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

Iddo Bentov[1], Pavel Hubáček[2], Tal Moran[3], and Asaf Nadler[4(✉)]

[1] Cornell Tech, New York, NY 10044, USA
iddobentov@cornell.edu
[2] Charles University, Malostranské nám. 25, 118 00 Prague, Czech Republic
hubacek@iuuk.mff.cuni.cz
[3] IDC Herzliya, 8 HaUniversita Street, 4610101 Herzliya, Israel
talm@idc.ac.il
[4] Ben-Gurion University of The Negev, 8410501 Beer-Sheva, Israel
asafnadl@post.bgu.ac.il

Abstract. We propose Meshcash, a protocol for implementing a permissionless ledger (blockchain) via proofs of work, suitable for use as the underlying consensus mechanism of a cryptocurrency. Unlike most existing proof-of-work based consensus protocols, Meshcash does not rely on leader-election (e.g., the single miner who managed to extend the longest chain). Rather, we use ideas from traditional (permissioned) Byzantine agreement protocols in a novel way to guarantee convergence to a consensus from any starting state. Our construction combines a local "hare" protocol that guarantees fast consensus on recent blocks (but doesn't, by itself, imply irreversibility) with a global "tortoise" protocol that guarantees irreversibility. Our global protocol also allows the ledger to "self-heal" from arbitrary violations of the security assumptions, reconverging to consensus after the assumptions hold again.

Meshcash is designed to be *race-free*: there is no "race" to generate the next block and 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

A full version of this paper is available as [4]

P. Hubáček—This work was performed while at the Foundations and Applications of Cryptographic Theory (FACT) center, IDC Herzliya, Israel

T. Moran—This work was supported in part by the Bar-Ilan Cyber Center

S. Dolev et al. (Eds.): CSCML 2021, LNCS 12716, pp. 114–127, 2021.
https://doi.org/10.1007/978-3-030-78086-9_9

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 bounded-delay communication model of Pass, Seeman and shelat, and against a constant fraction of Byzantine (malicious) miners; not just rational ones.

Keywords: Blockchain · Byzantine agreement · Consensus · Scalablility

1 Introduction

The problem of how to achieve a distributed consensus is one that has been widely studied, both as a theoretic question and as a practical matter. In the classical formulation of the problem (and the one most studied), the set of participating parties are fixed in advance and known to each other. This is a good model for the problems that motivated Lamport, Shostak and Peace in their seminal paper [14]—how a small number of servers, some of whom may be faulty, can still provably reach agreement.

Several decades later, with the advent of cryptocurrencies, we have a new motivation achieving distributed consensus. All currencies, and cryptocurrencies among them, inherently require consensus—if Charlie believes that Alice paid Bob, then Dana and Eve should not believe a contradicting claim.

The cryptocurrency setting doesn't fit neatly into the classical Byzantine agreement model. First, requiring every participating party to know every other party in advance is not feasible at "Internet Scale". In addition, without a trusted third party, the problem of identity verification on the Internet is notoriously hard. Together with the impossibility of Byzantine agreement without an honest majority, it seems that achieving provable consensus is impossible in this setting. Surprisingly, there is a way to sidestep these barriers—by changing the model to let participants prove that they possess scarce resources. Indeed, this is precisely what Nakamoto did with the invention of Bitcoin [20].

1.1 Consensus, Money, and Contracts

The technical descriptions of cryptocurrencies usually specify one intricate protocol that "solves" multiple problems at once: how to agree on history, what the currency can do (what is a transaction/smart contract) and the currency's monetary policy (e.g., how the coin supply is controlled). However, the solutions to these problems are, in many ways, independent.

We can separate the building blocks of a cryptocurrency intro three layers:

1. **Ledger**: The ledger layer is responsible for generating a consensus on an "append-only ledger". The ledger maintains a list of *transactions*: the protocol specifies how someone who attaches to the network can retrieve this list. Although different parties might get slightly different lists, the ledger protocol should guarantee several properties to be useful for a cryptocurrency:
   - *Safety (consensus on transaction order)*: all honest parties must agree on the set of transactions that appear in the ledger *and their order*. There may be disagreement about recent entries in the ledger, but as we look further back in history the probability of disagreement should go down exponentially.
   - *Irreversibility*: the ledger cannot be modified—only extended with additional entries (as with safety, irreversibility is only required to hold for "sufficiently old" transactions).
   - *Liveness*: the ledger grows over time (i.e., an adversary can't prevent *some* new transactions from eventually being added to the ledger).
   - *Fairness*: the fraction of honest transactions in the ledger is proportional to the accumulated resources of the honest users. In particular, attackers cannot force the ledger to include only their own transactions.

   This notion of ledger is equivalent to what Pass, Seeman and shelat formally define as a *blockchain* [21].
2. **Consensus Computer**: The consensus computer [17] is a state machine responsible for transforming an ordered list of transactions into a useful "state". At this layer we can define coins, accounts and contracts, and specify how transactions can manipulate them.
3. **Economy**: The economy layer describes how coins are created and destroyed and how monetary policy is determined and implemented. Examples of questions addressed in this layer are "is the supply of coins is capped?", "do old coins expire?" and "how do we allocate the initial distribution of funds?".

This work focuses on the design of the *ledger* layer. For the purposes of this layer, transactions are opaque strings—there is no need to interpret them in any way. This is an important property, since it allows us to "modularize" cryptocurrencies—the underlying ledger can replaced without changing the layers above it.[1]

1.2 Permissionless Consensus via PoW

In a *permissionless* distributed consensus protocol (cf. [3]), parties do not have to ask permission from others in order to join the protocol execution. The challenge in constructing a permissionless protocol is handling malicious adversaries that can create an unbounded number of "fake" identities.

[1] For optimization purposes, a cryptocurrency built on top of the ledger can add additional restrictions to prevent clearly invalid transactions from entering the ledger in the first place, but we ignore that here.

To solve this problem in Bitcoin, Nakamoto added a new assumption to the model: the adversary controls less than half of the total computing power invested in the protocol. At a high level, one can think of the participants in the Bitcoin protocol (called *miners*) as "voting" on the consensus value, but instead of having one vote per miner, each vote must be accompanied by a Proof of Work (PoW): a proof that computational power was "wasted" in order to generate that vote. Since we assume honestly-behaved users control a majority of the computational power, they can cast more votes than the adversary.

The details are, of course, a little more complex. In an Internet-scale protocol, it's infeasible to have every user constantly send "votes" to the entire network (this would require all users to have extremely high communication bandwidth and make verification very costly). Instead, Bitcoin employs a "lottery": one can think of the PoW as composed of many purchases of lottery tickets. Each purchase costs little in terms of computational power, but the probability of winning is very low, so many attempts (and hence, many CPU cycles) are required to find a winning ticket. The big advantage of this scheme is that only the lottery winner needs to publish a ticket. Since the cost of verifying a winning ticket is tiny compared to finding one, both communication and computation costs can be made very low.

1.3 Importance of Incentive-Compatibility

How reasonable is Nakamoto's assumption? Since cryptocurrencies have become a widespread phenomenon backed by "real" money, it seems unlikely that a majority of the computational power is controlled by *unconditionally honest* parties—at the sums involved, if parties can gain a significant financial advantage by deviating from the honest protocol is very likely that most will do so. That is, it seems more reasonable to model a majority of the parties as *rational* rather than honest. Thus, for security to hold—i.e., for honestly-*behaving* parties to control a majority of the computational power—we need to ensure that rational parties prefer to behave honestly. Protocols that satisfy this property are called *incentive-compatible*.

1.4 Drawbacks of Leader Election

The Bitcoin PoW "lottery" is a special case of what we call a *leader-election*-based consensus protocol. In leader-election-based protocols, a single party is (perhaps implicitly) selected as a "leader" at every time period (in the case of Bitcoin, the leader is the latest miner to "win" the lottery). Almost all existing cryptocurrency protocols are based on leader-election (most take an approach similar to Bitcoin's). Although it appears to be a mere technical distinction, it turns out that basing a protocol on leader-election can have negative consequences. In protocols based on leader-election, by definition, only one miner can "win" at any given time. On the other hand, in the real world the underlying communication network will have propagation delays (possibly under adversarial control), so multiple honest parties may believe they have won at the same time. This kind of collision creates a "race"; if the incentive structure for the protocol is tied to winning (e.g., in Bitcoin only the winning parties are rewarded), this can

create incentives to behave maliciously. For example, adversaries can increase their own probability of winning by using denial-of-service attacks against other users. Indeed, Bitcoin is known to be vulnerable to several types of "rational" attacks (such as "selfish mining" [9,10,13,21,23] and "undercutting" [5] attacks).

The risk of rational deviations can severely limit a cryptocurrency in other ways as well. For example, Bitcoin suffers from centralization of the mining power in the hands of few large data centers [7,19] as well as scalability barriers w.r.t. high volume of commerce [6,15,25]. Intuitively, these arise since to increasing the throughput of transactions (or rewards, in the case of the centralization problem) we must either increase the frequency of "winning", in which case we have more races, or increase communication per win, which in Bitcoin increases the network propagation delay—again causing more races (see the full version for additional details).

1.5 Our Contributions

In this work, we present a new permissionless ledger protocol that aims to either solve or mitigate the aforementioned risks.

Leaderless Protocol, Provable Security. The fundamental idea behind Meshcash is a novel permissionless consensus protocol that is not based on leader-election. Unlike most alternative permissionless consensus protocols, we prove our security guarantees with regards to *malicious* adversaries and in a semi-synchronous communication model (with bounded delay, cf. [21]). This is to say, our protocol is robust even against non-rational adversaries, as long as they do not have too large a fraction of the computation power.

Self-healing Consensus. A long-term protocol that is intended to be used for decades must take into account even very low-probability events, if the probability that they occur over the lifetime of the protocol is still significant. For example, the probability of a massive space-storm that disrupts global communication, or of widespread blackouts, is minuscule for any given day—but something that is entirely non-negligible over a century.

Violations of common security assumptions fall into this category; e.g., the bounded delay assumption would fail if communication was sufficiently disrupted, and widespread blackouts might cause a large fraction of honest miners to drop out of the network until power is restored. Thus, robust cryptocurrency ledger protocols must be resilient in the face of temporary violations of their assumptions.

In essence, we gives "two tiers" of security guarantees:

1. As long as the standard security assumptions hold, Meshcash achieves fast consensus against adversaries controlling up to 1/3 of the hash power, and consensus about sufficiently old blocks is irreversible against an adversary controlling up to 1/2 of the hash power. Note that since Meshcash disincentivizes pool mining, a broader distribution of hashpower is more likely (cf. Sect. 1.5). Thus, when compared to a ledger protocol that tends towards centralized pools, Meshcash's security assumption is more conservative even with an identical hashpower threshold.

2. Even if the security assumptions are temporarily violated (e.g., the adversary controls the hash power of the *entire* network for some period of time), once the attack is over the protocol can "self-heal"; after a period of "healing time" that depends on the severity and duration of the violation, all honest users will converge again to consensus (during the "healing time" we require slightly stricter security assumptions—the adversary must control less than 1/15 of the network hashpower). Note that irreversibility of consensus on old blocks continues to hold as long as the total computation power expended by the adversary after the block publication is less than the total computational power expended by honest users.

We note that Bitcoin self-heals in a similar manner, but its formal proofs [10,21] do not offer this analysis, whereas we give a rigorous proof of Meshcash's self-healing guarantee.

Ideas from Permissioned Distributed Consensus. One of our main technical contributions is using ideas from the permissioned Byzantine agreement literature in order to achieve a consensus on multiple generated blocks in each time period, rather than having a race to choose the "leader" of the next round. An example is the use of a "weak coin protocol" to give fast, probabilistic consensus even when honest parties are initially split in their opinions. Weak coins have been used as black boxes in traditional consensus protocols too. Possibly of independent interest, we construct a weak-coin protocol with provable guarantees based only on PoWs.

Our techniques give a **qualitatively different type of permissionless consensus protocol**, and may prove useful to improve scalability and incentive-compatibility in other ledger designs as well (including those not based on PoWs).

Semi-permissioned Committee Selection. Another novel contribution, that may be of independent interest, is a subprotocol used by miners to reach consensus on a "committee" that includes all honest miners who generated valid blocks in a given time period. This protocol is "semi-permissioned" in that the number of participants is bounded, but honest parties may not agree on *who* is participating. We construct an SPCS protocol in the synchronous model that is resilient to *any* fraction of corrupted adversaries. The protocol is based on the Dolev-Strong synchronous broadcast; a more detailed description appears in the full version. The goal and underlying ideas are similar to that of Andrychowicz and Dziembowski [1], but our protocol is simpler.

Race-Freeness. We prove that in the Meshcash protocol, one miner's success does not prevent the success of another. This "race-free" property is highly desirable; we show that it is a sufficient condition for a protocol to be incentive-compatible (under some simplifying assumptions). Thus, we can show that even if *none* of the parties are honest, and the non-malicious participants are merely rational, honest behavior is an equilibrium for the protocol. This makes the "honest majority" assumption far more believable in practice.

Improvements over Bitcoin. 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 min 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 the full version 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 [17]. Indeed, this behavior appears to be widespread among Bitcoin miners, as some miners lost a significant amount of funds due to the BIP66 softfork [18]. 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.
- **Incentive-compatible propagation.** A rational Bitcoin miner may decline to re-transmit 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. Since Meshcash divides the transaction fees among all miners who created blocks in the recent layers, 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 [16]. A rational miner may fork a high-value block by collecting only some of its transactions, to incentivize the next miners to extend her forked block and earn extra fees by picking up the rest of the transactions. A malicious adversary may even put a "poisonous" transaction tx_1 in the honest chain and then offer high fees for blocks that include another transaction that conflicts with tx_1, thus bribing rational miners to work on a fork. In Meshcash, the fees are shared and conflicting transactions in a layer do not invalidate blocks that reference them, 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 previous 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, Meshcash achieves the following guarantees.

Theorem 1 (Security—informal). *If the adversary controls less than a* $q < 1/3$ *fraction of the computational power then the Meshcash protocol satisfies the safety, irreversibility, liveness and fairness properties of a permissionless ledger.*

Theorem 2 (Self-healing—informal). *Regardless of the initial state of the honest parties, after a sufficiently long period in which the adversary controls a* $q < \frac{1}{15}$ *fraction of the computational power, the Meshcash protocol will satisfy the safety, irreversibility, liveness and fairness properties.*

For the formal statements see the full version. Note that 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.

1.6 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 [22]. However, many previous discussions of DAG-based cryptocurrencies lack formal analysis (most even lack a full specification). Unsurprisingly, the "devil is in the details"—constructing protocols that can withstand attack by a malicious adversary that can affect network messages is highly non-trivial. This is evidenced by the fact that even for Bitcoin, which is relatively simple and well studied, followup analysis showed vulnerabilities due to network delays [9,10,21]. It should be noted that many cryptocurrencies retain the chain topology but replace the PoW element with other kinds of

Sybil-resistant mechanisms, such as *Proof of Stake*. Examples of stake-based protocols include Algorand [11] and Ouroboros [12]. Indeed, [11, Sect. 2] raises the idea of improving the scalability of Algorand via a DAG topology.

In protocols based on leader election (e.g., GHOST [25], Bitcoin-NG [8]), consensus is achieved by selecting some "special" party (the leader) in each round of the protocol. Since only one party can 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 according to our definition.

There are far fewer examples of protocols that do not require a leader election. The best-known protocols (with formal analysis) are SPECTRE [24] and PHANTOM [26].

See the full version for a more detailed comparison of related works.

2 Informal Protocol Overview

In this section we provide an overview of Meshcash. The complete protocol is appears on the full version.

Meshcash is a permissionless ledger protocol; participants in the protocol can join and leave the protocol at any time.

System Stakeholders. Every participant may play one or more of the following roles:

- *miners* are responsible for the security of the system, and participate by running the Meshcash mining protocol. At a high level, this is very similar to Bitcoin's mining—it consists of listening for new blocks on the network while performing computations to generate blocks (which they then publish).
- *validators*, receive blocks published by the miners and are responsible for determining which blocks are valid (i.e., are contained in the ledger) and their order.
- *users* publish transactions that they would like to add to the ledger.

Communication Model. In terms of execution, miners and validators in the Meshcash protocol behave similarly to Bitcoin—the parties are connected via a "gossip network". Our assumption is that every two honest parties are connected via the gossip network with some bounded delay.

Block DAG. In Bitcoin, each block points to one previous block, forming a chain. In Meshcash, the structure is instead a layered DAG; each block belongs to a layer (it contains a field that explicitly declares the layer number) and points to blocks in previous layers.

Types of Block Validity. We classify validity rules into two types: *syntactic* and *contextual*. Syntactic validity is what can be determined entirely by the contents of the block (and the block's *view*—the blocks reachable from it in

the DAG). This includes things like whether the PoW is valid, and whether the blocks it points to have valid PoWs.

We call any validity rule that isn't syntactic a contextual validity rule. This includes rules that depend on other blocks received later (e.g., the Bitcoin "longest-chain" rule is contextual, since a block can be invalidated if another, longer, chain is received by the miner).

Syntactic Validity Rules. The syntactic validity rules are very simple. Like Bitcoin, we require the PoW to be valid for the block and match the block's difficulty level (optionally, as an efficiency optimization, we can also require all included transactions to be syntactically valid—but in any case we don't check conflicts with transactions in other blocks). In addition, we require every block to point to at least $T_{\min}$ blocks in the previous layer (where $T_{\min}$ is a tunable parameter). This is one of the innovations in our protocol; it makes it much harder for the adversary to pre-generate blocks in future layers—since to do so it would have to pre-generate $T_{\min}$ blocks in *every* layer; we rely on this heavily in our proof that the protocol can "self-heal" from an arbitrary adversarial state (in which the adversary may have pre-computed many blocks).

Contextual Validity Rules. The contextual validity rules are a little more complex, and the technical heart of the protocol. At a high level, the idea is that we let every block "vote" about all previous blocks in its view. For exposition purposes, think of these votes as being explicitly encoded in each block (in the actual protocol, we will do the encoding implicitly). To decide whether a block is valid, the validator counts the votes from all (syntactically-valid) blocks in its view, and takes the majority.

For very recently published blocks, the miners can't use this strategy (since not enough subsequent blocks have voted yet). Instead the miners who published blocks in the previous layer use local timing information to decide on validity of the blocks in that layer, and then run a "semi-permissioned" byzantine agreement protocol to reach consensus on their validity. The output of this local protocol (i.e., the validity of each block in the target layer) is signed by each miner and published. Validators decide on the validity of recent blocks by taking a majority of the signed outputs.

If we don't care about self-healing, the protocol as described above would suffice. However, if security assumptions fail—even temporarily—the local protocol is no longer guaranteed to reach consensus (since it requires a majority of the blocks to be honestly generated in each layer). In this case, by timing the publication of a block, the adversary could cause honest parties to disagree about its contextual validity, and then use a "balancing attack" to keep the honest parties evenly split. Balancing requires only a small fraction of the honest party's resources, so the split could continue indefinitely even after the security assumptions hold again.

To overcome this type of attack, we add another condition to the contextual validity rule: if the "vote margin" is small (i.e., the number of blocks voting for and against is similar), the miner will "flip a coin" instead of relying on the vote. The trick is that we will use a weak *common* coin—that is, with some

known, constant probability all honest parties will agree on the result of the coin flip. Intuitively, when the adversary guesses the coin's value incorrectly, it will support the "wrong" side and the balancing will fail.

Mining Algorithm. A Meshcash miner uses only syntactic validity in order to decide which blocks to point to—the miner will point to *every* syntactically-valid "head" block it sees (i.e., blocks with in-degree 0). Thus, although the contextual validity rules are more complex than Bitcoin's, the mining algorithm is almost as simple; indeed, we implemented the mining algorithm in just over 100 lines of python code (the count includes only the top-level algorithm, without e.g., the local protocol or PoW implementation).[2]

3 Meshcash Security

Our basic security properties are as defined in [21]:

- *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 T/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.

3.1 Security Proof Overview

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

Consistency (Safety). When security assumptions are satisfied, the set of blocks in every consecutive range of *ldist* layers will have an honest majority except with negligible probability. This follows from the fact that the adversary cannot pre-generate too many blocks (e.g., as shown in the full version, when $q < 1/3$ the adversary can't have much more than $\frac{1}{2}T_{\min}$ blocks of layer i at $\texttt{start}_i$), hence with high probability the fraction of adversarial blocks in a given time period cannot be much more than q (e.g., when $ldist = 2$ and $q = 1/3$, the adversary would have less than half of the blocks w.h.p.).

At layer t, the local protocol uses a consensus algorithm between miners who generated blocks in the past *ldist* layers to agree on the validity of blocks in layer $t - ldist$; this is guaranteed to achieve consensus when the majority of the blocks are honest.

[2] The code can be found on https://github.com/anon444/meshcash.git.

Future Self-consistency (Irreversibility). At a high level, we can view the global protocol as a voting process: every new block "votes" for or against each previous block. The irreversibility of the 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. We formalize this race analysis in the full version and show that the vote margin will grow linearly with the number of layers.

Self-healing Irreversibility. The main challenge 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 the full version, we show that, irrespective of the initial conditions, there will be a layer in which the adversary's future reserve reaches a steady-state. Additionally, we use the fact that with overwhelming probability no layer is "too long" 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 the full version).

When the adversary is in its future reserve steady state, consistency and future self-consistency are guaranteed (see explanation in the full version). The idea here is straightforward—once we have achieved consensus in the local protocol, the honest parties all vote in the same direction, hence the margin will grow until it reaches the threshold for irreversibility to apply.

Self-healing Consistency. 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 to cause disagreements among the honest miners on the contents of the layers.

Our main technical theorem that appears in the full version, shows that for any initial reserve of blocks (here we do not care about whether they are in the future or the past), the global 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.

Chain Quality (Fairness), Chain Growth (Liveness) and Race-Freeness. To show that honestly-generated blocks are always in the consensus (i.e., ensuring optimal $(1 - q)$ chain quality), we need to lower-bound the number of honest blocks in every layer (since honest blocks are "guaranteed" by the local protocol 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. Chain growth also follows from the property that every honestly generated block will be considered valid, i.e., Meshcash achieves g-chain growth where g is the expected number of honest blocks in a network round. See the full version for further details.

References

1. Andrychowicz, M., Dziembowski, S.: Pow-based distributed cryptography with no trusted setup. In: Gennaro, R., Robshaw, M. (eds.) Advances in Cryptology - CRYPTO 2015–35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16–20, 2015, Proceedings, Part II, volume 9216 of Lecture Notes in Computer Science, pp. 379–399. Springer (2015). https://doi.org/10.1007/978-3-662-48000-7_19
2. Babaioff, M., Dobzinski, S., Oren, S., Zohar, A.: On Bitcoin and red balloons. In: ACM Conference on Electronic Commerce, pp. 56–73 (2012)
3. Barak, B., Canetti, R., Lindell, Y., Pass, R., Rabin, T.: Secure computation without authentication. In: CRYPTO, Yehuda Lindell (2005)
4. Bentov, I., Hubáček, P., Moran, T., Nadler, A.: Tortoise and hares consensus: the meshcash framework for incentive-compatible, scalable cryptocurrencies. Cryptology ePrint Archive, Report 2017/300 (2017). https://eprint.iacr.org/2017/300
5. Carlsten, M., Kalodner, H., Weinberg, S.M., Narayanan, A.: On the instability of bitcoin without the block reward. In: ACM CCS, pp. 154–167 (2016)

6. Croman, K.: On scaling decentralized blockchains. In: Financial Cryptography 3rd Bitcoin Workshop (2016)
7. Eyal, I.: The miner's dilemma. In: IEEE S&P (2015)
8. Eyal, I.: Gencer, A.E., Sirer, E.G., van Renesse, R.: A scalable blockchain protocol. In: NSDI, Bitcoin-NG (2016)
9. Eyal, I., Sirer, E.: Majority is not enough: Bitcoin mining is vulnerable. In: Financial Cryptography (2014)
10. Garay, J., Kiayias, A., Leonardos, N.: The Bitcoin backbone protocol: analysis and applications. In: Eurocrypt (2015). http://eprint.iacr.org/2014/765
11. Gilad, Y., Hemo, R., Micali, S., Vlachos, G., Zeldovich, N.: Algorand: scaling byzantine agreements for cryptocurrencies. In: SOSP, pp. 51–68. ACM (2017). https://eprint.iacr.org/2017/454
12. Kiayias, A., Russell, A., David, B., Oliynykov, R.: Ouroboros: a provably secure proof-of-stake blockchain protocol. In: Katz, J., Shacham, H. (eds.) Advances in Cryptology - CRYPTO 2017–37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20–24, 2017, Proceedings, Part I, volume 10401 of Lecture Notes in Computer Science, pp. 357–388. Springer (2017)
13. Kiffer, L., Rajaraman, R., Shelat, A.: A better method to analyze blockchain consistency. In: ACM CCS 2018 (2018)
14. Lamport, L., Shostak, R., Pease, M.: The byzantine generals problem. ACM Trans. Prog. Lang. Syst. **4**(3), 382–401 (1982)
15. Lewenberg, Y., Sompolinsky, Y., Zohar, A.: Inclusive block chain protocols. In: Financial Cryptography and Data Security, pp. 528–547 (2015)
16. Liao, K., Katz, J.: Incentivizing double-spend collusion in Bitcoin. In: Financial Cryptography Bitcoin Workshop (2017)
17. Luu, L., Teutsch, J., Kulkarni, R., Saxena, P.: Demystifying incentives in the consensus computer. In: 22nd ACM CCS (2015)
18. Maxwell, G.: (2015). https://bitcointalk.org/index.php?topic=1108304.msg11786046#msg11786046
19. Miller, A., Kosba, A.E., Katz, J., Shi, E.: Nonoutsourceable scratch-off puzzles to discourage Bitcoin mining coalitions. In: 22nd ACM CCS (2015)
20. Nakamoto, S.: Bitcoin: a peer-to-peer electronic cash system. Bitcoin.org (2008). http://www.bitcoin.org/bitcoin.pdf
21. Pass, R., Seeman, L., Shelat, A.: Analysis of the blockchain protocol in asynchronous networks. In: Eurocrypt 2017 (2017). http://eprint.iacr.org/2016/454
22. "Maged" (pseudonym). Re: Unfreezable blockchain (2012). URL: https://bitcointalk.org/index.php?topic=57647.msg686497#msg686497
23. Sapirshtein, A., Sompolinsky, Y., Zohar, A.: Optimal selfish mining strategies in Bitcoin. In: Financial Cryptography (2016)
24. Sompolinsky, Y., Lewenberg, Y., Zohar, Spectre, A.: A fast and scalable cryptocurrency protocol (2016). https://eprint.iacr.org/2016/1159
25. Sompolinsky, Y., Zohar, A.: Secure high-rate transaction processing in Bitcoin. In: 19th Financial Cryptography and Data Security (2015)
26. Sompolinsky, Y., Zohar, A.: PHANTOM: a scalable blockdag protocol. IACR Cryptology ePrint Archive, 2018:104 (2018). http://eprint.iacr.org/2018/104