Q_1| \leq m^*/3$ (since all honest blocks are in I_1).

Consider iteration r of the main loop for an honest party P.

Case 1: The outcome of the ABA protocol was vk. Then vk is added to $Valid$, so $|T_{r+1} \cup Q_{r+1}| < |T_r \cup Q_r|$.

Case 2: The outcome of the ABA protocol was $\perp$

Case 2.1: The outcome of the BBA protocol was 0. In this case the party terminates and we are done.

Case 2.2: The outcome of the BBA protocol was 1. In this case for at least one honest party $S_i \neq \emptyset$ (otherwise all honest parties would have input 0 to the BBA protocol). Let vk^* be the minimal element in T_r. In this case, vk^* will be the minimal for any honest party that has it, Since each honest party for which $S_P \neq \emptyset$ sends its minimal element, it will be sent in Line 20 by some honest party. Since all messages sent by honest parties are eventually received by all honest parties, it will eventually be received by all honest parties. Let r' be the first round at which it was received by all honest parties.

Case 2.2.1: If $vk^* \in Valid_{r'}$ then $|T_{r'} \cup Q_{r'}| < |T_r \cup Q_r|$

Case 2.2.2: Otherwise, if there exists $vk^\dagger \in T_{r'}$ such that and $vk^\dagger < v^*$, then $Q_{r'} < Q_r$ (since $vk^\dagger$ could only have come from Q_r).

Case 2.2.3: Otherwise, vk^* must be minimal in $T_{r'}$, so all honest parties will use it as input to the ABA in call. By the intrusion-tolerance property of the ABA, the output must be $vk^* \neq \bot$, hence $|T_{r'+1} \cup Q_{r'+1}| < |T_{r'} \cup Q_{r'}|$

We showed that in every iteration, either $|T_r \cup Q_r|$ decreases, or $|Q_r|$ decreases, or there will be a future round r' in which one of those occurs. Since they are both non-increasing, eventually $|T_r \cup Q_r|$ must reach 0. When that happens, every honest party will have $S_i = \emptyset$, so all honest parties will use input $\bot$ to the ABA, guaranteeing the output will be $\bot$. All honest parties will then use input 0 to the BBA, guaranteeing output 0, so they will all terminate. □

Claim 7.2. *All honest parties will terminate in the same iteration of the main loop.*

Proof. The parties terminate only if the ABA protocol returns $\bot$ and then the BBA protocol returns 0. Since all honest parties agree on the outputs of the ABA and BBA protocols, they will all terminate at the same iteration. □

Lemma 7.3. *Every honest party will agree on the value of $Valid$ after termination.*

Proof. The only blocks added to $Valid$ are output from the ABA protocol. Since all honest parties agree on the outputs of the ABA protocol and terminate at the same time, they will agree on the sequence of blocks added. □

Lemma 7.4. *Every honestly-generated block will be included in $Valid$*

Proof. All Honestly-generated blocks appear in $I = bigcap_i G_i$, the intersection of all honest parties' "good sets", and hence appear in the intersection of the sets S_i. Blocks are removed from the set S_i only if they enter $Valid$. For the protocol to terminate, at least one honest party must have an empty S_i, therefore all honest blocks must have been added to $Valid$. □

8 Incentive-Compatibility of Race-Free Protocols

8.1 Generalized Blockchain Mining Games

In this section, we extend the *Blockchain Mining Game* described in [15] to allow for analysis of protocols based on blockDAGs in addition to blockchains. The *Generalized Blockchain Mining Game* is a stochastic game of complete information played among n players called miners.

Public State. The public state of the game is a layered DAG rather than a rooted tree as in [15]. A predefined initial set of nodes in the DAG is called the *root*. The nodes of the DAG represent mined blocks and each node is labeled by a miner who mined it and a number of layer it belongs to.

Private State. Similar to the public state but may contain more blocks (*private blocks*). The private and public states must share the same root. Since the game is played with complete information, the private state of all miners is visible. However, a player cannot mine on blocks not contained in its private state.

Stages. The game proceeds in stages. At each stage, some player mines a block, i.e., a new block is created such that its label is i with probability p_i (it holds that $\sum_{i=1}^{n} p_i = 1$ and each p_i corresponds to the computational power of player i). The miners then decide whether they wish to release any subset of their private states into the public state.

Strategies. The strategy of player i consists of two functions (μ_i, ρ_i):

1. *Mining function* μ_i: selects a subset of the private state to mine on and a layer number j. If the player succeeds in mining a block then the newly mined block has directed edges to all the blocks selected by μ_i and is labeled by layer j.
2. *Release function* ρ_i: selects when to move parts of the private state to the public state.

Block Validity. The validity is decided by a function f of the state that outputs the public state restricted to accepted blocks.

Utility. The utility of each miner is the fraction of its valid blocks in the blockDAG.

Immediate vs. Strategic Release. Similarly to [15], we consider the *immediate release* model in which the players release their newly mined blocks into the public state right after mining them (i.e., the release function is fixed for all the players). We also consider the *strategic release* model, where the parties can decide on when their blocks become part of the public state.

8.2 Race-Free Games

In this section, we introduce the game-theoretic definition of *race-free strategy profiles and games* and show how race-freeness relates to incentive compatibility in blockchain mining games.

Definition 8.1 ((ε, q)-Race-free strategy profile)**.** Let $\hat{\sigma} = (\hat{\sigma}_1, \ldots, \hat{\sigma}_n)$ be a strategy profile in a blockchain mining game. We say that $\hat{\sigma}$ is *(ε, q)-race-free* if for every player i, and for every $C \subseteq [n] \setminus \{i\}$ such that $\sum_{j \in C} p_j \leq q$, it holds that

$$\forall \sigma_C : u_i(\hat{\sigma}_{[n] \setminus C}, \sigma_C) \geq u_i(\hat{\sigma}) \cdot (1 - \varepsilon) \ .$$

That is, the strategy $\hat{\sigma}_i$ of player i guarantees, up to a multiplicative factor, at least the payoff achieved in $\hat{\sigma}$ irrespective of the actions of any coalition of other players controlling at most a q-fraction of computational power. We say that a strategy profile is *q-race-free* if it is (ε, q)-race-free for $\varepsilon = 0$.

Definition 8.2 (Race-free game)**.** A blockchain mining game is *race-free* if it has a race-free strategy profile.

8.2.1 Properties of Race-Free Games.

We prove that following a race-free strategy is not only a low-risk behavior but it is also incentive-compatible. In particular, any (ε, q)-race-free strategy is also an ε-Nash equilibrium equilibrium if every player controls at most a q-fraction of the computational power.

Theorem 8.3. *For all $\varepsilon \geq 0$. If $\hat{\sigma}$ is an (ε, q)-race-free strategy profile in a blockchain mining game and $p_i \leq q$ for every i then $\hat{\sigma}$ is an ε-Nash equilibrium.*

Proof. Let σ_i be any strategy of player $i \in [n]$. Since $p_i \leq q$, it follows from the (ε, q)-race-free property of $\hat{\sigma}$ and the fact that any blockchain mining game is a constant-sum game that:

$$\begin{aligned} u_i(\sigma_i, \hat{\sigma}_{-i}) &\leq 1 - \sum_{j \in [n] \setminus \{i\}} u_j(\hat{\sigma})(1-\varepsilon) \\ &= \sum_{k \in [n]} u_k(\hat{\sigma}) - \sum_{j \in [n] \setminus \{i\}} u_j(\hat{\sigma})(1-\varepsilon) \\ &= u_i(\hat{\sigma}) + \sum_{j \in [n] \setminus \{i\}} u_j(\hat{\sigma})\varepsilon \\ &\leq u_i(\hat{\sigma}) + \varepsilon \ . \end{aligned}$$

Where the first and last inequalities follow, since all utilities in the game are non-negative and their sum is equal to 1. □

8.2.2 Race-Free Blockchain Protocols.

We now formally define the *race-free* property of a blockchain protocol. Our definition is based on the blockchain mining game as described in Section 8.1.

Definition 8.4 (ε-Race-free blockchain protocols)**.** A blockchain protocol is *ε-race-free* if in the corresponding blockchain mining game, the strategy profile in which all the players follow the protocol is ε-race-free. A blockchain protocol is *race-free* if it is an ε-race-free blockchain protocol for $\varepsilon = 0$.

8.3 Meshcash is Race-Free

We show that in the setting with constant rewards per block, the Meshcash protocol is race-free. We need to specify the blockchain mining game corresponding to Meshcash, and in particular the validity function. Recall, that a block in layer j is considered valid if it has at least $T_{\min}$ outgoing edges to blocks in layer j. The utility function is then simply the fraction of player's valid blocks in the valid blockDAG.

Immediate Release Model. In the immediate release model, the players' strategies are restricted to publishing newly mined blocks at the stage of their creation. The players can only select the subset of the public state to mine on. Any block of player i in layer j must have at least $T_{\min}$ outgoing edges to blocks in layer $j-1$ for it to be considered valid. Hence, rational players are restricted to strategies that correspond to the honest behavior. Note that in the model of blockchain mining games, there are no delays and all the players see published blocks immediately, and the players are in consensus about the contents of the layers of the public state at every stage. Also, no coalition of players can invalidate blocks of any player by not pointing to them. Thus, the honest strategy profile is race-free.

Although the immediate release model is rather restrictive, we stress that it allows to capture interesting strategies of players. In particular, Kiayias et al. [15] demonstrated that for players controlling at least 41.8% of the computational power in the system, the honest behavior in Bitcoin is not incentive-compatible even with immediate release (cf. Section 3.2 in [15]).

Strategic Release Model. In the strategic release model, the players can withhold their blocks from the public state. This allows them to create valid blocks in layers ahead of the layer corresponding to the contents of the public state (in case player's private state contains at least $T_{\min}$ blocks in the current layer).

Proposition 8.5. *Meshcash is a race-free protocol with respect to coalitions controlling up to 1/3 of the computational power.*

Proof. It follows from the future self-consistence of the protocol that the blocks of every honest party cannot be with overwhelming probability invalidated by an adversary controlling up to 1/3 of the computational power. Thus, the fraction of the valid blocks of any honest party cannot be decreased and the utility remains at least the same irrespective of the actions of the other players (controlling at most 1/3 of the computational power). □

8.4 When Fees Dominate Coinbase

In the previous section, we used the model of [15] to analyze the properties of Meshcash in the setting where the coinbase reward dominates over the reward gained from the transaction fees. However, the major strategic considerations in Meshcash seem to be in place when the system reaches the steady-state and no new coins are created, i.e., when the real incentive for miners comes from the transaction fees.

For example in Bitcoin, when the miners can include transactions in their blocks depending on the fees, it is possible to mount the following "bribery" attack. When the last published block contains many transactions with high fees, an attacker can try to extend the second last block instead of the last one. If he succeeds in mining a block and includes in it only a few of the transactions with high fees, then he creates a situation in which it is advantageous for the rest of the network to mine on his block. In particular, the other parties can now include the remaining transactions with heavy fees that would have otherwise been already claimed by the original block. The adversary "bribes" the rest of the network to prioritize his block by splitting some of the transaction fees with them.

Reward Allocation. We suggest a reward allocation scheme where the transaction fees collected in layer i are divided among the last k layers (i.e., layers $i-k,\ldots,i$) proportionally to the actual number of blocks in each layer. Formally, let π_i denote the transaction fees collected in layer i, and for $m \leq n$, let $\chi_{m,n}$ be the total number of blocks mined in layers $m,\ldots,n$. The reward for block A block in layer i is:

$$r_i = \text{coinbase}_i + c_A^* \cdot \sum_{j=i}^{i+k} \frac{\pi_j}{\chi_{j-k,j}} \ ,$$

where $c_A^* \in [0,1]$ is the fraction of included transactions in A compared to the adjustable transaction cap.

When $\text{coinbase}_i = 0$, a player might have an incentive to deviate from the protocol specifications in order to increase the amount of transaction fees split among layers containing his blocks. We discuss in particular the following *Temporal Robin Hood attack.*

The "Temporal Robin Hood" attack. The name of the attack comes from the fact that an adversary can "steal" transaction fees from a future layer and split them among the blocks in the past layers. Assuming a constant rate of fees over time, on each layer i, the attacker continues mining on the layer until time $\texttt{start}_{i+1} + \delta$. By doing so, a miner risks not finding a block but if he does, he manages to steal rewards from the next layer and splits the rewards between the past layers. Note that the expected stolen reward is at most a $\frac{\delta}{k \cdot \ell}$-fraction of fees accumulated over the expected layer time ℓ.

References

[1] I. Abraham, D. Malkhi, K. Nayak, L. Ren, and A. Spiegelman, "Solidus: An incentive-compatible cryptocurrency based on permissionless byzantine consensus," 2016. [Online]. Available: http://arxiv.org/abs/1612.02916

[2] M. Babaioff, S. Dobzinski, S. Oren, and A. Zohar, “On Bitcoin and red balloons,” in *ACM Conference on Electronic Commerce*, 2012, pp. 56–73.

[3] B. Barak, R. Canetti, Y. Lindell, R. Pass, and T. Rabin, “Secure computation without authentication,” in *CRYPTO*, 2005.

[4] J. Bonneau, “Why buy when you can rent? - bribery attacks on bitcoin-style consensus,” in *Financial Cryptography Bitcoin Workshop*, 2016.

[5] X. Boyen, C. Carr, and T. Haines, “Blockchain-free cryptocurrencies. a rational framework for truly decentralised fast transactions,” Cryptology ePrint Archive, Report 2016/871, 2016, http://eprint.iacr.org/2016/871.

[6] M. Castro and B. Liskov, “Practical byzantine fault tolerance,” *OSDI*, 1999.

[7] K. Croman, C. Decker, I. Eyal, A. E. Gencer, A. Juels, A. Kosba, A. Miller, P. Saxena, E. Shi, E. G. Sirer, D. Song, and R. Wattenhofer, “On scaling decentralized blockchains,” in *Financial Cryptography 3rd Bitcoin Workshop*, 2016.

[8] C. Decker and R. Wattenhofer, “Information Propagation in the Bitcoin Network,” in *13th IEEE International Conference on Peer-to-Peer Computing (P2P)*, 2013.

[9] I. Eyal, “The miner’s dilemma.” in *IEEE S&P*, 2015.

[10] I. Eyal, A. E. Gencer, E. G. Sirer, and R. van Renesse, “Bitcoin-NG: A scalable blockchain protocol,” in *NSDI*, 2016.

[11] I. Eyal and E. Sirer, “Majority is not enough: Bitcoin mining is vulnerable.” in *Financial Cryptography*, 2014.

[12] J. Garay, A. Kiayias, and N. Leonardos, “The Bitcoin backbone protocol: Analysis and applications,” in *Eurocrypt*, 2015, http://eprint.iacr.org/2014/765.

[13] A. Gervais, H. Ritzdorf, G. O. Karame, and S. Capkun, “Tampering with the delivery of blocks and transactions in bitcoin,” in *22nd ACM CCS*, 2015.

[14] E. Heilman, A. Kendler, A. Zohar, and S. Goldberg, “Eclipse attacks on bitcoin’s peer-to-peer network,” in *24th USENIX Security*, 2015.

[15] A. Kiayias, E. Koutsoupias, M. Kyropoulou, and Y. Tselekounis, “Blockchain mining games,” in *Proceedings of the 2016 ACM Conference on Economics and Computation, EC ’16, Maastricht, The Netherlands, July 24-28, 2016*, 2016, pp. 365–382.

[16] A. Kiayias and G. Panagiotakos, “On trees, chains and fast transactions in the blockchain,” p. 545, 2016. [Online]. Available: http://eprint.iacr.org/2016/545

[17] E. Kokoris-Kogias, P. Jovanovic, N. Gailly, I. Khoffi, L. Gasser, and B. Ford, “Enhancing bitcoin security and performance with strong consistency via collective signing,” in *USENIX Security Symposium*, 2016.

[18] S. D. Lerner, “Dagcoin: a cryptocurrency without blocks,” 2015, https://bitslog.wordpress.com/2015/09/11/dagcoin/.

[19] Y. Lewenberg, Y. Bachrach, Y. Sompolinsky, A. Zohar, and J. S. Rosenschein, “Bitcoin mining pools: A cooperative game theoretic analysis,” in *International Conference on Autonomous Agents and Multiagent Systems*, 2015.

[20] Y. Lewenberg, Y. Sompolinsky, and A. Zohar, “Inclusive block chain protocols,” in *Financial Cryptography and Data Security*, 2015, pp. 528–547.

[21] K. Liao and J. Katz, “Incentivizing double-spend collusion in bitcoin,” in *Financial Cryptography Bitcoin Workshop*, 2017.

[22] L. Luu, V. Narayanan, K. Baweja, C. Zheng, S. Gilbert, and P. Saxena, “SCP: A computationally-scalable byzantine consensus protocol for blockchains,” in *CCS*, 2016.

[23] L. Luu, J. Teutsch, R. Kulkarni, and P. Saxena, “Demystifying incentives in the consensus computer,” in *22nd ACM CCS*, 2015.

[24] “Maged” (pseudonym), “Re: Unfreezable blockchain,” 2012, https://bitcointalk.org/index.php?topic=57647.msg686497#msg686497.

[25] G. Maxwell, 2015, https://bitcointalk.org/index.php?topic=1108304.msg11786046#msg11786046.

[26] B. McElrath, “Braiding the blockchain,” 2015, https://scalingbitcoin.org/hongkong2015/presentations/DAY2/2_breaking_the_chain_1_mcelrath.pdf.

[27] S. Micali, “Byzantine agreement made trivial,” 2016, https://people.csail.mit.edu/silvio/Selected%20Scientific%20Papers/Distributed%20Computation/BYZANTINE%20AGREEMENT%20MADE%20TRIVIAL.pdf.

[28] A. Miller, A. E. Kosba, J. Katz, and E. Shi, “Nonoutsourceable scratch-off puzzles to discourage bitcoin mining coalitions,” in *22nd ACM CCS*, 2015.

[29] A. Mostéfaoui, H. Moumen, and M. Raynal, “Signature-free asynchronous binary byzantine consensus with $t < n/3$, $O(n^2)$ messages, and $O(1)$ expected time,” *J. ACM*, vol. 62, no. 4, p. 31, 2015. [Online]. Available: https://hal.archives-ouvertes.fr/hal-01176110/file/JACM.pdf

[30] A. Mostéfaoui and M. Raynal, “Signature-free asynchronous byzantine systems: From multivalued to binary consensus with t < n/3, o(n2) messages, and constant time,” in *Structural Information and Communication Complexity - 22nd International Colloquium, SIROCCO 2015, Montserrat, Spain, July 14-16, 2015, Post-Proceedings*, 2015, pp. 194–208. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-25258-2_14

[31] S. Nakamoto, “Bitcoin: A peer-to-peer electronic cash system,” *Bitcoin.org*, 2008. [Online]. Available: http://www.bitcoin.org/bitcoin.pdf

[32] R. Pass, L. Seeman, and abhi shelat, “Analysis of the blockchain protocol in asynchronous networks,” Eurocrypt, 2017, http://eprint.iacr.org/2016/454.

[33] R. Pass and E. Shi, “Fruitchains: A fair blockchain,” Cryptology ePrint Archive, Report 2016/916, 2016, http://eprint.iacr.org/2016/916.

[34] ——, “Hybrid consensus: Efficient consensus in the permissionless model,” Cryptology ePrint Archive, Report 2016/917, 2016, http://eprint.iacr.org/2016/917.

[35] J. Poon and T. Dryja, “The bitcoin lightning network,” https://lightning.network/lightning-network-paper.pdf.

[36] P. Rizun, “A transaction fee market exists without a block size limit,” 2015, https://scalingbitcoin.org/papers/feemarket.pdf.

[37] M. Rosenfeld, "Analysis of hashrate-based double spending," http://arxiv.org/abs/1402.2009, 2014.

[38] A. Sapirshtein, Y. Sompolinsky, and A. Zohar, "Optimal selfish mining strategies in Bitcoin," in *Financial Cryptography*, 2016.

[39] Y. Sompolinsky, Y. Lewenberg, and A. Zohar, "Spectre: A fast and scalable cryptocurrency protocol," 2016, https://eprint.iacr.org/2016/1159.

[40] Y. Sompolinsky and A. Zohar, "Secure high-rate transaction processing in bitcoin," in *19th Financial Cryptography and Data Security*, 2015.

[41] P. Wuille, "Mining centralization pressure from non-uniform propagation speed," 2015, http://www.mail-archive.com/bitcoin-development@lists.sourceforge.net/msg08161.html.

A Nonlinear Rewards

In a blockchain based cryptocurrency of the Bitcoin mold, scaling the system to support a larger amount of transactions per second can be done in two ways:

1. Increasing the maximal block size, meaning that the blocks are allowed to contain more transactions.

2. Decreasing the expected interval between blocks, i.e., allowing blocks to be solved more frequently by having the network re-adjust the PoW difficulty according to a lower difficulty target value.

Both of these methods imply that the network propagation latency becomes greater relative to the expected interval between blocks. With the first method, this would be the case because larger blocks take longer time to propagate. With the second method, the propagation latency remains unaffected but blocks are solved at shorter intervals.

However, there exists a conflict between a *decentralized* blockchain-based cryptocurrency, and a system that should scale to support an increasing rate of transactions per second. A scalable system is useful because growth in the level of popularity of the cryptocurrency implies that more commerce takes places as a result. This conflict manifests itself in what we refer to as the "nonlinearity of rewards" problem, meaning that if one miner has x fraction of the total hashrate and another miner has $y < x$ fraction of the total hashrate, then the amount of rewards that the first miner obtains is expected to be larger than that of second miner by a factor that is strictly greater than $\frac{x}{y}$. At the intuitive level, the nonlinearity phenomenon occurs because a miner who solved a block at the current moment gets a head-start over the rest of the miners, because she is aware of the longest chain and can start to extend it immediately, while the other miners have to wait until the longest chain propagates to them. Since this event happens more frequently to miners who have larger amounts of the hashrate, the overall result is that block rewards are distributed in a nonlinear manner. The problem becomes more amplified either when the propagation latency is greater, or when the interval between blocks is shorter.

To describe the nonlinearity of rewards phenomenon in a more precise fashion, let us consider a simple model with two *honest* miners M_1, M_2 who possess p and $q = 1-p$ of the mining power.

At the starting point, the longest chain that M_1 is aware of consists of X blocks, and the longest chain that M_2 is aware of consists of Y blocks. Both M_1 and M_2 try to extend that longest chain that they are aware of, and whenever each of them succeeds in extending her chain she transmits the extended chain to the other miner. When a miner receives a chain that is strictly longer than the chain that she was aware of, she switches to this received chain and tries to extend only that chain from now on.

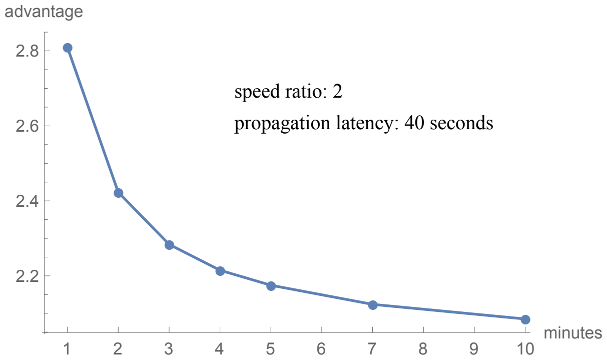

Figure A.1: Effective rewards as a function of the blocks interval and network latency.

Denote by L the average time until a new block is solved by either M_1 or M_2 at the current difficulty level. Denote by δ the network propagation latency, meaning that it takes δ time to transmit a chain from M_1 to M_2 and vice versa. Suppose that M_1 successfully solves her next block at time t. Now,

- If $X \geq Y$ then M_1 will switch only if M_2 solved $X - Y + 1$ blocks between time 0 and time $t - \delta$.
- If $Y > X$ then M_1 will switch if $t > \delta$, or $t \leq \delta$ and no more than $Y - X - 1$ blocks were solved by M_2 between time $-\delta$ and $-\delta + t$, as otherwise M_2 did not transmit a chain of length at least $X + 1$ on time.

The solved blocks of M_2 are given according to exponential distribution with parameter $\lambda = q/L$. In a symmetric fashion, M_2 will switch to a longer chain that M_1 created according to the same conditions, except that the solved blocks of M_1 are generated according to exponential distribution with parameter $\lambda = p/L$.

When a miner switches to a longer chain that was transmitted to her at greater frequencies, this in a sense represents a bigger loss of her share of the rewards, because a larger portion of the work that she does goes to waste.

We ran computer simulations according to the above conditions, and the results are summarized in Fig. A.1. The parameters in our simulations are $p = \frac{1}{3}, q = \frac{2}{3}, \delta = 40$ seconds (in accord with [8, 7]), and a variable L. In particular, for $L = 10$ minutes we get that the amount of block rewards that M_2 obtains is larger than that of M_1 by a factor of 2.085, for $L = 3$ minutes this factor is 2.285, and for $L = 1$ minute this factor is 2.81.

Let us note that similar results were obtained in [19], and experiments in a more comprehensive model that deals with various coalitions of miners and even takes computational superiority into account were done in [41].

Of course, the simplified two-miners model serves only as a demonstration for the basis of the tension. In the real world, the unfair rewards problem is likely to manifest itself in the form of small miners who would be hesitant to include a large amount of transactions in the blocks that they prepare, as it lowers their probability to win the race to create the next block [36].

The implications of the nonlinearity problem can be quite severe. Miners who posses a relatively small fraction of total hashrate may become less inclined to continue their participation, which would diminish the decentralized nature of the cryptocurrency. Furthermore, each individual miner may wish to run his mining equipment in a geographical location that is closer to large concentrations of the total hashrate, which would make it easier for an adversary to mount network isolation attacks [13, 14].

In Bitcoin, the expected interval between blocks is 10 minutes, and a network propagation latency of $\delta = 40$ seconds can be considered to be a reasonable estimate [8, 7]. For now, the block size limit of Bitcoin is 1 megabyte, implying that the Bitcoin economy currently supports a maximum of about 7 transaction per second. The scalability concerns [7] in Bitcoin have led to the controversial *Bitcoin Classic*[4] and *Bitcoin XT*[5] forks by some of the earliest Bitcoin developers, as well as other forks such as *Bitcoin Unlimited*[6]. Let us note that keeping the 7 transactions per second limit is also problematic, since the miners will need to either prune or keep an ever-growing amount of extra transaction in their local storage (a.k.a. local memory pool) in case the users of Bitcoin broadcast an average of more than 1 megabyte of transaction data per 10 minutes. Thus, it may well be the case that an increased popularity of a blockchain-based cryptocurrency can ultimately be accommodated either via technology advancements with regard to the speed of network transmission, or via different kinds of performance and trust trade-offs (cf. [35]).

In Meshcash, the unfairness problem that the nonlinearity of rewards phenomenon implies is significantly less severe. This is due to the cooperative nature of Meshcash: each miner who solves a block in the current layer will be rewarded, irrespectively of the other miners who also currently solve blocks. Hence, Meshcash can continue to operate in a decentralized mode while accommodating blocks with a higher size limit relative to Bitcoin, since the effect of a higher network propagation latency does not cause a nonlinear distribution of rewards to the Meshcash miners. Still, a higher propagation latency would improve the prospects of an active attack on Meshcash, cf. Section 3.4.

B Some standard tail bounds for the Poisson distribution

We will use the following standard tail bounds on the Poisson distribution:

$$\Pr_{X\sim\mathrm{Pois}(\lambda)}[X \le x] \le \frac{e^{-\lambda}(e\lambda)^x}{x^x} \qquad \text{, for } x < \lambda \tag{B.1}$$

$$\Pr_{X\sim\mathrm{Pois}(\lambda)}[X \ge x] \le \frac{e^{-\lambda}(e\lambda)^x}{x^x} \qquad \text{, for } x > \lambda \tag{B.2}$$

These imply for $c > 1$:

$$\Pr_{X\sim\mathrm{Pois}(\lambda)}[X \le \lambda/c] \le e^{-\frac{\lambda}{c}(c-\ln c-1)} = 2^{-\lambda\frac{c-\ln c-1}{c\ln 2}} \tag{B.3}$$

$$\Pr_{X\sim\mathrm{Pois}(\lambda)}[X \ge c\lambda] \le e^{-c\lambda(\frac{1}{c}+\ln c-1)} \le 2^{-\lambda\frac{c}{\ln 2}(\frac{1}{c}+\ln c-1)} \tag{B.4}$$

or equivalently

$$\Pr_{X\sim\mathrm{Pois}(c\lambda)}[X \le \lambda] \le 2^{-\lambda(c-\ln c-1)/\ln 2} \tag{B.5}$$

$$\Pr_{X\sim\mathrm{Pois}\left(\frac{\lambda}{c}\right)}[X \ge \lambda] \le 2^{-\lambda(\frac{1}{c}+\ln c-1)/\ln 2} \tag{B.6}$$

[4] See `https://bitcoinclassic.com/`.

[5] See `https://bitcoinxt.software/`.

[6] See `https://www.bitcoinunlimited.info/`.

For $c = 2$ the approximations are:

$$\Pr_{X\sim\mathrm{Pois}(\lambda)}\left[X \leq \frac{1}{2}\lambda\right] \leq 2^{-\frac{1}{5}\lambda} \tag{B.7}$$

$$\Pr_{X\sim\mathrm{Pois}(2\lambda)}\left[X \leq \lambda\right] \leq 2^{-\frac{2}{5}\lambda} \tag{B.8}$$

$$\Pr_{X\sim\mathrm{Pois}(\lambda)}\left[X \geq 2\lambda\right] \leq 2^{-\frac{1}{2}\lambda} \tag{B.9}$$

$$\Pr_{X\sim\mathrm{Pois}(\lambda/2)}\left[X \geq \lambda\right] \leq 2^{-\frac{1}{4}\lambda} \tag{B.10}$$

Nomenclature

$M_i^{\dagger}(A)$ The i^{th} actual confirmation margin for a block A claiming to be in layer $j < i$, page 20

δ A bound on the combined network propagation time/local clock differences between honest nodes, page 15

Δ_{j-1} the number of blocks published by the adversary from its reserve at time $\texttt{start}_j + \delta$, page 20

$R_j(i)$ the adversary's reserve for blocks in layer i at time $\texttt{start}_j + \delta$ (not including blocks that were published at time $\texttt{start}_j + \delta$), page 20

θ If the vote margin (sum of the votes) for a block is less than θ (in absolute value) nodes will vote according to the output of the weak coin (instead of using the majority), page 15

F_i the adversary's future reserve, subset of blocks in its reserve that claim to be in future layers (i or greater), at time $\texttt{start}_i$, page 20

ℓ The average length of a layer interval, page 15

ℓ_ε^* an upper bound on the duration of a layer, page 19

ℓ_ε^- a lower bound on the duration of a layer in which the adversary is in the future-reserve steady-state, page 19

$M_i^{(P)}(A)$ the i^{th} confirmation margin for block A as seen by party P, page 20

t^{coin} Time for weak coin protocol (in multiples of δ), page 15

$T_{\min}$ The minimum number of blocks in a layer, page 14

Y_j the number of maliciously-generated blocks in the interval $[\texttt{start}_j + \delta, \texttt{start}_{j+1} + \delta]$, page 19

Y_ε^* an upper bound on the number of maliciously-generated blocks, page 19

Z_ε^* a bound on the number of honestly-generated blocks in interval $[\texttt{start}_j + \delta, \texttt{start}_{j+1}]$, page 19

Z_j^{coin} the number of non-abstaining blocks in layer j, page 19