spacemesh

Technical White Paper 1

Revision 1.2[1]

https://spacemesh.io team@spacemesh.io @teamspacemesh

A fair, secure, decentralized and scalable
cryptocurrency and a smart contracts computer

Abstract

Thus far, all of the deployed blockchains exhibit centralization tendencies and scalability bottlenecks as they gain in popularity. Bitcoin and other Proof of Work (PoW) based systems contribute to massive energy consumption, whereas Proof of Stake (PoStake) based systems require trusted checkpoints and high collateral.

Enter SpacemeshOS: a blockmesh operating system powering an energy-efficient, decentralized, secure, and scalable smart contracts global computer and a cryptocurrency in the permissionless settings. The heart of Spacemesh is a new consensus protocol that replaces PoW with Proof of Space Time (PoST), and a chain with mesh topology.

The Spacemesh protocol allows newcomers to contribute to the security of the cryptocurrency network via unused storage space on their hard drives. PoST retains the properties of PoW that are useful for permissionless consensus, but with drastically lower energy emissions. In contrast to PoStake, the PoST mechanism enables users to participate in the system without large security deposits that are locked for long periods of time, and also avoids other nothing-at-stake problems, such as the higher risk of divergence due to hard fork replicas.

SpacemeshOS can support a highly scalable network with thousands of transactions per second, without compromising the decentralized nature of the system. This is achieved using an incentive-compatible protocol that is based on mesh topology - layered directed acyclic graph (DAG). As opposed to the simpler chain topology, the mesh avoids race conditions and guarantees that an honest miner will obtain a fair share of the rewards, regardless of the actions of other miners. Furthermore, the blockmesh enables far more frequent rewards than a blockchain, which means that pressures against decentralization are thwarted, since the individual miners have no need to rely on centralized mining pools.

Spacemesh is the first free open source, permissionless consensus protocol that neither relies on PoW nor PoStake, and is provably secure under the assumption that a majority of the participating storage is honest. The incentive-compatibility of Spacemesh increases the likelihood that this assumption will hold.

Scope, Goals, and Target Audience

Present-day blockchain technologies face several problems in their current applications. By outlining these imperative problems and our solutions to the blockchain community, we intend to open a dialogue on our point of view with our audience so that we may further improve our ideas, approaches, and new protocols together. Our goal with this paper is not to generate a token sale event, but rather to use this text as an anchor for discussion of these prominent systemic issues and our solutions in order to formalize the various components of this project.

It is worth noting that we are not attempting to provide complete security and incentive compatibility proofs of our proposed blockchain protocols in this paper. The proofs will be made available online and submitted for peer review by academic cryptographers later in 2018. Spacemesh is a work in progress, as active research is underway. New versions of this paper will appear at https://spacemesh.io. Comments and suggestions may be submitted by contacting us at team@spacemesh.io .

This paper is written to specifically address technical readers who have an understanding of the capabilities and issues of existing blockchain protocols. The audience may also be considered as participants; that is, readers who are interested in exploring approaches to solve these issues. Collaborating in this way will advance the state of blockchain infrastructure to make it more decentralized, useful, robust, scalable, and secure.

Changelog

Revision 1.1 5/7/2018

- Problem statement and first protocol informal description

Revision 1.2 5/27/2018

- Added definition of hdist
- Minor cosmetic and grammar corrections
- Updated figures numbering

Table of Contents

Section 1 - Motivation and Problem Statement

Our vision is to build a *decentralized*, *secure,* and *scalable* cryptocurrency and smart contracts computer that solves major problems of 1st and 2nd generation blockchain networks. We are committed to *fairness* as a cornerstone value of the network and aim to deliver it by developing low barriers to entry, setting sustainable transaction fees and enabling widespread, distributed participation. To realize our vision, we are developing SpacemeshOS, which is a blockmesh operating system designed to run general-purpose smart contracts at scale, powered by a set of provably secure protocols that are energy-efficient and incentive-compatible.

The main breakthrough idea in Bitcoin was a protocol that achieved consensus on a public ledger in a permissionless setting, meaning that anyone with a PC and an internet connection could contribute to the security and capabilities of the network. The original vision was to create a global internet currency that could be used as cash without a central issuing bank, that had fewer intermediaries, and that charged lower transaction fees. The Ethereum project took the core ideas of Bitcoin and greatly expanded the “smart contracts” capabilities that enabled coin holders to participate in complex financial transactions and to execute contracts ruled by blockchain deployed code. These ground-breaking projects paved the way for both a broader industry and public acceptance of the importance of decentralized open ledger and computation networks.

With 2018 marking Bitcoin’s tenth year in production and Ethereum’s third, we now have insight regarding several serious theoretical, economic, and practical problems regarding cryptocurrency networks. Some of the problems are fundamental to the core protocols, communities, and practical deployments -- and have led us to re-imagine the open blockchain network.

To achieve the original vision of Bitcoin, we propose a system that is designed from the ground up to achieve a high degree of decentralization while providing improvements in security and scalability at the same time. We understand the core issues of the current blockchain networks as centralization, scalability, security, and energy waste, and our analysis presents solutions to these problems.

Decentralization

Centralization is an issue with present-day blockchain networks as well as new blockchain protocol proposals. The original concept for cryptocurrency mining was that individuals would use their home computers to provide security and liveness in the blockchain network.

Decentralization in the permissionless settings is typically defined using the following criteria:

○ The ability of a home computer to function as a full p2p network node
○ The number and geographic distribution of the nodes

- The proportional distribution of nodes among all entities

Broadly speaking, a network is decentralized when it has a large number of geographically distributed entities, where no entity operates more than a small fraction of all nodes.

Centralization occurs due to a high variance for individual miners to obtain rewards, as well as the superiority of dedicated hardware that performs the PoW computation. In Bitcoin, a whole class of dedicated custom hardware was created, which provided several factors of performance improvement over a home PC. Mining farms were then set up at world locations where cheap electricity was available. The overall network hashrate was predominantly determined by ASIC hardware running in massive hardware farms, making it uneconomical to mine Bitcoin using a general-purpose PC.

Using a dedicated ASIC unit at home is also unrealistic. As of March 2018, a high-end dedicated ASIC mining unit with a cost of about $2,000 USD is expected to mine 1 block and receive its block mining reward about once every 36 years.[2] With ongoing operating costs and an investment in mining equipment that may fail before yielding a return, the individual miner has a strong incentive to combine resources into centralized pools. Over time, pools tend to increase in size as miners continue to bring resources together to optimize rewards. Each mining pool is centrally managed as an entity, with the largest three mining pools controlling 54% of all hashing power on the Bitcoin network, and the largest six controlling 77%.

To take another example, the Ethereum hashing algorithm ethash was designed to discourage ASIC development.[3] However, the largest Bitcoin ASIC manufacturer and mining pool operator, Bitmain, recently released an Ethereum ASIC miner product to the market.[4] According to published analysis, the product is about 3 times more cost effective than a homemade mining rig,[5] which is likely to price PC users out of the market over time. In addition, it is possible that Bitmain had secretly been using thee ASIC products for an unknown period of time prior to their announcement to mine Ethereum. It is also possible that Bitmain or any other hardware provider has more powerful mining products that are only used internally, to maintain an advantage and maximize their mining revenue.

The other main driver of centralization is mining pools, which give control over the network to pool administrators who may filter transactions. In particular, these administrators may engage in frontrunning by receiving side payments via direct API service. For instance, mining in Ethereum with a modern GPU on a home PC is only expected to successfully mine a block

[2] https://99bitcoins.com/bitcoin-mining-calculator/
https://www.cryptocompare.com/mining/calculator/btc?HashingPower=11&HashingUnit=TH%2Fs&PowerConsumption=1293&CostPerkWh=0.12&MiningPoolFee=0

[3] http://www.ibtimes.co.uk/could-ethereums-security-be-tested-price-ether-rockets-1550103

[4] https://www.coindesk.com/bitmain-confirms-release-first-ever-ethereum-asic-miners/
https://shop.bitmain.com/product/detail?pid=00020180403174908564M8dMJKtz06B7

[5] https://cryptoslate.com/bitmain-e3-asic-ethereum-miner/

and receive the coin reward once for every 3 years in operation. This makes mining without joining a pool impractical.[6]

We see that the world's leading blockchain network has fallen short of achieving the stated goal of decentralization. Our intent is to incorporate the lessons from the challenges and unexpected outcomes of first and second generation solutions into our design.

We intend to achieve decentralization via solutions that involve core *protocol* properties and the *incentive* structure. We will also discuss the surrounding *governance* issues.

1. Protocols

Spacemesh offers a solution that returns to the original idea of home user gear by making it more rewarding than purchasing professional equipment or joining a pool. Spacemesh blockchain protocols are specifically designed for running a node on a standard desktop PC, making it economically unattractive for Spacemesh-specific ASIC hardware to be developed, and making it unprofitable to use cloud computing. As we will further explain in this section, our protocols achieve these conditions through two key requirements for hardware setup:

○ 250GB[7] of free disk space: used in our Proof of Space Time (PoST) protocols
○ A modern GPU: used in our PoST protocols and in the ongoing consensus protocols

Disk space is a highly underutilized and readily available decentralized resource. It is estimated that 50% of global storage space is free. 412 million hard disks were sold in 2016 and 300 million new disks are forecast to be sold annually until 2020.[8] Decentralized data storage projects such as Filecoin are also based on this premise.[9]

When setting up a node, users specify how much storage to allocate based on the free space of their local hard drive. Moreover, the rewards for contributed storage are more linear as disk space has limited economies of scale and has homogeneity in that there is no difference in the efficacy of one disk vs. another. This ensures that any participant who contributes storage will be rewarded accordingly and cannot be outspent for higher quality storage.

Our PoST protocol creates a cryptographic data structure on the required free space in an initial, one-time ~48-hour setup period using the GPU to perform PoW. Using our PoST protocol, provers demonstrate that they have reserved a portion of their storage for a specified period of time, which we term "space time." The reserved storage cannot be used for anything else during that time (including other PoST proofs). For example, a user can prove that he or she has reserved 100GB for a month: the PoST actually proves the statement "either I stored

[6] https://www.coinwarz.com/calculators/ethereum-mining-calculator/
[7] This figure is just for educational purposes. The final production net may require a different minimum free storage based on further research and testnet simulations.
[8] https://www.statista.com/statistics/398951/global-shipment-figures-for-hard-disk-drives/
[9] https://filecoin.io/filecoin.pdf

100GB for a month, or I wasted a corresponding amount of CPU work." Whatever the actual outcome, we set the parameters so it is more cost effective to use storage than CPU work.

Provable storage entitles nodes to participate in block sealing rewards. The rewards are shared between all honest participating nodes; for instance, owners of 5% of network storage should get 5% of the rewards.

Once the setup period is complete, the stored data is used to establish a node's identity and eligibility to participate in the consensus protocol, which is done in a fair and egalitarian manner. The ongoing operation of the Spacemesh consensus protocols are not CPU, GPU, or I/O intensive and are designed to run in the background without any interruption or performance degradation.

It would not be profitable to create custom hardware (or use it) since there is no hashing algorithm that requires continuous fast computation. Moreover, the main cost factor in disk space is magnetic storage, which has limited potential for exponential performance improvement. That being said, an exponential improvement in storage ability or reduction in cost would be a lucrative opportunity and likely be released to the global general-purpose storage market, making it quickly available to all. Since it is unlikely that custom hardware will be developed for running Spacemesh nodes, the risk of physical mining pools running custom ASIC hardware farms is eliminated and with it a source of the serious centralization issues created by mining pools.

Beyond centralization concerns, mining pools act as a middleman that introduces additional fees on the network, making transactions more expensive through royalty rates of 1- 3%.[10] Additionally, these pools make protocol innovation difficult, as changes may be perceived as likely to reduce mining fees, and thus mining profitability.

Spacemesh protocols eliminate the advantages of joining a virtual mining pool and make it unnecessary to pay a percentage of block rewards to a central entity. The main motivations for joining mining pools are as follows:

- Long intervals between rewards / high interval variance
- Unfair advantages gained by centralization
- Economies of scale: ASIC obtained in bulk at a cheaper price than an individual miner

Our protocol solves the first by allowing a much higher reward rate (we can easily support 200 blocks in every 10-minute period), the second by making reward allocation fairer (there's no advantage to learning the next block earlier than other nodes), and the third by PoST, since specialized hardware for storage space isn't more effective than consumer-level hard drives.

Additionally, PoST protocol is optimized to below marginal cost for validators using personal computers and to be high marginal cost for validators who do not. The target node operator

[10] https://investoon.com/mining_pools/eth

for PoST has already made the investment required to operate the node, including the disk drive used for storing proof of storage.

2. Incentives

A protocol is *incentive-compatible* if following the protocol instructions honestly gives at least as much utility (i.e., profit) as deviating from them (assuming other parties are also following the protocol instructions). In permissionless blockchain protocols, security depends on the majority of the resources being controlled by parties that behave honestly (resources refers to CPU in PoW-based schemes and storage in PoST-based schemes, etc.). However, it seems that most miners are motivated by profits, rather than by morals. The incentive-compatibility property bridges the gap between profit-driven miners and the assumption of an honest majority. We recognize the importance of incentive compatibility; thereby, the Spacemesh protocols currently being developed are designed to be incentive-compatible.

Our network validators' coin reward scheme is designed to make it unprofitable to set up a Spacemesh node on a dedicated PC at home, in a data center, or on the cloud with a provider such as Google or AWS. The validation reward configuration is designed to let a person operating a desktop PC at home, using an existing PC and GPU to run a Spacemesh node. With this configuration, the validator earns validation rewards according to our setup without investing any upfront amount for hardware and without having to put up a PoStake deposit.

It is worth noting that individual validation rewards will gradually decrease over time as more validators join the platform and in turn more disk space is contributed to the system. Consequently, in the long run it will become increasingly unprofitable to use dedicated or cloud storage as the costs will eventually exceed the rewards, making disk space a sunk cost. Ultimately, this incentive scheme is designed to ensure fair, highly-predictable, and frequent compensation to home users who provide the network with security, storage, and decentralized computation capabilities.

Recent proposals for blockchain protocols replace PoW with PoStake. However, there are incentive problems and centralization risks with PoStake protocols. In the case of hardforks, stakeholders may find it rational to participate in all the forks as a short-term dominant strategy that increases their holdings. There is also an inherent costless simulation attack on PoStake, that can be mitigated using checkpoints -- which introduce an element of trust and potential centralization concerns.

A major centralization concern with PoStake is that the system will be dominated by a few large stakeholders. PoStake protocols require the participants to lock a security deposit of a large enough value, in order to deter deviation from the protocol. This makes it prohibitive to participate in the protocol for many node operators with modest financial means, and hence only the larger stakeholders will be able to create blocks and collect transaction fees.

Another problem with PoStake is that the degree of the actual network decentralization relies heavily on initial stake allocation. A PoStake network with a centralized initial stake distribution cannot be decentralized. This is not a theoretical problem: several PoStake-based networks in production suffer from highly-centralized currency allocation.

3. Governance

Governance centralization is an additional important centralization concern. Governance in cryptocurrencies relates to decisions about two things: the rules of the protocol (the code) and the incentives on which the network is based (the economics). Decentralized governance means that the community with a stake in the project participates in the governance of the platform's core technical and financial decisions; governance should also be built into the platform so that it can be readily enforced. When governance of a blockchain protocol is not decentralized, healthy evolution of the protocol that benefits all stakeholders is hard to achieve in practice.

The Spacemesh foundation is a non-profit organization responsible for ensuring network growth, decentralization, and innovation. To execute these responsibilities, the foundation is structured to own a relatively small amount of Meshcoin as well as to retain funds from any Meshcoin sale event.

The foundation is responsible for maintaining the canonical protocol definitions, its open source implementations, and the Spacemesh p2p node software. It performs this duty by appointing maintainers to the various project code repositories. The foundation also has the authority to replace these maintainers at any time. The appointed maintainers decide on feature prioritization as well as on major technical protocol modifications.

To balance between short-term interests of a portion of Meshcoin coin holders and long-term protocol interests, the foundation decisions are made by elected foundation members who are elected in yearly election cycles, with foundation membership open to anyone. Each year, one foundation member seat is open for election for a newly-elected member to replace a current member. This mechanism is designed to allow the community to periodically change the foundation while also providing stability in short-term decision making. We believe that this mechanism helps avoid hardforks or project abandonments as more constructive and less drastic means are provided to influence critical protocol decisions.

In addition, the foundation may occasionally decide to delegate critical protocol decisions to the Spacemesh community. This voting will occur on-chain on the Spacemesh Mainnet through the acceptance of PoST storage commitments. An on-chain vote that achieves a super majority overrides any foundation decision.

Security

In our perspective, security is critical to blockchain networks that are designed to maintain cryptocurrency and financial smart contracts such as Spacemesh. We believe that modern blockchains should be provably secure under reasonable honesty assumptions while being ecologically friendly. A high-degree of security can be achieved by a combination of theoretical and practical measures:

- Carefully consider security strength compared to alternative consensus protocols
- Provide rigorous mathematical security proofs for all protocols and clearly articulate any and all assumptions
- Provide incentive compatibility proofs for all protocols
- Put in place a serious security bug bounties program to find unknown attacks and security vulnerabilities
- Conduct large-scale network simulations of known attacks and modify the security parameters to figure out the best tradeoff between security and scalability

We also recognize the importance of incentive compatibility; thereby, the protocols that we are currently developing are designed to be incentive-compatible in the sense that it will not be rational for a validator to deviate from protocol and become dishonest. The proofs we plan to publish for our protocol will include incentive compatibility proofs.

The following sections detail the Spacemesh foundation’s security measures:

Spacemesh security proofs

Our security proofs assume that a majority of the PoST committed storage is honest. Safety is defined as all honest nodes on the network maintaining the same protocol-validated consensus for a specific set of conditions. Liveness is defined as the blockchain continuing to progress over time, even under adversarial conditions. We are currently developing *incentive compatibility*, *liveliness* and *safety correctness* proofs of our consensus protocol under adversarial conditions and commit to publish them in 2018, prior to launch of the production network. Liveliness and safety under attacks are critical for a robust blockchain platform in the permissionless setting.

Security bounty program

The Spacemesh foundation will invest in a large-scale and continuous bounty program designed to identify both theoretical and implementation-related security vulnerabilities. We intend to specify this resource allocation in the Spacemesh foundation bylaws, which can only be modified by a super majority of Meshcoin holders.

Independent security audits

We will encourage and fund independent security experts interested in conducting comprehensive security audits of our protocols as well as in our software implementation. The budget for such audits will be defined in the Spacemesh foundation budget and can only be modified by a super majority of Meshcoin holders.

Large-scale network simulations

We are in the process of adding tests that perform large-scale network simulations across several regions around the world in the go-spacemesh open source project. These tests will be able, among other things, to demonstrate the network security in case of adversarial conditions, such as ddos attackers, and to harden the network behavior under these common p2p network conditions.

Security and PoStake

Spacemesh protocols use different security assumptions than PoStake-based protocols. For PoStake protocols, one needs to assume some kind of continuity of honest nodes in order to solve the *costless simulation* problem. PoST+PoET is self-contained like PoW. The need for self-containment depends on the degree to which continuity assumptions will hold in real world blockchain networks. Stated differently, Spacemesh relies on different assumptions for its security, which are in some ways more conservative — in particular, the security relies on honest majority of a “physical” resource rather than a “virtual” one. In cryptography, having alternatives for the underlying assumptions is always a good thing.

Scalability

In PoW-based blockchains where there is a single chain of blocks and a race to generate the next block, the network latency implies a strong upper-bound on the maximum transaction throughput. This is due to a combination of two factors:

1. The rate at which blocks are generated cannot be faster than the network latency, otherwise “accidental forks” will occur often. This happens when honest parties try to extend different chains because they do not agree on the order of newly-generated blocks. In turn, a high accidental fork rate compromises security: Instead of the honest parties all working on the “right” chain, their power is split among different chains. Thus, instead of requiring a majority of the computing power to “win,” the adversary must simply gain more computing power than the largest fork.

2. The size of each block is limited since large blocks take longer to transmit. This is undesirable for two reasons: the first is that it increases the network latency, which thus lowers the block rate; and the second is that the miner who generated the block gets an additional unfair advantage in mining the *next* block, since that miner will know the block before everyone else. This increases the motivation to centralize mining in pools.

Since the transaction rate is a function of the number of transactions per block (limited by size) multiplied by the block rate, the total transaction rate is limited.

By removing the “race” aspect of mining and allowing the system to support larger block sizes, the propagation delay becomes much less relevant. It is too soon in the project lifecycle to state the expected transactions per second that will be achieved in practice, however, it is feasible that the lifecycle will be several orders of magnitude better than PoW-based protocols and most likely in the same order as the throughput of well-designed PoStake-based protocols.

Energy Waste

The PoW protocols at the core of many blockchain networks involve irresponsible energy waste on a massive scale. As of January 2018, it is estimated that the Bitcoin network consumes between $2 to $4 million USD worth of energy per day[11] and that the Ethereum network consumes around $2.5 million USD worth of energy per day.[12]

This energy waste is an unfortunate byproduct of the cryptographic hashing puzzle that is used in these networks to determine fair entitlement for block rewards. Due to the dominance of the electricity cost, these protocols give unfair advantage to entities that operate from world locations that have lower electricity costs compared to tech enthusiasts residing in major cities where energy costs are higher. The high energy costs require a resultant increase in block mining to offset costs.

At its core, the Spacemesh consensus protocol achieves consensus between network nodes without creating a race between nodes in order to solve a cryptographic puzzle that incentivizes all nodes to constantly execute computations that cause considerable energy waste. Compared to a PoW, PoST requires substantially less energy use, as the “difficulty” can be increased by extending the time period over which data is stored without increasing computation costs and energy use. This is our core innovation: a protocol that achieves similar or, in some aspects, improved properties in comparison to PoW-based consensus protocols, while being ecologically friendly.

While the production process of computer storage is not by itself environmentally friendly, the total environmental impact of a network that is designed to utilize an existing resource is significantly less than a network where each node consistently consumes large amounts of power.

[11] It is difficult to estimate the exact energy costs for the Bitcoin network as most mining happens in dedicated data centers running custom ASIC farms and about 30% to 40% of data center operational costs are cooling expenses that go beyond the electricity spent to power servers. https://digiconomist.net/bitcoin-energy-consumption#validation http://www.wired.co.uk/article/how-much-energy-does-bitcoin-mining-really-use A quick back of the envelope calculation considering current energy prices and the cost and performance of a modern Bitcoin ASIC mining rig, excluding any cooling expenses and considering the current average network hashing rate, amounts to about $1.8M USD per year in electricity costs. http://blog.zorinaq.com/bitcoin-electricity-consumption/

[12] At 0.12 USD per kWh - average US consumer electricity price. 139 th/s total network hashrate according to https://ethstats.net/ on 1/29/2018.

High Transaction Fees

High transaction fees are a major blockchain issue for all cryptocurrencies, especially for cryptocurrencies like Spacemesh, which are designed to be used as means of payment rather than merely as a store of value.

Given the current number of daily transactions, each Ethereum transaction costs about $4.50 USD. Some estimates for the price of a Bitcoin transaction place the actual cost at $58 USD per transaction.[13]

In a steady-state cryptocurrency, the compensation for maintaining the network through mining and other infrastructure services comes entirely from transaction fees. This means that the total amount of fees awarded must be at least the total cost of network maintenance, and still higher for validators to be able to profit from the services they provide to the network. Denote c the total maintenance cost and n the total number of transactions; the per-transaction fee is lower-bounded by c/n. In consequence, in order to reduce transaction fees one must either reduce the cost of maintenance, such as by replacing PoWs with a cheaper alternative, or increase the transaction rate, which requires high scalability.

Spacemesh manages to both reduce the cost of network maintenance (c) and increase the transaction rate (n), allowing a much lower transaction fee than is possible for PoW-based blockchains.

As Meshcoin aims to be a means of payment and not a store of value or a speculative financial security, a successful Spacemesh network translates to a high daily transaction volume. With such a volume, it is possible to keep transaction fees low while still making it profitable for validators to participate in the network. This is due to the fact that validators will not need to generate revenue to offset investment in dedicated mining hardware. The challenge ahead of us is to find a transaction fee design that satisfies these conditions.

[13] 481 kWh per transaction at 0.12 kWh consumer US power bill rate. https://digiconomist.net/bitcoin-energy-consumption

Section 2 - Spacemesh Technology Overview

Spacemesh protocol replaces the notion of a blockchain with a mesh, a layered directed acyclic graph (DAG) that provides irreversibility and economic finality--and is race free. Spacemesh is the first provably secure cryptocurrency protocol that doesn't involve PoW or PoStake, assuming honesty of a majority of the PoST storage.

When a Bitcoin miner verifies and includes certain transactions in a block that she creates, she then 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.[14] Indeed, this behavior appears to be widespread among Bitcoin miners, as some miners lost a significant amount of funds due to the BIP66 softfork.[15]

In Spacemesh 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 Spacemesh relative to blockchain protocols.

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.[16] Such behavior would damage the performance of the Bitcoin system from the point of view of its users, as transactions would be confirmed at a slower overall rate. In Spacemesh, 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.

In Bitcoin, rational and malicious parties may benefit from offering bribes to other miners, by sending in-band messages in an anonymous fashion.[17] A rational miner may fork a high-value block by collecting only some of its transactions, therefore 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 tx0 in the honest chain and offer high fees for blocks that include another transaction that conflicts with tx0, subsequently incentivizing rational miners to work on a fork. Similar 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.

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

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

[16] On Bitcoin and red balloons - https://arxiv.org/abs/1111.2626

[17] J. Bonneau, "Why buy when you can rent? - bribery attacks on bitcoin-style consensus," in Financial Cryptography Bitcoin Workshop, 2016. K. Liao and J. Katz, "Incentivizing double-spend collusion in bitcoin," in Financial Cryptography Bitcoin Workshop, 2017.

In Spacemesh, the fees are shared among all the blocks of a layer; additionally, conflicting transactions do not invalidate referencing blocks, which renders these kinds of bribe strategies ineffective.

Proof of Space Time (PoST)

The concept of PoST, first published in 2016, was pioneered by Professor Tal Moran, SpacemeshOS chief scientist, and Dr. Ilan Orlev.[18] An updated paper that covers theoretical and practical improvements to PoST, an overview of the PoST protocol used in Spacemesh, and the protocol security proofs is in development and currently under academic peer review.

A PoST allows a prover to convince a verifier that she spent a “space-time” resource - storing data in space over a period of time. Formally, we define the PoST resource as a trade-off between CPU work and space-time. Under reasonable cost assumptions, a rational user will prefer to use the lower-cost space-time resource over CPU work. Compared to a PoW, a PoST requires less energy use, as the “difficulty” can be increased by extending the time period over which data is stored without increasing computation costs.

Our definition is similar to “Proofs of Space”[19] but, unlike the previous definitions, takes into account amortization attacks and storage duration. Moreover, our protocol uses a very different but simpler technique, making use of the fact that we explicitly allow a space-time tradeoff, and does not require any non-standard assumptions beyond random oracles.

Using a PoST protocol, provers show that they have reserved a portion of their storage for a specified time, hence “space-time.” The reserved storage cannot be used for anything else during that time, and in particular, cannot be used for other PoST proofs. For example, a user can prove that she reserved 100GB for a month. Specifically, the PoST actually proves the statement, “Either I stored 100GB for a month, or I wasted a corresponding amount of CPU work”; however, we set the parameters, so it is less expensive to utilize storage than it is to use CPU work.

All permissionless consensus protocols must have some limit on the power of the adversary, in order to avoid theoretical impossibility results. In PoW-based protocols, the limit is the amount of CPU throughput that the adversary can control. For PoStake-based protocols, it is limited by how much money is controlled, while in Proof of Space and PoST-based protocols, control is based on storage space.

The main difference between Proof of Space and PoST is that in Proof of Space, the “time” element is implicit. For our purposes, we can think of Proof of Space protocols as PoST protocols with a fixed initialization cost that depends linearly on the space used. Therefore, in

[18] Rational Proofs of Space-Time https://eprint.iacr.org/2016/035.pdf
[19] Proofs of Space https://eprint.iacr.org/2013/796

order to make storage preferable to work, Proof of Space protocols require either enormous amounts of storage or proofs repeated at short intervals.

For the Spacemesh implementation of the PoST initial setup phase, we choose a GPU-friendly and a memory-hard hash function that makes it uneconomical to build a dedicated ASIC to perform the task, since a modern GPU is close to the ideal ASIC for this function.

Proof of Elapsed Time (PoET)

A drawback of PoST compared to PoW is that PoST is not "self-contained." A PoW proof is a single string that can be verified algorithmically "on its merits." In contrast, a PoST proof has two phases: a commitment phase and a proof phase, each of which generates a string. Given these two strings (we name *commit* and *π*), the verifier can be convinced that the prover stored X GB of data for a length of time equal to the interval between the generation of the commit and the time at which the prover learned the challenge for the proof phase.

In reality, the length of this interval is unknown but is critical towards understanding how much of the resource was spent: storing X GB for a minute is much cheaper than storing it for a month. To solve this problem, we have introduced Proof of Elapsed Time (PoET).

A PoET is a single string that proves at least T time must have passed from the time the initial challenge was learned. We use the PoET to convert a PoST into a single, self-contained proof. The basic idea is that the commit string from the PoST is used as the challenge for the PoET; then the PoET is used as a challenge for the proof phase of the PoST. The final, complete proof consists of the commit string, the PoET and π, the PoST proof string. If T, the time parameter for the PoET, was one month, the verifier can be convinced that the prover must have reserved the data for at least a month, since it could not learn the outcome of the PoET before then.

The Spacemesh Foundation will provide and maintain PoET public servers as a free public utility for the network. These servers will provide a PoET service to network nodes. Nodes do not need to trust the PoET servers as they can efficiently and independently verify the correctness of the output of the PoET servers. Note that the Spacemesh protocol will accept *any* valid PoET, not just those generated by the Spacemesh Foundation PoET servers. Thus, nodes can choose to use alternative servers for PoET (or generate the proofs themselves).

However, the cost-per-proof for a PoET can be reduced enormously by increasing the number of PoETs per server since each additional proof adds only a small number of hash evaluations to the total cost. Thus, as long as the Spacemesh foundation PoET server is available and free, there is no incentive to create new servers.

We are also considering introducing an incentive system for providing PoET proofs and are going to fully open source the PoET server source code, so that anyone, without restriction, will be able to run a PoET server and receive compensation in Meshcoins for providing this utility to the network.

Non-Interactive Proof of Space-Time (NIPST)

A PoST, formally defined in “Proofs of space-time and rational proofs of storage,”[20] is a two-phase protocol. In the first phase, the prover commits to store some random data, with respect to a specific ID (seed). In the second phase, the prover receives a challenge, and proves that the data is still stored (or was recomputed). Unlike a PoST, a NIPST has only a single phase. Given an id, a space parameter S, a duration D and a challenge ch, a NIPST is a single message that convinces a verifier that the prover expended S · D space-time after learning the challenge ch.

NIPST API

A NIPST scheme consists of three algorithms. See fig. 1.3.

- InitializeNIPST(id: seed ID, S: space)
- GenerateNIPST(id: seed ID, S: space, D: duration, ch: challenge , σ)
- VerifyNIPST(id: seed ID, S: space, D: duration, ch: challenge, π: proof)

Constructing a NIPST from a PoST and a PoET

Given a PoST and a PoET, the construction of a NIPST is straightforward: the PoST commitment will serve as the PoET challenge and the PoET will serve as the PoST challenge. The proof itself consists of the PoST commitment, the PoET proof and the PoST proof. Note that the PoST must have sigma-protocol form: the initialization must consist of the prover sending a single “commitment” message, and the execution phase of the verifier sending a random challenge and the prover responding with a single proof message (the API for a sigma-PoST is summarized in fig. 1.1, and for the PoET in fig. 1.2).

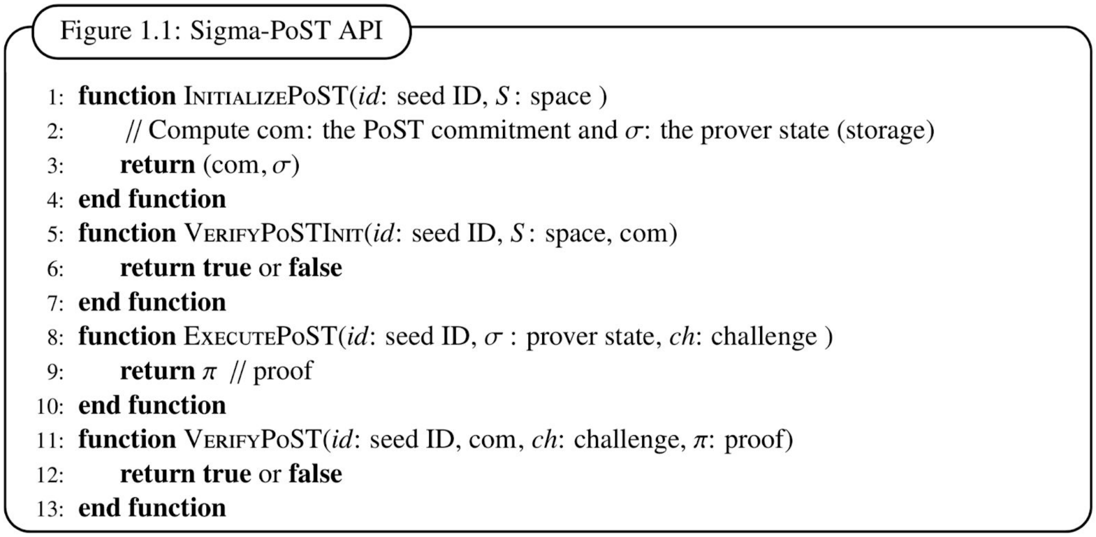

Figure 1.1: Sigma-PoST API

```
function InitializePoST($id$: seed ID, $S$: space )
    // Compute com: the PoST commitment and $\sigma$: the prover state (storage)
    return (com, $\sigma$)
end function
function VerifyPoSTInit($id$: seed ID, $S$: space, com)
    return true or false
end function
function ExecutePoST($id$: seed ID, $\sigma$ : prover state, $ch$: challenge )
    return $\pi$  // proof
end function
function VerifyPoST($id$: seed ID, com, $ch$: challenge, $\pi$: proof)
    return true or false
end function
```

[20] https://eprint.iacr.org/2016/035

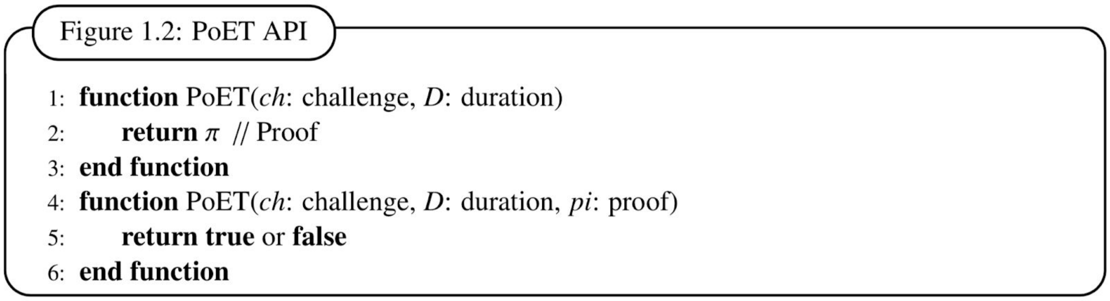

Figure 1.2: PoET API

```
**function** PoET($ch$: challenge, $D$: duration)
    **return** $\pi$ // Proof
**end function**
**function** PoET($ch$: challenge, $D$: duration, $pi$: proof)
    **return true** or **false**
**end function**
```

Intuitively, an adversary cannot guess the commitment value without first generating the PoST data (otherwise it could either break the binding property of the commitment or the initialization work property of the PoST), and cannot guess the PoET outcome before the elapsed duration since learning the PoST commitment (otherwise it could violate PoET security).

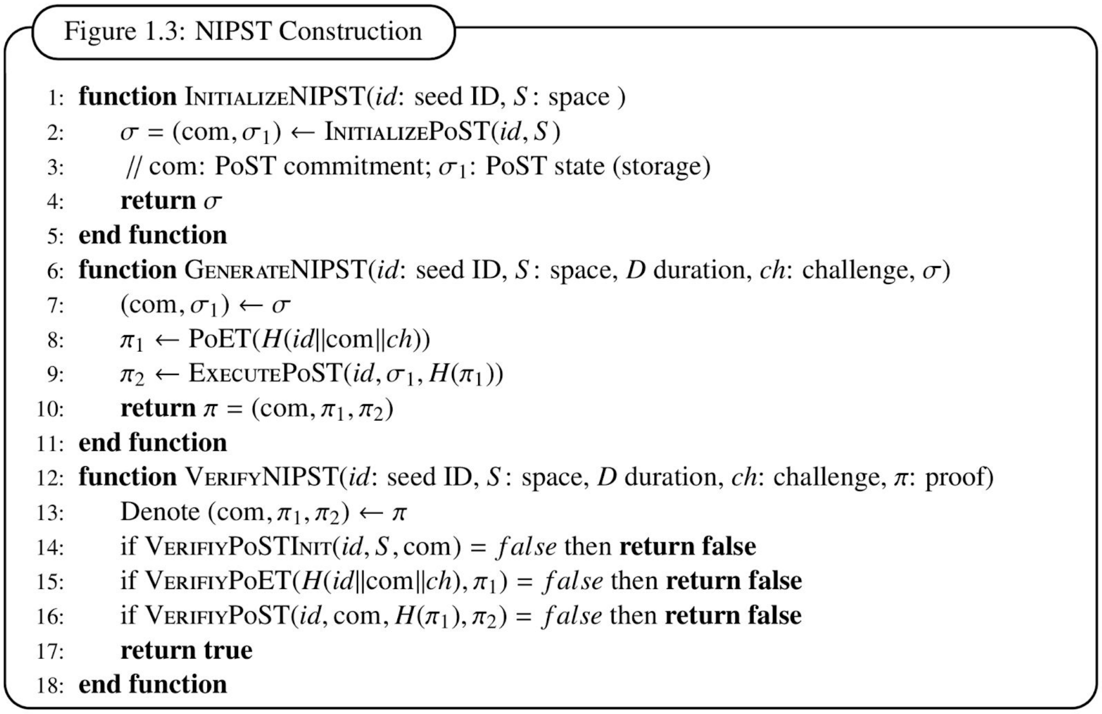

Figure 1.3: NIPST Construction

```
**function** InitializeNIPST($id$: seed ID, $S$: space )
    $\sigma = (\text{com}, \sigma_1) \leftarrow$ InitializePoST($id, S$)
    // com: PoST commitment; $\sigma_1$: PoST state (storage)
    **return** $\sigma$
**end function**
**function** GenerateNIPST($id$: seed ID, $S$: space, $D$ duration, $ch$: challenge, $\sigma$)
    $(\text{com}, \sigma_1) \leftarrow \sigma$
    $\pi_1 \leftarrow$ PoET($H(id\|\text{com}\|ch)$)
    $\pi_2 \leftarrow$ ExecutePoST($id, \sigma_1, H(\pi_1)$)
    **return** $\pi = (\text{com}, \pi_1, \pi_2)$
**end function**
**function** VerifyNIPST($id$: seed ID, $S$: space, $D$ duration, $ch$: challenge, $\pi$: proof)
    Denote $(\text{com}, \pi_1, \pi_2) \leftarrow \pi$
    if VerifiyPoSTInit($id, S, \text{com}$) = $false$ then **return false**
    if VerifiyPoET($H(id\|\text{com}\|ch), \pi_1$) = $false$ then **return false**
    if VerifiyPoST($id, \text{com}, H(\pi_1), \pi_2$) = $false$ then **return false**
    **return true**
**end function**
```

The Spacemesh Protocol - An Informal Description

1. Consensus, Contracts and Crypto-currencies

The technical descriptions of cryptocurrencies usually describe 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 supply of coins 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:

1.1 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 must guarantee several properties:

- **Consensus on transaction order:** All honest parties must agree on the order in which transactions appear on the ledger.[21]
- **Irreversibility:** The ledger cannot be modified—only extended with additional entries.[22]
- **Liveness:** The ledger grows over time (i.e., an adversary can’t prevent some new transactions from eventually being added to the ledger).[23]
- **Fairness:** The fraction of honest transactions in the ledger is proportional to the honest users’ resources. In particular, this means adversaries can’t force the ledger to include only their own transactions.

For the purposes of this layer, transactions are opaque strings—there’s no need to interpret them in any way. This is an important property, since it allows us to “modularize” crypto-currencies—the underlying ledger can be replaced without changing the layers above it. (For optimization purposes, a cryptocurrency built on top of the ledger can add additional restrictions, in order to prevent clearly invalid transactions from entering the ledger in the first place, but we ignore that here).

1.2 Consensus Computer

The consensus computer 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 be used to manipulate them. At an abstract level, one can think

[21] 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.

[22] Like the consensus property, irreversibility is only required to hold for “sufficiently old” transactions.

[23] This property is sometimes called “chain growth.”

of the consensus computer as describing a deterministic computer whose only input is the list of transactions. By “running” this computer over the list, we can compute the “current state.”

The ledger layer guarantees that all honest users see the same list of transactions. Since the state machine is deterministic, they will reach the same state after running it on the transactions in the ledger (of course, there may be disagreements on the most recent entries in the ledger, so honest users are only guaranteed to completely agree on “sufficiently old” states).

1.3 Economy

The economy layer describes how coins are created and destroyed, how monetary policy is determined and implemented. Examples of questions addressed in this layer are “is the supply of coins capped?”, “do old coins expire?” and “how do we allocate the initial distribution of funds?”

2. Preliminaries

The purpose of the spacemesh consensus protocol is to implement the ledger layer of a crypto-currency. In this document, we don’t discuss the layers above (although of course a complete crypto-currency implements all of them).

We implement the ledger by implementing consensus on a **blockmesh** - a layered DAG in which each node is a *block*, each block is a member of an indexed layer, and edges only point “backwards”: i.e., a block in layer i can have an edge to a block in layer j only if $j < i$.

2.1 From Blockmesh to Ledger

A ledger must return a list of *transactions*. How do we get from a blockmesh (where each block may contain multiple transactions) to a ledger?

Block IDs We assume each block has a unique ID that can be computed from the contents of the block. W.l.o.g., the ID can be the entire contents, but in practice we use a collision-resistant hash of the contents. For a block B, we denote id_B the ID of block B.

Total Order on Blocks We use the following rules to define a full ordering between blocks. Let A be a block in layer i and B a block in layer j. Then the order between A and B is the lexicographic order between the tuples (i,id_A) and (j,id_B), where the first element is most significant (i.e., order first by layer, then by ID).

Total Order on Transactions Each block contains an ordered list of transactions. These lists are not mutually exclusive—the same transaction can appear in multiple blocks. To get a full order on transactions from the blockmesh, we consider only the first block (according to the total block order) in which each transaction appears.

If transactions s and t appear in different blocks, they are ordered according to the block ordering. If they appear in the same block, they are ordered according to their indices in the block.

From Total Order to Consensus on Complex State The layer above consensus in the crypto-currency protocol stack is the state machine: in this layer the participants agree on the state of a virtual computer and its continuing evolution. Once users agree on a total order of *transactions*, the agreement on the current state of the virtual computer is immediate.

Starting from genesis (where the virtual computer has, by definition, an agreed initial state), every transaction modifies the state deterministically. To compute the current state, users execute all transactions, in their agreed order.

Of course, since users are guaranteed to agree only on a *prefix* of the transaction list (there may be disagreements about recently-posted transactions), they are also guaranteed to agree only on a prefix of the state evolution. That is, users may disagree on the state after running the *entire* list of transactions if they disagree on the order of the recent ones.

2.2 Communication Model and Security Assumptions

Our communication model is basically identical to that of existing permissionless consensus protocols: all network nodes are connected via a “gossip network.” Messages sent via the gossip network eventually reach all connected nodes; honest nodes that receive a new message will gossip it to their neighbors. We assume a known, bounded delay δ on the communication over the gossip network: every message seen (or sent) by an honest node will be seen by *all* honest nodes within time δ (if a node is disconnected from the network, it no longer counts as “honest” in terms of our security assumptions).

We don’t assume anything about the network beyond the bounded delay. In particular, we must tolerate an adversary that can schedule messages arbitrarily subject to the delay bound (e.g., it can ensure that one honest node receives a message immediately, while another must wait time δ before receiving the message—even if the message was sent by an honest node).

Security Assumptions The security of the Spacemesh protocol relies on several assumptions; some are limits on the power of the adversary compared to honest users, and some on the power of the adversary to solve certain computational problems at all:

- **Honest super-majority of resources:** Our main non-standard assumption is that honest nodes control a super-majority of the space-time resources devoted to this network. The size of the required majority depends on protocol parameters, but think of it as ⅔.
- **Standard cryptographic assumptions:** We also assume that the adversary satisfies various standard cryptographic assumptions: for example, we rely on public-key signature schemes, as well as slightly more exotic cryptographic primitives, such as verifiable random functions.
- **Random oracles:** The Spacemesh protocol uses a “random oracle” in its formal description and analysis. We instantiate the oracle using a cryptographic hash function (such as SHA-256). While no hash function has all the properties of a random oracle, we assume the adversary cannot exploit the internal structure of the hash function to attack the protocol (this assumption is made by all existing crypto-currency protocols based on PoW).

Space-Time Resources The Spacemesh protocol replaces proof-of-work (PoW) with proof-of-space time (PoST). Whereas a PoW proves that a user has performed a certain amount of CPU work, a PoST proves that a user has allocated a certain amount of storage space for a certain amount of time (e.g., stored 100GB for one week).[24]

We use a non-interactive version of the PoST protocol that, syntactically, is similar to a PoW: a user receives a challenge, stores data for the specified amount of time following the challenge, and then computes a proof that the data is still stored correctly. Given the challenge and the proof, any verifier can check that the space-time resources were expended as required.

2.3 Challenges in Achieving a Race-Free Consensus

The underlying gossip network ensures that all honest parties eventually receive every block transmitted over it. Thus, they agree on which blocks were transmitted and on their contents. As we describe in section 2.1, once we agree on which blocks should be included in the blockmesh, agreeing on the order is straightforward.

The main challenge is that, while nodes do agree on which blocks were transmitted, they may not agree on the timing and order in which blocks were received. However, the timing information is critical in determining which blocks should be included—otherwise an attacker can wait until consensus is achieved, and then publish new blocks that claim to belong to “old” layers. If these blocks are not rejected, the irreversibility property of the ledger is violated.

Naively, we can simply use a “cutoff” time for each layer. That is, a block will be accepted to layer *i* only if it was received before time t_i. If we instruct honest nodes to switch to layer-$i + 1$ sufficiently early before the cutoff, we can ensure that every honestly-generated block is accepted by every honest node.

The problem with the naive scheme is that the adversary can publish blocks late; for example, just before the cutoff. In this case, some honest nodes might receive the block before the cutoff, and some afterwards. The consensus protocol must ensure that all honest nodes agree on whether these “late” blocks should be included in the blockmesh.

3. The Spacemesh Protocol

The Spacemesh protocol follows the basic structure of Meshcash framework.[25] At a high level, each block “votes” about the validity of all previous blocks, and nodes decide whether or not a block is valid (i.e., should be included in the block-mesh) by counting the votes (for blocks that are sufficiently “old” to have a significant vote count) or by using a separate, off-chain byzantine-agreement protocol (for recent blocks).

[24] Formally, users prove that they either used space-time or performed CPU work, but we set the parameters so that the cost of performing the CPU work is much greater than just storing the data.

[25] I. Bentov, P. Huba ́cek, T. Moran, and A. Nadler. Tortoise and hares consensus: the meshcash framework for incentive-compatible, scalable cryptocurrencies. *IACR Cryptology ePrint Archive*, 2017:300, 2017. http://eprint.iacr.org/2017/300 .

3.1 Node Registration and Block Eligibility

Unlike a PoW, which can be thought of as a “lottery,” the time to generate a valid PoST proof is basically deterministic. Thus, we can’t use the bound on the proof generation rate to limit the number of blocks generated in each time period. Instead, we require every node to register their ID by posting a special registration transaction. Thus, in every layer we have an explicit list of the nodes that could potentially generate a block in that layer.

We select a subset of eligible blocks by applying a “random function” to the IDs of the registered nodes, and selecting only those who pass the threshold. The threshold itself is set so that the expected number of eligible nodes remains constant (similar to the difficulty adjustment in Bitcoin).

We set parameters so that generating a new ID is costly, and the expected time between eligible blocks for a given ID is fairly small, so rational adversaries won’t bother “grinding” on IDs in order to create eligible blocks.

3.2 Types of Validity

We consider two kinds of validity: syntactic validity is what can be determined entirely by the contents of the block (and the blocks reachable from it in the DAG). This includes things like whether the PoST is valid. We call any validity rule that isn’t syntactic a contextual validity rule. This includes any 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), as well as local information such as the time at which blocks were received. For a block to be valid, it must be *both* syntactically and contextually valid.

3.3 Syntactic Validity Rules

In order for a block B to be syntactically valid in layer i, it must satisfy the following properties:

- ○ **Node Eligibility:** The Node ID embedded in B must be eligible to generate a block in layer *i*.
- ○ **PoST Validity:** The header of block B must contain a valid proof that sufficient space-time resources were expended by the node that generated it.
- ○ **Recursive Syntactic Validity:** Every block in B’s visible mesh must be syntactically valid.
- ○ **Transaction Syntactic Validity (optional):** Every transaction included in the block is syntactically valid. Syntactic validity of transactions can depend only on their contents, and not on the order relation to other transactions (this is just an optimization; it is also okay to depend on the order of sufficiently old transactions, for which consensus and irreversibility already hold with high probability).

3.4 Contextual Validity Rules

The contextual validity rules are more complex. At a high level, the idea is as follows: For recent blocks (i.e., blocks that are less than hdist layers in the past) we run a separate “hare protocol” (in this case, one that performs off-chain byzantine agreement) to determine the contextual validity of the block.

For older blocks, we let succeeding blocks “vote” about their validity. The validity rule is recursive: every block “counts” the votes of all the preceding blocks, and uses that to compute its own vote. If we don’t care about maliciously-generated blocks, this strategy would suffice. However, 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 (using a small fraction of the honest party’s resources to keep the two sides “balanced”).

To overcome this type of attack we use an off-chain byzantine agreement protocol between the parties who published a block in the previous layer. This guarantees that honest parties agree on every recent layer, and thus that the voting margins for older blocks will always increase.

While the protocol as described above will guarantee our desired security properties, it could fail catastrophically if our security assumptions are violated (e.g., if the adversary manages to gain temporary control of a majority of the space-time resources).

To ensure the protocol can “self-heal” from this type of extremely powerful attack (or very low probability accident), spacemesh includes an additional layer of protection: it guarantees eventual consensus even if the hare protocol completely fails. In order to do this, we use an additional tool: a “weak common coin” that is used to decide validity in some cases where an attack has been detected.

3.4.1 Coin Flips and Layer Challenges

Every layer is associated with two “random” values:

- k-layer challenge: The challenge for layer *i*, denoted ch_i, is an *unpredictable* bit string; the probability that the adversary can output ch_i before the start of layer $i - k$ is negligible. On the other hand, at the start of layer *i*, all honest parties must agree on the value of ch_i except with negligible probability. This is the challenge used for producing PoST proofs, for example.

- *p*-coin flip: We denote $coin_i$ the result of the coin flip for layer *i*. No adversary can predict coin *i* with probability more than $1 - p$ before the start of layer *i*. Moreover, with probability at least *p*, all honest parties agree that $coin_i = 0$ at the start of layer *i*, and with probability at least *p* all honest parties agree that $coin_i = 1$ at the start of layer *i*. The coin flip is used to guarantee eventual consensus in cases where the hare protocol fails (due to temporary violation of our underlying majority assumptions, or a very-low probability “accident”).

We implement the challenges and coin-flips using the blockmesh itself (we defer the detailed descriptions to the full version of the paper).

3.4.2 Late Blocks

The first level of contextual validation is filtering out late blocks. A layer-i block is late if it was received more than δ time after the start of layer i (according to the node's local clock).

Note that honest nodes will always accept honestly-generated blocks, since those are transmitted at the start of layer *i*, and are guaranteed to be received by all honest nodes within time δ. On the other hand, honest nodes may not agree on the validity of maliciously-generated blocks, since those may be transmitted at any time.

3.4.3 Voting Rules

Hdist: The number of recent layers whose validity is determined by the Hare protocol (e.g., when hdist = 2 the Hare protocol decides on the validity of layers $t - 2$ and $t - 1$ at time t).

In order to be contextually valid at time t, block B in layer $i < t$ must be syntactically valid and satisfy two properties:

- **Majority rule:** Evaluating this rule depends on the value of $t - i$

  Case 1. If $t - i \leq$ hdist, the contextual validity is determined by executing the hare validity function.

  Case 2. If $t - i >$ hdist, the contextual validity is determined by executing the *tortoise* validity function.

- **Unique ID rule**: Every node ID can generate at most one block per layer. If two blocks in the same layer have the same node ID and both are considered valid according to the majority rule, only the block with the lowest ID (lexicographically) is considered valid.

Note that a block can be contextually valid even though it is late (otherwise honest nodes might never reach consensus on blocks that appear late to a minority of the honest nodes).

3.5 Simplified Tortoise: Complete Voting Transcripts

Here, we describe a simplified (but communication-inefficient) version of the Tortoise protocol. In this version of the protocol, each block includes a "vote" (accept/reject) for every block in the visible mesh. (In the full version, we will show how to optimize the protocol so that this explicit list can be omitted.)

In a nutshell, to determine the validity of a block B, the tortoise validity function counts votes on B's validity from every block in the succeeding layers; if there's a significant majority in either direction (i.e., for or against), the majority rules. Otherwise, it takes a second poll, counting only the last layer. Again, if there's a significant majority, the majority rules. Otherwise, the node uses the value of the common coin to decide on validity.

An important point is that syntactically-valid blocks can get a vote even if they're not contextually valid. This prevents recursive loops in computing validity. A more subtle issue is that each node id is only allowed one vote per layer. If it attempts more votes, they are all considered invalid (this is because generating many PoSTs for the same id in the same layer does not require significantly more resources than generating one).

3.5.1 Non-Voting Blocks

One of the main differences between basing security on PoW and basing it on space-time is that, unlike a PoW, the space-time resource can't bound to a specific "vote" (since we need to reuse the same storage space for multiple blocks). This means that, potentially, a malicious adversary can attack in the following way:

1. At time t, generate 2 layer-t blocks with the same node ID, one that votes x and the other $\neg x$. Note that this doesn't require extra resources. Publish block x.
2. At time $t + s$, publish block $\neg x$.

Looking only at syntactic validity, both blocks are equally valid. Thus, in counting the votes, we also consider the time at which blocks were received, and ignore late blocks.

Unfortunately, this causes another problem: for maliciously-generated blocks, honest nodes might not agree on which blocks were late. Using the naıve rule of simply ignoring votes from late blocks, a single malicious node ID can cause a difference of 2 in the vote counts of honest users by "double voting": the malicious node generates two blocks, one voting *for* and the other *against* block B.

One honest user will receive the *for* block early and the other late, while another honest user receives them in opposite order. In order to reduce the power of the adversary to launch such an attack, we introduce the following rule:

No double-voting rule: If two blocks B, B' are both in layer i and have the same Node ID, and both were received before time $i + \delta$, then both are considered *invalid*. Note that the vote is invalidated even if one of the blocks is late (this ensures that if any honest node counts the node ID as voting in one direction, no honest node will count it voting in the other direction).

3.5.2 Choosing the Thresholds

Correctly choosing the global (θ_G) and local (θ_L) vote thresholds is critical for the security of the protocol. In order for security to hold, we need the following properties to be satisfied:

1. **Self-Perpetuating Local Agreement**: If all honestly generated blocks in layer t vote X about block B, then in the view of *all* honest parties at time $t + 1$ it holds that $v_t^X > \theta_L T_{ave}$.
2. **α-Local to Global Agreement:** For every honest party P, every block B at layer i, and every intermediate vote total for B at layer s, there exists $t \geq s$ such that if an α-fraction of honestly-generated blocks in layers $s, \ldots, t$ vote X about block B, then the vote total for block B according to P is greater than $\theta_G(t - i)T_{ave}$.
3. **No Local Disagreement:** If, in the view of some honest party P, $v_t^X > \theta_L T_{ave}$, then there does not exist an honest party P' such that $v_t^{\neg X} > \theta_L T_{ave}$.

4. **No Strong Global Disagreement:** If, in the view of some honest party *P*, $v_t^X > \theta_G(t - i)T_{\text{ave}}$, then there does not exist an honest party *P*' such that $v_t^{\neg X} > \theta_G\,(t - i)T_{ave}$.
5. **Local *α*-Catch-up:** If, in the view of some honest party *P*, $v_t^X > \theta_L T_{\text{ave}}$, then the fraction of honest blocks in layer *t* that voted *X* is at least *α*.

3.5.3 Security Intuition

For security, we rely on the fact that a large-enough majority of the "active" ids are honest; thus, the *eligible* ids—a large random subset of the active ids—will also have approximately the same majority of honest ids (with high probability).

Standard operation In standard operation, the hare protocol guarantees that all honest parties agree on the validity of all recent blocks. Initially, no block reaches the global vote threshold, so the validity of blocks is decided by the last-layer voting rule. Since all honest parties agree, all honestly-generated blocks will vote the same way in the first non-hare layer, hence by "Self-Perpetuating Local Agreement" (property 1) of the local threshold, all honest parties will continue to agree in all succeeding layers. By property 2, eventually all honest parties will have global agreement. Once this occurs, reversing history will require reducing the honest lead (compared to the global threshold) but the probability that this will happen is negligible (and vanishes exponentially quickly in the number of layers, using an argument similar to the Bitcoin "race" analysis.

Self-healing from limited failures Suppose all honest parties disagree on at most $q \cdot (1 + \varepsilon) \cdot T_{\text{ave}} \cdot x$ votes in any layer interval of length *x*. We consider the following cases:

Case 1. **No honest party passed the global threshold**. In this case, all honest parties use the local threshold, and then the coin. If any honest party passed the local threshold, then whenever the coin matches, all honest parties will locally agree, and the combination of properties 1 and 2 will ensure eventual global agreement. (if no honest party has passed the local threshold, then all honest parties agree on the coin).

Case 2. **Some honest party has passed the global threshold with vote *X*.** By our threshold settings (property 4), this implies that no honest party passed the threshold with vote ¬*X*. We consider the subcases:

Case 2.1 *All honest parties passed the local threshold with vote X*. In this case, it's just a matter of time until all honest parties pass the global threshold (due to properties 1 and 2).

Case 2.2 *At least one honest party did not pass the local threshold with vote X, but no honest party passed the local threshold with vote ¬X*. In this case, honest parties either agree with *X* due to passing the local threshold, or use the coin. Hence, if the coin succeeds and gives outcome *X*, all honest parties will agree, putting us back in case 2.1. Since this happens with constant probability, the probability that it will not occur within *k* rounds is exponentially small in *k*.

Case 2.3 *At least one honest party passed the local threshold with vote¬X*. In this case, by property 5, the fraction of honest blocks voting ¬*X* must be at least α. If a layer of this type occurs too many times consecutively, by property 2 eventually all honest parties will reach global agreement on ¬*X*.

For every case but case 2.3, the number of times the case can appear (in total) without "falling through" to consensus is bounded. Since case 2.3 can't appear too many times consecutively, eventually consensus must be reached.

Free Open Source Software (FOSS)

Our goal is to ensure that Spacemesh becomes a free internet protocol for blockchain networks. To achieve this goal, we are developing Spacemesh as a free, open-source software (FOSS)[26] under the permissive MIT license. We also ensure that Spacemesh is free from patent and from proprietary closed-source software. All of the various Spacemesh software systems will be fully openly sourced, with all novel protocols and proofs used by Spacemesh publicly published, thus encouraging review and open community debate.

We also believe that software applications such as the Spacemesh node, designed to be installed by people on their home PCs and particularly blockchain consensus software, must be fully open source so that independent security researchers can fully review the code and ensure that is not malicious in any way. We stand behind the code to actually faithfully implement the published protocols; thus advanced users may easily compile and run directly from source code.

Additionally, we believe that developing Spacemesh as FOSS is important for the creation of a healthy and vibrant global community around the project, in which we welcome collaboration and encourage contributors to take a major role in the early adoption and success of the project.

Additional Resources

Spacemesh website https://spacemesh.io
Go-spacemesh project https://github.com/spacemeshos/go-spacemesh
Roadmap https://github.com/spacemeshos/go-spacemesh/wiki/Roadmap
FAQ https://github.com/spacemeshos/go-spacemesh/wiki/Spacemesh-FAQ
Join the conversation https://gitter.im/spacemesh-os/Lobby
Spaemesh on Twitter https://twitter.com/teamspacemesh

[26] https://github.com/spacemeshos/go-spacemesh