Alternative Sybil Resistance Methods

Marcus Daly (mrd2194), Nathan Cuevas (njc2150),

Griffin Klett (gk2591), Lynn Zhu (jz2969)

COMS-6998 Foundations of Blockchains

Professor Roughgarden

December 17, 2021

Abstract

With growing energy usage and potential positive externalities such as file storage, research on a myriad of Proof of Work alternatives has been published over the past decade. In this paper, a summary of benefits, tradeoffs, and mechanisms of some of the more prominent and promising sybil resistance mechanisms is examined. Additional analysis focuses on the real-world implementations of such blockchains, as well as a larger comparison of energy usage, vulnerabilities, and decentralization.

Background

One of the most desirable properties of blockchains is decentralization; in other words, the responsibility of maintaining and appending blocks onto the blockchain is not limited to a central authority. Current implementations of blockchains achieve this by pseudo randomly selecting a leader among a network of full nodes. A truly decentralized protocol must be "permissionless," meaning that it should be easy for anyone that wishes to participate as full node to join the network. A permissionless blockchain must also be "sybil-resistant," meaning that a single actor can't run a large number of full nodes to essentially control a large fraction of the network. Currently, the most successful blockchain protocols such as Bitcoin and Ethereum solve the "sybil-resistance" problem by allowing the probability of a full node being selected as a leader to be proportional to how much the node has invested a scarce resource in the system, making it expensive for a single actor to control a large fraction of the network.

In its current state, the most popular "sybil-resistance" solutions are Proof of Work (PoW) and Proof of Stake (PoS). In a PoW blockchain such as Bitcoin and Ethereum, a full node is appointed the leader (or miner) of a block if she is the first in the network to submit a PoW solution. A PoW solution is an easily and transparently verifiable proof that the miner has solved a difficult cryptographic puzzle implying that she has invested her resources and energy to get this solution. In a PoS blockchain like Solana or Cardano, a full node proves what she has invested in a network by staking some number of coins, which will be slashed if the node is provably dishonest. There is much debate about whether the current solutions are optimal. For instance, PoW requires a massive amount of energy, most it not even being used to generate the new block. PoS has a staking system that has no external dependencies, meaning the amount invested in the system doesn't depend on anything outside of the protocol. There is ongoing research in finding the optimal "Proof of X" mechanism that doesn't have negative side effects while being able to prove securely and reliably what a full node has invested in the system. This paper will explore a few of the most promising means to achieve this.

Motivations

Why Alternatives to PoW?

PoW has received much scrutiny since its release, and deservingly so, as PoW-based Bitcoin is not only the first blockchain, but (as of 2021) has maintained status of largest cryptocurrency by market capital since its inception. Many of the arguments against the use of PoW are very convincing, so much so that significant research has been invested in alternatives to PoW; here we present the top arguments in this regard.

Energy Consumption

Perhaps the most popular argument against the use of PoW is its energy consumption. Using data from blockchain.com, in December of 2021 total hashrate of Bitcoin is approximately $160 \cdot 10^{18}$ $\mathrm{H/s}$ which equates to $5.0 \cdot 10^{27} H/\mathrm{year}$. Assuming the most efficient hardware is used (ASICs which have an efficiency of about $10^{9} H/J$). Then the annual energy usage can be estimated to $5 \cdot 10^{17} J$ or $1.4 \cdot 10^{11} kWh$, which is roughly 0.5% of the global energy usage, more than the country of Switzerland. Since a Bitcoin has a block production rate of 10 min and the average number of transactions per block is $1973\ \mathrm{txs/block}$ (ycharts.com), then each transaction requires $1350\ kWh$ or about a month's worth of electricity to the average American household. Note that the hashrate (and hence energy usage) will also increase when more miners join the network.

Increased Centralization due to ASICs

In PoW based blockchains such as Bitcoin, the miner who provides the first PoW solution produces the next block and is rewarded with a block reward. Therefore, miners are incentivized to increase the hashes per unit energy of their mining hardware, so the use of ASICs have become the standard. Consequently, the barrier to entry of mining has become increasingly difficult due to the exclusivity of obtaining/designing the hardware. Those who have the means to design custom hardware for mining rigs typically issue patents making it more difficult for actors with a small capital to be able to participate in Bitcoin mining. It can also be observed that over the years, the hashrate of large mining pools have dominated the total hashrate.

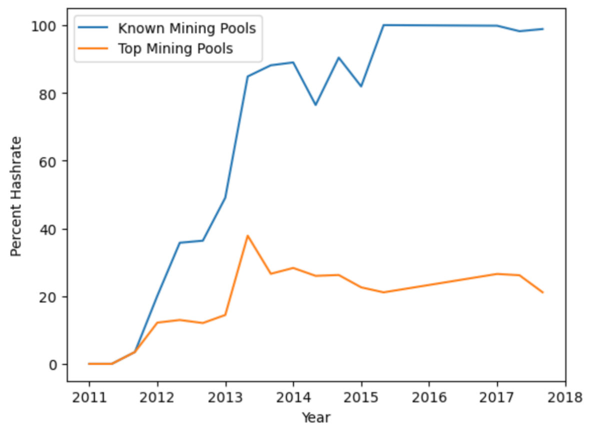


Figure 1: Mining Pool Statistics over Time (BTC.com)

Figure 1 shows the percent hashrate of the known mining pools and the hashrate of the largest mining pool over the years. It is clear that independent mining has become essentially obselete over time and large actors are controlling significant fractions of the total hashrate.

Mining has also been biased to where energy has been cheap. In January of 2021, the top 5 locations for mining (by hashrate) are China (53.3%), United States (10.41%), Russia (6.91%), Kazakhstan (6.17%), Malaysia (5.18%). Large scale operations are typically held in rural locations of these countries where cost of electricity is cheapest, often competing with nearby towns for energy resources.

Selfish Mining Attacks

A popular argument amongst PoW enthusiasts is that PoW protocols are more "secure" than alternatives. The backbone of this argument lies in the fact that it is practically impossible (assuming cryptography isn't broken) to guess a nonce without brute forcing every possible solution. In Satoshi Nakamoto's original whitepaper of Bitcoin, the trust assumption was if less then 50% of the hashrate is controlled by a single adversary, then no actor can take full control of the blockchain. This has shown to be false with Selfish Mining, where an adversary can mine blocks in secret and only announce the secret chain when it is most advantageous for the miner. An adversary can provably achieve this with $<$ 50% of the overall hashrate. This attack is possible in PoW since these proofs can be generated and withheld without the network knowing about them and hence the attacker has the advantage of having fast network speeds and no difficulty adjustment applied to their mining.

Why Alternatives to PoS?

PoS is the second most popular sybil-resistance mechanism by market capital (~$115B as of December 2021, not including Ethereum 2.0). Since the sybil-resistance mechanism in PoS does not involve computing a cryptographic puzzle, it is significantly more energy efficient. PoS is promising technology that has seen many of the top blockchain projects adopt it (Ethereum 2.0, Solana, Cardano); however, it isn't without its faults. We will present the most convincing arguments against PoS.

Closed System

The only way for a node in a PoS system to participate in mining is to stake some amount of that blockchain's native currency. This means that the protocol is a closed system as staking does not depend on any external factors. In a PoW protocol like Bitcoin, an actor with a large amount of capital can invest in creating a powerful mining rig and immediately have a large fraction of the total hashrate allotted to her. In a PoS setting, if an actor wishes to participate in staking, she must buy the coins from someone else within the network, making it much more difficult for her to have a large fraction of the stake. This has the arguable disadvantage that over the long term, more of stake will be concentrated to within the same set of actors.

Nothing at Stake

Because a miner in PoS does not require rigorous computational power that PoW protocols require to generate proofs, a miner can claim to extend many blocks without penalty, this is known as the "nothing at stake" problem. This can potentially reward dishonest miners and can slow down the consensus of the protocol.

Metrics to Compare Proof Mechanisms

Learning from the common criticisms of PoW and PoS that was discussed in the previous sections, we can identify properties that alternative sybil-resistance mechanisms can improve upon.

Energy Consumption

Like previously mentioned, the blockchain/cryptocurrency criticism that gets a lot of limelight (and rightfully so) is the enormous amount of energy consumption of PoW based solutions. Although PoS solves this issue, an alternative sybil-resistance protocol that preserves the energy efficient property of PoS is favorable.

Predictability

PoW proofs are generated by guessing nonces via brute force until a guess is found that when hashed, is less than the difficulty parameter. By the random oracle assumption, the probability of finding a solution given a particular guess is uniformly distributed; this also implies that it is hard to predict the next leader. The probability of finding a solution is also directly proportional to the amount of computation invested in trying to find the solution. Non-PoW based consensus doesn't always have this advantage of an easily verifiable and random leader selection. When analyzing alternate mechanisms, it is important to gauge to what extent the next leader is predictable.

Attacks

Any protocol is subject to attacks, but we want the outcomes of these attacks to have as little destruction as possible. It is also worth looking into the practicality of such attacks; e.g., Selfish Mining isn't seen much in practice.

Decentralization

It is clear from previous analysis that ASIC-based PoW suffers from economies of scale and has become more centralized over the years. When looking at new sybil-resistance mechanisms, we want to ensure that anyone can participate in mining with a low barrier of entry. Closed systems such as PoS can arguably lead to more centralization in the long term because staking requires the coins to be bought from within the closed environment of the blockchain, whereas a miner in a PoW network can participate in mining using wealth from the outside world. This fact can lead to a higher concentration of wealth within a small set of stakers in the long term; therefore, "open systems" such as hardware-based mining are worth looking in to.

Proof of Space (PoSpace)

Overview

Due to the large amount of computational power and energy consumption required for Proof of Work (PoW), many proposals appeared to find alternative approaches for PoW. One of the alternative approaches, Proof of Space (PoSpace), was first introduced by Dziembowski et al. [16] in 2015. The idea was motivated by memory or disk space. Due to a significant amount of free disk space available for many users, it would be profitable and reasonable to dedicate some of this free space to a blockchain. Furthermore, running a PoSpace node is essentially for free, compared to the demand for computation hardware like GPU.

Mechanisms

Proof of Space is an interactive protocol between a prover P and a verifier V, and there are two phases. According to [16], during the initialization phase, P stored non-trivial information data $\mathcal{F}_\gamma$ of size $N \in \mathbb{N}$, and V stores a short commitment γ to the data. After some point when V initializes the execution phrase, P will provide a proof with an amount of memory to the protocol to V, and V will run a PoS to verify that the proof was well generated according to $\mathcal{F}$ stored in P and thus, only outputs either accept or reject based on the verification result.

As for definition, suppose there are two parties P and V with shared input IN, local inputs IN_P, IN_v, and local outputs OUT_p and OUT_v, the execution of PoSpace takes place as follows:

$$\left(OUT_v, OUT_p\right) \leftarrow < V(IN_v), P\left(IN_p\right) > (IN)$$

For initialization phase, IN includes parameters, such as an identifier id, a storage bound $N \in \mathbb{N}$, etc. The output OUT_v would be ϕ, which represents the small piece of information stored in V, and OUT_p would be S with size N, which represents the information stored in P. Thus, the formula becomes

$$(\phi, S) \leftarrow < V, P > (id, N, \ldots)$$

For execution phase, both P and V take output from initialization phase as input. After this phase, P has no output, while V chooses either accepts or rejects.

$$(\{accept, reject\}, NIL) \leftarrow < V(\phi), P(S) > (id, N, \ldots)$$

More specifically, the prover P needs to generate a large file $\mathcal{F}$ locally during an initialization process with takes some time I, which must be at least linear in the size of file. This is used for having small computation, storage requirement, and communication complexity of the verifier V during both phases. After filling the amount of memory that V requires, P sends a commitment γ of the memory it has reserved to V and then generates proofs during the execution phase. V can set a challenge to P, then P will retrieve a memory statement S and send it to V. Finally, V can accept or reject based on the memory statement S from P and commitment γ.

Based on the construction, a PoSpace in the random oracle model should satisfy the following properties:

- *Efficiency*: Let λ be a security parameter. The verifier runs in $O(1)$ time during initialization phase and $O(\lambda \cdot \log(N))$ time during the execution phase. The honest prover runs in $O(N \cdot loglog(N))$ time during initialization phase and $O(\lambda \cdot loglog(N))$ during execution phase.
- *Security*: any prover that can convince verifier must either dedicate N bits of space if the prover is honest, or simply run the execution phase in $\theta(N)$ time on average if the prover is dishonest.

Techniques

As for the construction of proof, the first approach for Proof of Space is to use a standard technique called graph pebbling for proving lower bounds on space complexity of computational problems. During initialization process, it generates a directed acyclic graph $G = (V, E)$ with $|V| = N$ vertices. It labels l_i for each vertex $i \in V$, which can be computed as:

$$l_i \coloneqq \mathcal{H}(\mu, i, l_{p_i}, \dots, l_{p_t})$$

where $\mathcal{H}$ is a hash function, and $p_1, \dots, p_t$ are parent vertices/predecessors of vertex i for $\forall i \in V$. Next, P computes and stores labels. It then creates a Merkle tree with these values and share the root with verifier V. In the execution process, V simply asks P for a subset of labels in the graph, and P will show the corresponding nodes of the Merkle tree. With the root and values, V will be able to verify P.

In 2017, Abusalah et al. [12] proposed a new approach to Proof of Space based on storing tables of random functions. The prover P is given the description of random functions $g_f: N \times N \to N$ and $f: N \to N$, where f is chosen by the verifier or a random public challenge. During the initialization phase, it finds all pairs $x, y \in N$ such that $x \neq y$ and $f(x) = f(y)$, then it stores $(g(x, y), (x, y))$. During the execution phase, verifier V sets up a challenge called c to P, and V will accept if the proof can find a pair (x, y) so that $g(x, y) = c$.

Implementation

SpaceMint

SpaceMint was first proposed by Park, Sunoo, et al. [19] in 2018. It is a cryptocurrency based on PoSpace so that miners in SpaceMint can dedicate disk space rather than computational power, in hope of alleviating large energy consumption from Bitcoin and finding the alternative effective approach to Proof of Work.

Compared to any PoW-based blockchains like Bitcoin, generating blocks using PoSpace is computationally cheaper and easier. Therefore, SpaceMint needs to consider about how to determine the miner who first found the next block. It defines a quality function by assigning a quality value $Quality(\pi_i)$ to PoSpace and making sure that the probability of any miner being the first to find an eligible next block is proportional to the space it dedicates.

Specifically, suppose there is a set of m valid proofs π_i, where $1 \leq i \leq m$. $Quality(\pi_i)$ represents the probability that has the best quality among m proofs. So $Quality(\pi_i)$ can be computed as:

$$Quality(\pi_i) \coloneqq D_{N_\gamma}(\mathcal{H}(a))$$

D_N is a maximum output of the distribution that samples of size N values in $[0,1]$ at random, and $\mathcal{H}(a)$ denotes sampling randomness for D_N.

Tradeoffs

Energy Efficiency

Based on the mechanism of Proof of Space, before mining, users dedicate some reserved non-trivial amount of memory to the blockchain with the filling data. When the miner tries to mine a block on the blockchain, the reserved space is used for generating solution to the challenges. Since the solution for puzzle can be regarded as memory accesses, PoSpace is more energy and time efficient.

Take SpaceMint as an example. According to the data from [19], a miner takes $210kJ/min$ to add one block on SpaceMint using PoSpace, which means the annual energy consumption would be approximately $1.1 \cdot 10^{11} J$. Compared to the number we got about the annual energy consumption of Bitcoin (i.e., $5 \cdot 10^{17} J$) from blockchain.com in December 2021, the ratio of energy consumption between SpaceMint and Bitcoin is approximately $1: (5 \cdot 10^6)$, which is significantly small.

Nothing-at-stake Issue

The fast proof generation for Proof of Space may create the nothing-at-stake issue. It happens because the proof is computationally easy and cheap to create, and a malicious miner can generate lots of alternative blocks at close to no cost until the miner finds a relatively high success of attack or some advantages over the process that follows the protocol. Therefore, PoSpace needs to be combined with other mechanism that can counter the issue, such as combining time as Proof of Space and Time (discussed in the later section).

Digging Attack

According to Martinho [18], a digging attack can be described as a miner trying to extend a block, such as trying different transaction combinations to gain a better proof, in order to obtain an unfair advantage over honest behaviors.

Long-range Attack

A long-range attack can be described as a malicious miner creating an alternative chain of transactions privately starting from the same genesis block, until the number of blocks in the private chain exceeds the chain created by honest miners. Since proofs are computationally easy to generate, the malicious miner can create blocks at close to no costs. Since PoSpace alone is vulnerable to this attack, in some applications like Chia (which will describe in a later section), it makes use of both Proof of Space and Proof of Time.

Proof of Space and Time (PoS&T)

Overview

Proof of Space and Time uses elements from both PoW and PoSpace to provide a low-computation option for sybil resistance. In essence, this combines Proof of Space with the new notion of a "Proof of Time," which requires time to pass between each block on any chain. This is accomplished by the use of a

Verifiable Delay Function (VDF). This function requires sequential (non-parallelizable) computation to solve, but is easy to verify.

Using a Proof of Time based on a VDF, we prevent adversaries from performing a long-range attack against the blockchain. Because a VDF necessarily requires time to pass in order to make the sequential calculations required, an adversary cannot speed up the process of adding to a new chain forked far down the longest chain to be significantly faster than farming on the longest chain. Therefore, this bootstrapped chain will be unable to catch up to the longest chain.

Mechanisms

Proof of Space and Time shares a mechanism similar to Proof of Space, where the addition of each new block shares the same setup exactly as Proof of Space. The crucial difference is that after a block B has been accepted by a Proof of Space, A Proof of Time must be submitted for before this block is “completed," and additional blocks can be added with Bas their predecessor. Note, however, that this proof of time may be submitted by any participant in the network. This may be accomplished (as in Chia) through the use of a dedicated server that continuously works on a Proof of Time for the most recent block on the longest chain. This Proof of Time is implemented in the form of a Verifiable Delay Function.

Verifiable Delay Function (VDF)

First, we will review the definition of a verifiable delay function, along with some potentially useful properties a VDF may exhibit. Although named a “function," a VDF in reality consists of three algorithms: one $Setup$, $Eval$, and $Verify$. These algorithms have the following inputs, outputs, and purposes, as defined in [17]:

$Setup$: $Setup$ takes as its input two parameters: a security parameter λ and a delay parameter t. It then outputs a setting of public parameters $pp = (ek, vk)$, used to fix the domain and range of the VDF challenge. Specifically, ek and vk are the evaluation and verification keys, respectively. Note that $Setup$ must run in time polynomial in terms of λ and t must be sub-exponential in terms of λ.

$Eval$: $Eval$ takes as its input two parameters: the evaluation key ek output from $Setup$ and an input x from the domain specified by ek. It then outputs a new (deterministically calculated) value y in the range specified by ek and optionally a proof π that y is the correct output for x with evaluation key ek. Note that $Eval$ must run in time t for all valid inputs with any number of parallel processors polynomial in terms of $log(t)$ and λ.

$Verify$: $Verify$ takes as its input four parameters: the verification key vk output from $Setup$, an x from the domain specified by vk, a y from the range specified by vk, and a proof π that y is the correct output corresponding to input x under verification key vk. It outputs a Boolean value, whether the proof proves y is the output corresponding to x under verification key vk. Note that $Verify$ must run in time polynomial in terms of $log(t)$ and λ.

In order for these functions to together be a VDF, they must also satisfy the following properties:

Correctness: A VDF is *correct* if $\forall \lambda, t, (ek, vk)$, input $x \in X$ specified by (ek, vk), if (y, π) is returned from $Eval(ek, x)$, then $Verify(vk, x, y, \pi) = True$.

Soundness: A VDF if *sound* if for all algorithms A that run in $O(poly(t,\lambda))$ time, $Pr[Verify(vk,x,y,\pi) = True, y \neq Eval(ek,x) \mid pp = (ek,vk)$ is returned from $Setup(\lambda,t)$, and (x,y,π) is returned from $A(\lambda,pp,t)] \leq negl(\lambda)$, where $negl(\lambda)$ is a negligible function of λ.

Sequentiality: For functions $\sigma(t)$ and $p(t)$, a VDF if $(p,\sigma) - sequential$ if no pair of randomized algorithms A_0, A_1, where A_0 runs in $O(poly(t,\lambda))$ and A_1 runs in parallel time $\sigma(t)$ on at most $p(t)$ processors, can win the *sequentiality game* with probability greater than $negl(\lambda)$.

Note that the *sequentiality game* for an adversary (A_0, A_1) follows the below steps:

1. pp returned by $Setup(\lambda,t)$
2. L returned by $A_0(\lambda,pp,t)$
3. x returned by $rand\_choice(X)$
4. y_A returned by $A_1(L,pp,x)$

The Adversary wins the game if and only if $y_A = y$ where $(y,\pi) = Eval(pp,x)$.

Implementation

Chia Network

The Chia Network is based on a longest-chain blockchain that uses Proof of Space and Time for its sybil resistance [15]. Specifically, For Chia's Proof of Time, they use a VDF server, known as a "Timelord," to continuously evaluate the VDF for the honest end of the longest chain. The entire network can function with only a single running Timelord, as only the fastest timelord will be used to mark the completion of a block, although in practice additional Timelords do help with redundancy and security in the case that other Timelords stop functioning or stop behaving honestly.

The block difficulty takes a form similar to that in Proof of Work, choosing a difficulty so that 32 blocks are completed on average every 10 minutes. This difficulty is calculated based on the total amount of space in the network and the speed of the fastest Timelord.

Tradeoffs

Energy Efficiency

The energy usage from Proof of Space and Time comes from three main sources: Storage manufacturing, storage operation, and VDF calculation. Note that the first two are nearly the same as Proof of Space, as the mechanism for storing data is the same. However, there is time between blocks honest due to the VDF. This means that there will be fewer disk reads/writes per second, leading to slightly less energy usage. The VDF calculation itself may seem as though it would be significantly energy intensive, as it requires near-continuous running of CPU cores. However, the sequentiality property of the VDF guarantees that the computation cannot be negligibly parallelized. Further, because only a single proof of time must be submitted for a given block, which can be effectively performed off just a single VDF server, the amount of energy does not necessarily scale as the coin price increases or the total space grows, so any computation from VDF calculation is asymptotically negligible.

Validator Predictability

Thanks to the use of a VDF for block completion, a block must have its VDF be solved before being able to use its resulting hash to predict the next validator. Because this VDF requires significant time to evaluate, an adversary cannot predict future validators significantly faster than the proof of time would be submitted.

Long-range Attacks

In a long-range attack, an adversary A tries to generate a chain forked from the honest longest chain many blocks ago. If A has enough time and enough space to be successful, this could potentially invalidate a large number of blocks on the previously longest chain. However, because it takes a significant amount of time to complete adding any block to the blockchain, an adversary would not be able to add more blocks to a long-range chain at a faster rate than blocks are added to the longest chain, meaning the attacking chain will not catch up to the longest chain (with very high probability), rendering long-range attacks ineffective.

Decentralization

For the same reasons as Proof of Space, Proof of Space and Time does not suffer from centralization in the form of large cryptocurrency exchanges having a disproportionate amount of the stake. However, there is still the potential for large farming pools to form, allowing a significant amount of the total network space to be owned under a single pool.

Proof of Spacetime

Overview

Proof of Spacetime aims to achieve sybil resistance by using the scarce resource of not just instantaneous storage, but storage over time. By requiring data to be stored over time, participants can make up for lower amounts of total storage by increasing the time data is held, allowing for less energy to be expended in the act of storing the data.

Mechanism

A Proof of Spacetime as defined in [14] is an interactive protocol between a prover $P = (P_{init}, P_{exec})$ and a verifier $V = (V_{init}, V_{exec})$. The protocol takes place in two phases. Note each initialization phase may be followed by multiple repetitions of the execution phase.

Initialization: Both P and V receive an input id bit string id. At the end of initialization, P and V output state bit strings σ_P and σ_V respectively.

Execution: Both P and V receive the id from the initialization phase and their corresponding state. At the end of execution, V outputs a Boolean value out_V representing whether V has accepted the input and state.

Any such protocol of the above form must have the following properties of $\eta - completeness$ and $f - soundness$ in order to be a $(w, m, \epsilon, f) - PoSt$:

$Completeness$: A PoSt of the above form is $\eta - complete$ if for every $id \in \{0,1\}^{poly(k)}$ and every random oracle H^{work},

$$Pr\left[out_V = 1 \middle| (\sigma_P, \sigma_V) \leftarrow < P_{init(id)}, V_{init(id)} >, (\bullet, out_V) \leftarrow < P_{exec}(id, \sigma_P), V_{exec}(id, \sigma_V) > \right] \geq \eta$$

$f - soundness$: A PoSt of the above form is $(\epsilon, f) - sound$ if for all $T_0, T_1, s \geq 0$ and $n \geq 1$ for every adversary (A_0, A_1) if it satisfies the following conditions in the $(n, s, T_0, T_1) - PoST\ Security\ Game$:

- *Rational Storage*: If A_0 made fewer than $\epsilon \cdot w$ queries to the random oracle, then the probability of success negligible in the security parameter (at most $negl(\lambda)$)
- *Space-Time Trade-Off*: $Pr[(A_0, A_1)$ wins $(n, s, T_0, T_1) - PoST\ Security\ Game] \leq f(n, s, T_0, T_1)$

Note that the $(n, s, T_0, T_1) - PoST\ Security\ Game$ is defined as 2 phases:

Initialization: A_0 chooses a set of bit-string ids $\{id_1, \dots, id_n\}$. A_0 then interacts with n independent, honest verifiers executing the initialization phase of the PoSt protocol, sending each id to the corresponding verifier. σ_A is the resulting output of A_0, and $(\sigma_{V_1}, \dots, \sigma_{V_n})$ are the outputs of the verifiers.

Execution: A_1 interacts with n independent, honest verifiers executing the execution phase of the PoSt protocol, where each verifier i gets (id_i, σ_{V_i}) as input.

The adversary (A_0, A_1) *succeeds* iff $|\sigma_A| \leq s$, A_0 makes at most T_0 queries to the random oracle, A_1 makes at most T_1 queries to the random oracle, and all n verifiers output 1.

Implementations

Spacemesh

Spacemesh is a consensus protocol that uses a “mesh” rather than a “chain” with PoSt for sybil resistance [13]. The “block-mesh” architecture allows for the creation of multiple blocks in parallel rather than a sequence of single blocks as in a typical blockchain. Because of this, individual miners are able to contribute blocks without having to participate in a pool.

The proof of spacetime takes place in two phases. First, a miner “commits” to the data to be stored, occupying S bits. After this, the miner continues to prove that this data is still being stored by supplying additional proofs over the time T that the data is stored.

Tradeoffs

Energy Efficiency

The energy usage from Proof of Spacetime comes primarily from the initial computation in the initialization phase and the energy cost of storage over time. As participants in the network can simply increase the time they store a given amount of data, the energy from the initialization phase becomes negligible as the time stored increases. In the analysis of proof of replication below, we give estimates of energy usage that are comparable to those of spacetime.

Decentralization

Centralization varies implementation to implementation. For reasons similar to Proof of Space and Proof of Space and Time, centralization of storage may be moved to pools, but with alternative implementations like Spacemesh, decentralization through both easy-to-access hardware and easily-mined blocks can be made possible.

Proof of Replication

Overview

Proof of Storage can be considered a generalization of *Provable Data Possession* (PDP) and *Proof-of-Retrievability* (PoRet), which were both independently introduced prior to 2008, and discussions of these schemes fall out of scope for this paper but can be found in [1,2]. Proof of Replication (PoRep) extends these to include additional properties which ensure that the data is given its own unique physical location (i.e., that it has been replicated by the server/prover). Proof of Replication was proposed as a novel Proof of Storage implementation in 2017 as a way to combat the attack vectors which prevent PDP or PoRet from being used as a sybil resistance mechanism for Nakamoto consensus. This scheme was first proposed in [3] at Protocol labs, and as we will see below, forms the thesis of some blockchain projects.

We first discuss the issues/attack methods which PoRet resolves, as laid out explicitly by the creators:

> "*Sybil Attacks*: Malicious miners could pretend to store (and get paid for) more copies than the ones physically stored by creating multiple Sybil identities, but storing the data only once
>
> *Outsourcing Attacks*: Malicious miners could commit to store more data than the amount they can physically store, relying on quickly fetching data from other storage providers.
>
> *Generation Attacks*: Malicious miners could claim to be storing a large amount of data which they are instead efficiently generating on-demand using a small program. […]" [3]

Note that while Sybil attacks must be dealt with in any permissionless consensus method, the outsourcing and generation attacks specifically arise when storage is used as the arbiter of stake in a system. In the generation attack, collusion between a client and server can maliciously inflate the server's relative amount of storage in use in the total environment, unless the PoRep is publicly verifiable as well. I.e., if the goal is to incentivize the network to store more user data, then the public verification of is critical to the security of the network.

These additional three properties being fulfilled on top of the earlier PDP and PoRets allows for the use case in which PoRep proofs to be submitted as a type of "proof of resource" allowing a network to substitute PoW for these proofs, as we shall see.

Mechanisms

The original PoRep paper proposes an adversarial game RepGame that that a PoRep scheme "must pass to be secure" and which formalizes the definition of PoRep schemes—the game essentially presents three challenges, which we discussed above: *Sybil Attack, Outsourcing Attack, and Generation Attack* [3]. While we will not repeat the entire details here, we will present a formal definition as it will be important context for explaining how these are implemented in today's cryptocurrencies.

> "Definition: A general $\prod^{PoRep}$ proving scheme $\prod^{PoRep} = (Setup, Prove, Verify)$ is a set of algorithms that together enable a prover $\mathcal{P}$ to convince a verifier $\mathcal{V}$ that $\mathcal{P}$ is storing a replica $\mathcal{R}^{\mathcal{D}}$ of data $\mathcal{D}$. No two replicas $\mathcal{R}_i^{\mathcal{D}}, \mathcal{R}_j^{\mathcal{D}}$ can be deduplicated into the same physical storage; they must be stored independently. The three algorithms are:

- $\mathcal{R}^{\mathcal{D}}, \mathcal{S}_{\mathcal{P}}, \mathcal{S}_{\mathcal{V}} \leftarrow PoRep.Setup(1^{\lambda}, \mathcal{D})$ where $\mathcal{S}_{\mathcal{P}}$ and $\mathcal{S}_{\mathcal{V}}$ are scheme-specific setup variables for $\mathcal{P}$ and $\mathcal{V}$ respectively, that depend on the data $\mathcal{D}$, and on a security parameter λ. $PoRep.Setup$ is used to initialize the proving scheme and give $\mathcal{P}$ and $\mathcal{V}$ information they will use to run $PoRep.Prove$ and $PoRep.Verify$. Some schemes may require either party to compute $PoRep.Setup$, require it to be a secure multi-party computation, or allow any party to run it.

- $\pi^{c} \leftarrow PoRep.Prove(\mathcal{S}_{\mathcal{P}}, \mathcal{R}^{\mathcal{D}}, c)$, where c is a challenge, and π^{c} is a proof that a prover has access to $\mathcal{R}^{\mathcal{D}}$ a specific replica of $\mathcal{D}$. $PoRep.Prove$ is run by $\mathcal{P}$ to produce a π^{c} for $\mathcal{V}$

- $\{0,1\} \leftarrow PoRep.Verify(\mathcal{S}_{\mathcal{V}}, c, \pi^{c})$, which checks whether a proof is correct. $PoRep.Verify$ is run by $\mathcal{V}$ and convinces $\mathcal{V}$ whether $\mathcal{P}$ has access to $\mathcal{R}^{\mathcal{D}}$" [3]

While the authors leave implementation details open to various schemes, they make two design decisions which are currently employed in at least one major blockchain, Filecoin [4]. Firstly, to prevent the Sybil Attack, the provers use an encoding key for each replica, such that encodings of replicas of the same data are distinguishable. I.e., for n replicas, there are n encoding keys. Formally, $\mathcal{R}^{\mathcal{D}}_{ek_i} \neq \mathcal{R}^{\mathcal{D}}_{ek_j}$ when $ek_i \neq ek_j$. Secondly, to prevent the *Outsourcing Attack* and *Generation Attack,* the scheme uses a time bounding—essentially, force the encrypting/sealing to take time that cannot be sped up with additional computing power, so that the prover could not generate a response on demand. Only an honest prover with the sealed data already stored would be able to respond within the time window. The authors formalize this such that in [3]:

$$\mathcal{T}^{honest} = RTT^{\mathcal{V}\to\mathcal{P}\to\mathcal{V}} + Time\left(PoRep.Prove\left(\mathcal{S}, \mathcal{R}^{\mathcal{D}}_{ek}, c\right)\right) \ll$$
$$\mathcal{T}^{attack} = RTT^{\mathcal{V}\to\mathcal{P}\to\mathcal{V}} + Time\big(PoRep.Prove(\mathcal{S}, Encode(\mathcal{D}, ek), c)\big)$$

Thus, the choice for encoding becomes a critical decision. The original whitepaper from 2017 left this as an open question, and we will discuss below the implementation choices which Filecoin has made in the intervening years, and which type of encoding they now run. Broadly, the solution suggested is for the encode to use some sort of pseudo-random permutation (PRP) and slow down the encode arbitrarily by forcing the person encrypting to sequentially chain parts of the encoding some τ iterations. Decisions about this method impact many things, including how the size of the data $\mathcal{D}$ and the time to decode would scale, allowing the encoding time to be tuned by some parameter so that if RTTs change or if data needs to be retrieved faster or slower, the network could adjust this, as well as making sure it is not compressible. Finally, within the domain of Decentralized Storage Networks, we would like this to be publicly verifiable, so the encode would be within a SNARK/STARK scheme, and not prohibitive to compute.

Tradeoffs

Within Distributed Storage Networks (DSNs), the term *durability* refers to the probability that data remains available in the face of failures. There are multiple ways to try to ensure that data is not lost. In the scheme described above, the data $\mathcal{D}$ becomes unavailable if all provers with a replica $\mathcal{R}^{\mathcal{D}}$ go offline or leave the network. That scheme is at the heart of the Filecoin protocol, discussed more below. Durability in this way can be thought of as the probability that all holders of the data simultaneously leave the network before it is able to be *repaired*. In this context, if a user specifies they would like $n = 10$ copies of their data made, then when one prover stops responding (and we assume they have left the network), a new replica of the data is made on a new prover—in this way the network can be said to be self-healing.

Durability is used as one metric in which DSNs are measured against. However, durability is also linked to network bandwidth usage and network expansion factor, which is again defined for DSNs as the storage overhead for storing some amount of data with a durability above some accepted tolerance. Perhaps the single largest tradeoff for using PoRep is that expansion factor scales linearly with the number of replicas required. If the user requires a high degree of probability that the data will not be lost—say, 10 copies, this corresponds to a 1000% expansion factor. The argument against replication here is that there are much more efficient ways of distributing data such that the expansion factor and consequent overhead can be minimized without sacrificing durability.

Erasure codes offer one alternative. This is "an encoding scheme for manipulating data durability without tying it to bandwidth usage, and have been found to improve repair traffic significantly over replication. Importantly, they allow changes in durability without changes in the expansion factor." [4] The details of erasure code history within distributed and peer-to-peer storage systems falls outside the scope of this report, but they have been around since and as a product of Reed-Solomon [5]. Erasure codes depend on two variables (n, k) where n is the total number of erasure shares created, and any k are required to recreate $\mathcal{D}$. The authors of the Storj DSN whitepaper model durability for erasure codes as a CDF of the Poisson distribution, where $p = 10\%$ churn rate/prover loss every month, and $\lambda = p \cdot n$ the amount of erasure shares lost in a month for some data $\mathcal{D}$. Then

$$P(\mathcal{D}) = e^{-\lambda} \sum_{i=0}^{n-k} \frac{\lambda^i}{i!}$$

From this we can see that in order to provide the same level of durability, erasure codes lead to much lower expansion rates in comparison with full replication, and direct the reader to [4] for further assumptions and arguments about durability/expansion factor interplay. They argue that lower expansion factor leads to storage nodes being paid more, with high expansion factors "dilut[ing] the incoming funds per byte across more storage nodes…a much more direct passthrough of income to storage node operators." PoRep cannot use such erasure codes since "the PoRep mixes all the input data in a way that loses the erasure coding property." [A]

Implementations

Filecoin

The Filecoin whitepaper in 2017 proposes itself as a DSN which "works as an incentive layer on top of IPFS" where the native protocol token is earned by providing storage to clients [6]. More specifically, their specifications are for the "probability that the network elects a miner to create a new block…is proportional to their storage currently in use in relation to the rest of the network." Filecoin uses a modified PoW where miners submit PoSts onto the blockchain—each of these proofs of spacetime are sequential proofs of replications, as modeled above. The security of the consensus and the network thus depend on the security of the PoSt, in which they “cannot lie about the amount of assigned storage they have…since this would require time fetching running the slow PoSt.Setup and they cannot generate proofs faster by parallelizing the computation, since PoSt.Prove is a sequential computation.” Their modified Proof of Storage follows similar probabilistic leader election as in Algorand and Snow White.

Filecoin, as an L1 blockchain with its own native currency, has also added functionality via smart contracts based on Ethereum’s model for users to program and modify the conditions on which they want to their data to be serve, availability, and other tunable parameters. Their whitepaper also mentions ongoing effort to create cross-chain bridges so that other blockchains like Ethereum and Tezos could take advantage of the Filecoin storage system, as opposed to keeping data directly on Ethereum or Tezos, where storage capacity is more expensive. As of today, it appears that such a bridge was recently completed via Polygon network [7].

Storj

It is interesting to look at the use case of DSNs and the different approaches that are possible, which highlight the different design choices with respect to performance, decentralization, and transparency. Storj is an application on Ethereum which offers the same service as Filecoin, but which does not have its own blockchain. Storj acts as a trusted third party and performs the role of validator, checking that provers (not technically miners in this scenario) maintain storage of the data they have committed to, and rewarding them with their ERC-20 token. Their whitepaper mentions these checks are Proof of Storage-based, but not as rigorous as the PoSt and PoRep in Filecoin, and Storj additionally chooses to use erasure codes instead of full replication [8]. Here there is a clear tradeoff between transparency and efficiency, as Storj claims to passing these savings from erasure codes and bandwidth reduction for data repair onto the storage operators. Filecoin puts proof information and contracts between users and providers on the public blockchain. Finally, Storj uses a Market Maker model, where the user directly pays the Storj application (again lack of transparency in pricing) – whereas the more decentralized Filecoin uses a matching model in which users and servers post their availability, and are freely paired in the Filecoin market [6].

Comparisons

Energy Efficiency

Straightforward comparisons of energy usage and efficiency between PoRep and other alternatives are inherently flawed by individual blockchain and consensus mechanism’s differing goals. A DSN has different design goals than a strictly transactional blockchain. PoRep, when applied to a Nakamoto style longest chain consensus mechanism, can be viewed as a type of Proof of Useful Work – under the assumption that storing files is a more useful byproduct of incentivization by the blockchain than solutions to hashing inequality, as in Bitcoin.

Additionally, comparisons are applicable to implementations, rather than to proof schemes.

However, Filecoin has released their internal methodology for their network’s energy consumption, as well as data for the entire network, which allows us a sense of how much energy are they consuming and where in the process that energy is being used [9].

They model the framework for on-chain energy consumption with the following equation:

$$P = (A \cdot SR + B * Cap) \cdot PUE$$

Where P=Power in Watts, A is sealing/encoding energy constant in $Wh/byte$ [Note: this A constant is the variable discussed above which takes non-parallelizable time so that it forces the provers not to generate the data on demand or fetch from elsewhere, see Mechanism section for more], SR is sealing rate in bytes/hour, B is the electrical power required to store data in $W/byte$, Cap is capacity in bytes of data stored, and PUE is the Power Usage Effectiveness, which they define as "ratio of total electrical power consumed to that consumed by IT processes," which they pulled from Database Center literature to get estimates.

Since the sealing rate is the part unique to PoRep proofs, we focus on that. Filecoin estimates a value of $3.42 \cdot 10^{-8} Wh/byte$, and a storage energy of $3 \cdot 10^{-12}\ W/byte$ [10].

The below figure shows the total network energy usage estimate:

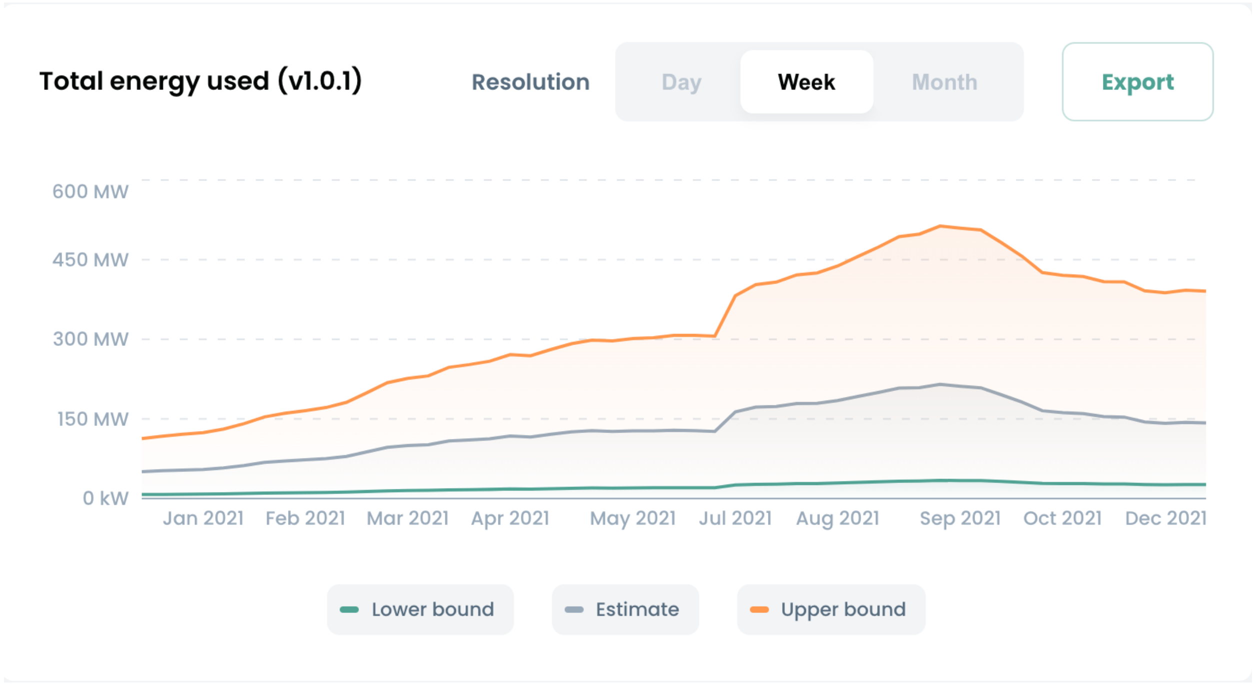


Figure 2: Filecoin Total Energy Used

This data shows they are consuming $\cong 150\ MW$ of energy, about $\frac{1}{3}\ or\ 50MW$ of which is dedicated to power the hard storage drives, and about 45 MW of which is used to seal/encode data. This data is used to store 14.9 EiB of data, as of today. While not only providing useful byproducts of the blockchain, a recent analysis found that the "total energy consumption of the Filecoin network is comparable that of similarly sized datacenters," which is a better comparison dimension and shows the relative efficiency of the distributed system [11].

Conclusion

While Proof of Work and Proof of Stake remain the most common forms of sybil resistance, alternative methods do have purpose and potential tradeoffs that may make them more or less desirable than PoW or PoS for different applications. In all the methods detailed above, storage in some form is used as a scarce resource. While other alternative sybil resistance methods do exist, these storage-based alternatives seem to offer some of the best properties in terms of energy efficiency, positive externalities, and decentralization.

References:

[1] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song. Provable data possession at untrusted stores. In Proceedings of the 14th ACM Conference on Computer and Communications Security, CCS ’07, pages 598–609, New York, NY, USA, 2007. ACM

[2] A. Juels and B. S. Kaliski, Jr. Pors: Proofs of retrievability for large files. In Proceedings of the 14th ACM Conference on Computer and Communications Security, CCS ’07, pages 584–597, New York, NY, USA, 2007. ACM

[3] Protocol Labs. Technical Report: Proof-of-Replication. https://filecoin.io/proof-of-replication.pdf

[4] Storj Labs. Storj: A decentralized Cloud Storage Network Framework https://www.storj.io/storjv3.pdf

[5] Jeff Wendling and JT Olds. Introduction to Reed-Solomon.
https://innovation.vivint.com/introduction-to-reed-solomon-bc264d0794f8

[6] Protocol Labs. Filecoin: A Decentralized Storage Network. https://filecoin.io/filecoin.pdf

[7] Yahoo. Filecoin and Polygon Deploy Interoperable Bridge to Expedite Web3 Development
https://www.yahoo.com/now/filecoin-polygon-deploy-interoperable-bridge-162000622.html

[8] Storj Labs. Storj: A Decentralized Cloud Storage Network Framework.
https://www.storj.io/storjv3.pdf

[9] Alan Ransil, Protocol Labs. Estimating Filecoin Electricity Consumption from On-Chain Proofs.
https://github.com/redransil/filecoin-energy-estimation/blob/main/methodology/filecoin-electricity-methodology-paper.pdf

[10] Filecoin Energy Use Estimate Methodology. https://filecoin.energy/methodology

[11] Introducing the Filecoin Energy Dashboard. http://www.coin.one/article/25624

[12] H. Abusalah, J. Alwen, B. Cohen, D. Khilko, K. Pietrzak, and L. Reyzin, “Beyond hellman’s time-memory trade-offs with applications to proofs of space,” Cryptology ePrint Archive, Report 2017/893, 2017, https://eprint.iacr.org/2017/893.

[13] I. Bentov, J. Loss, T. Moran, B. Shani. The Soacemesh Protocol: Tortoise and Hare Consensus… In.. Space https://whitepaper.io/coin/chia-network

[14] T. Moran, I. Orlav. Simple Proofs of Space-Time and Rational Proofs of Storage
https://eprint.iacr.org/2016/035.pdf

[15] B. Cohen, K. Pietrzak. The Chia Network Blockchain https://whitepaper.io/document/759/chia-network-whitepaper

[16] S. Dziembowski, S. Faust, V. Kolmogorov, and K. Pietrzak, “Proofs of space,” in Annual Cryptology Conference. Springer, 2015, pp. 585–605.

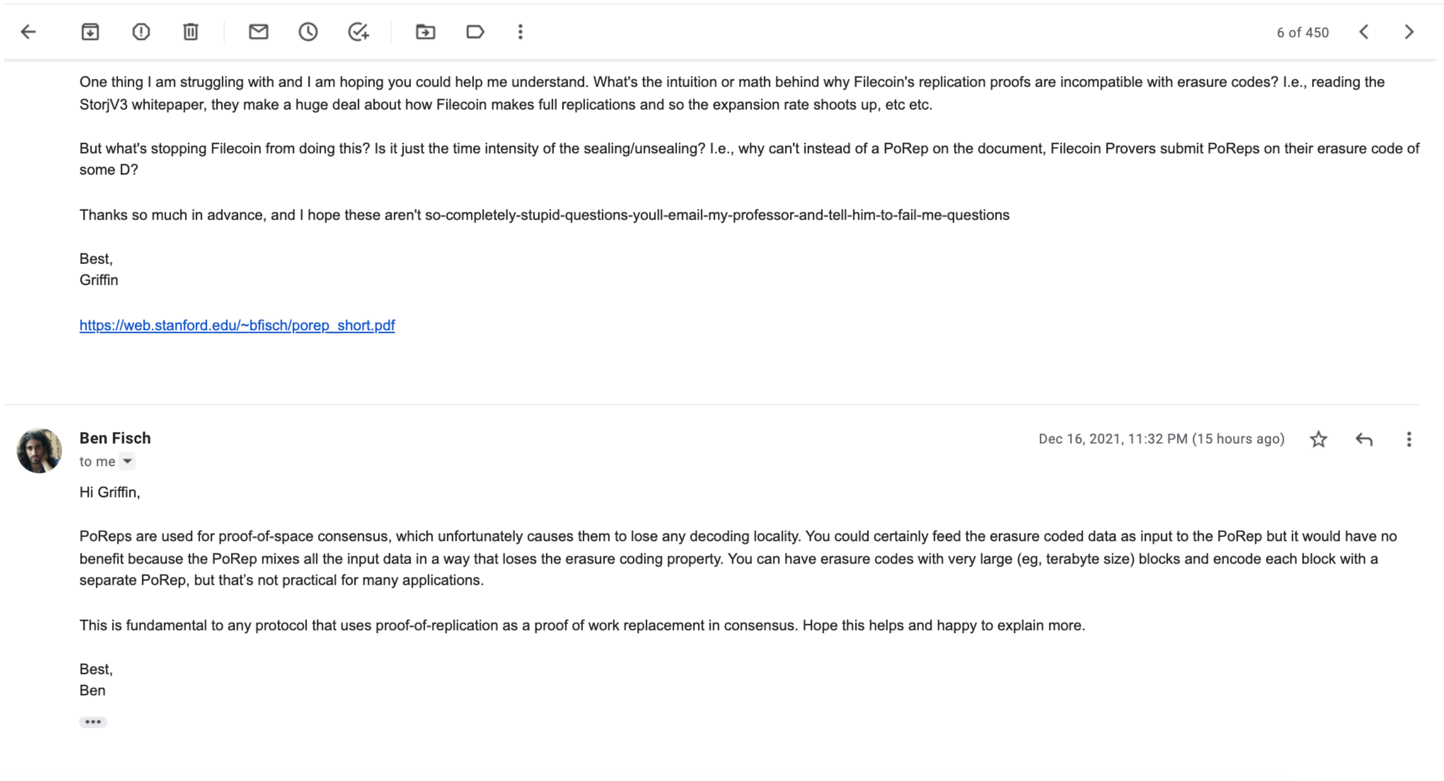
[17] D. Boneh, J. Bonneau, B. Bünz, B. Fisch. Verifiable Delay Functions. https://eprint.iacr.org/2018/601.pdf

[18] Martinho, Joao Vasco Estrela. "Efficient Proof-of-Space approaches for permissionless blockchains."

[19] Park, Sunoo, et al. "Spacemint: A cryptocurrency based on proofs of space." International Conference on Financial Cryptography and Data Security. Springer, Berlin, Heidelberg, 2018.

Appendix:

[A]. Email Correspondence with Ben Fisch, Stanford, author of multiple papers on PoRep and VDFs [https://web.stanford.edu/~bfisch/porep_short.pdf, https://eprint.iacr.org/2018/601.pdf]



6 of 450

One thing I am struggling with and I am hoping you could help me understand. What's the intuition or math behind why Filecoin's replication proofs are incompatible with erasure codes? I.e., reading the StorjV3 whitepaper, they make a huge deal about how Filecoin makes full replications and so the expansion rate shoots up, etc etc.

But what's stopping Filecoin from doing this? Is it just the time intensity of the sealing/unsealing? I.e., why can't instead of a PoRep on the document, Filecoin Provers submit PoReps on their erasure code of some D?

Thanks so much in advance, and I hope these aren't so-completely-stupid-questions-youll-email-my-professor-and-tell-him-to-fail-me-questions

Best,
Griffin

https://web.stanford.edu/~bfisch/porep_short.pdf

Ben Fisch
to me

Dec 16, 2021, 11:32 PM (15 hours ago)

Hi Griffin,

PoReps are used for proof-of-space consensus, which unfortunately causes them to lose any decoding locality. You could certainly feed the erasure coded data as input to the PoRep but it would have no benefit because the PoRep mixes all the input data in a way that loses the erasure coding property. You can have erasure codes with very large (eg, terabyte size) blocks and encode each block with a separate PoRep, but that's not practical for many applications.

This is fundamental to any protocol that uses proof-of-replication as a proof of work replacement in consensus. Hope this helps and happy to explain more.

Best,
Ben