Building a permissionless blockchain based on proof-of-space for archiving the world wide web

Bruno Cotrim - 54406
Estudo Orientado
Mestrado em Engenharia Informática
Faculdade de Ciências, Universidade de Lisboa
fc54406@alunos.ciencias.ulisboa.pt

Abstract

As technology advances and digital content grows, web archives such as the Wayback Machine and Arquivo.pt face an exponential increase in the volume of websites and data archived, significantly increasing storage and operational costs. A scalable solution to mitigate these expenses is essential. This work investigates how these costs can be mitigated through an incentive-based, permissionless blockchain model. We introduce ArchiveMint, a blockchain system built upon Proof-of-Useful-Space and Proof-of-Time consensus mechanisms, alongside Proof-of-Storage primitives, to allow the verifiable, secure, and efficient archival of data with rapid accessibility. ArchiveMint aims to enhance energy efficiency and reduce participation costs compared to current blockchain solutions, such as Bitcoin, prioritizing the storage of useful data in its consensus rather than random useless data approaches like those employed by Chia. By incentivizing network participants to archive data through cryptocurrency rewards, ArchiveMint provides a sustainable model that reduces the rising costs of web archiving while enabling broader applications in decentralized systems, including cryptocurrencies, smart contracts, and more.

Keywords Blockchain, Distributed Systems, Proof-of-Useful-Space, Proof-of-Storage, Outsourced storage

1 Introduction

With technological advancements and the expansion of digital content, web archives like the Wayback Machine [1] and Arquivo.pt [2] have experienced an increase in the volume of websites and data archived. This in turn leads to higher storage and operational costs, creating a need for a scalable solution that can reduce rising costs effectively. In this work, our goal is to help scaling Arquivo.pt, which is overseen by FCCN (Foundation for National Scientific Computing) [3], and reduce its operational costs. Current numbers indicate that Arquivo.pt is storing 1 Petabyte (PB) of compressed data (websites, images, and videos), and is using 77 servers with 2180 vCPU, 18 TB of RAM and 1234 hard drives (5.18 PB) [4].

With this goal in mind, we will explore the use of blockchain technologies for distributing data storage. A blockchain is an implementation of a distributed ledger that allows for secure, transparent, and tamper-proof transactions, reaching consensus on a ledger view without the need for a central authority. These transactions can range from a simple cryptocurrency to executing programs through smart contracts and, most importantly in our case, distributed storage. Hence in this work we will focus on how to build a new blockchain system that can provide distributed data storage, through a self-incentive mechanism, to reduce the costs of Arquivo.pt.

However, building a blockchain for Arquivo.pt will not be easy, as no similar system has ever been built. Some of the challenges we will need to address include: (*i*) how to create a stable blockchain network where the consensus mechanism relies on energy-efficient primitives; (*ii*) how to scale the blockchain network into a fully distributed storage system that can serve Arquivo.pt's files to clients that want access its archived webpages; (*iii*) the definition of an incentive model that rewards farmers (i.e., ordinary people that will contribute to the network) for storing Arquivo.pt's files and maintaining the network; (*iv*) what role will FCCN fulfill in the network and how will it interact with the system, in such a way that makes sense for them and that allows for a great cost reduction compared with Arquivo.pt's current version; (*v*) how to avoid attacks on the blockchain network and rational farmers that will try to maximize their profit while minimizing their effort, possibly at the expense of other honest farmers; (*vi*) how to ensure that farmers do not modify the webpages attributed to them by FCCN; (*vii*) and other distributed systems concerns that should be considered such as how to efficiently disseminate messages through the network and store data in the farmers.

To solve these challenges, we will explore recent mechanisms from the literature on reliable distributed systems. In more detail, we will explore: a recent consensus mechanism called Proof-of-useful-space (PoUSp) [5], which avoids expensive energy and useless data consumption of other blockchains, instead leveraging on Arquivo.pt's data itself for the consensus layer of the blockchain; Proof-of-Data-Possession (PDP) to compliment PoUSp and allow continuous verification of file storage and integrity, so that FCCN can safely outsource its archival data to farmers; and Proof-of-Time (PoT) to ensure a real-time delay between PoUSp via the execution of a sequential function execution (e.g., Verifiable Delay Functions (VDFs) [6]), which is crucial to prevent costless and long-range attacks.

The remaining of the report is structured as follows: Section 2 covers blockchain fundamentals, including consensus mechanisms, Proof-of-Storage, and Proof-of-Time; Section 3 reviews state-of-the-art PoSp protocols and related work; Section 4 details the proposed solution, protocol, and properties; Section 5 presents current work and implementation, and Section 6 discusses future work and conclusions.

2 Background

In this section, we explore the theoretical foundations of blockchains, focusing on their structure, properties, and the ways in which they utilize nonforgeable resources to maintain decentralization.

2.1 Distributed Ledgers and Blockchains

A ledger object maintains an immutable, ordered record of transactions, ensuring the system state history is verifiable and auditable. Unlike registers or state machines, a ledger retains all historical data, enabling transparency and traceability. Transactions encompass information to ensure a functionality (e.g. exchange parties and amount in a cryptocurrency) modifying the ledger via append operations.

Distributed Ledgers replicate this object across peers, requiring consensus to agree on a consistent view of the ledger. Byzantine Agreement [7], rooted in Game Theory [8], is pivotal, ensuring honest participants reach consensus even in adversarial conditions. Nakamoto's Bitcoin [9] introduced an incentive-driven Proof-of-Work (PoW) system to achieve consensus and resist attacks like Sybil or double-spending.

Blockchains implement distributed ledgers, structuring data in linked blocks secured with cryptographic hashes. Inherently any tampering is easily detected. Permissioned blockchains (e.g. HyperLedger [10], Corda [11]) prioritize control, governance, and efficiency but rely on trusted authorities to identify participants, risking centralization. Permissionless blockchains (e.g. Bitcoin [9]) allow participants to join simply by listening to the network offering decentralization and transparency but face lower throughput and exposure to Sybil attacks, where an attacker can easily create many fake identities and obtain a majority quorum and gaining control of the system. Preventing Sybil attacks involves implementing mechanisms that make it costly to create numerous fake identities, usually by requiring nonforgeable resources.

Bitcoin [9] laid a foundation for blockchain cryptocurrencies. Being a permissionless system, its' security issues are exposed and participants are constantly under attack, from the consensus layer to the application layer [12], the adoption of these technologies is increasing consequent losses from these attacks. In Nakamoto's work [9] it is explained how the chain makes progress and is consistent if the honest participants control the majority of CPU power. However, given the increasing attacks and losses, a more theoretical security analysis was needed to define the core properties of such a system.

Garay et al. [13] provided one of the earlier rigorous security analyses of the Bitcoin backbone protocol, and identified two main properties, that robust public transaction ledgers should follow, and states as follows:

Persistence with $k \in \mathbb{N}$: Under the honest majority assumption, once a transaction is considered stable by one node, all other nodes will see the transaction in the same ledgers' position and agree on the entire prefix of the ledger. A transaction is considered stable if the block that contains it is at least k blocks deep in the ledger.

Liveness with $u \in \mathbb{N}$: Under the honest majority assumption, if all nodes in the system attempt to include a certain transaction, then after u time intervals, all nodes will see that transaction as stable in its' blockchain.

Under the majority assumption, **persistence** and **liveness** properties can be derived from the following properties that a secure blockchain should guarantee - all with high probability- [13–16]:

Common-Prefix/Consistency with $k \in \mathbb{N}$: At any point, the chains of two of the honest parties can only differ in K blocks. The deeper a block is the lower the probability of two chains differing considering a security parameter, i.e, mining difficulty.

Honest Chain Growth: At any point, the chain of honest players grows by at least T messages in the last $\frac{T}{G}$ rounds, G being the chain growth of the system.

Chain Quality: At any point, for any T consecutive blocks in an honest party chain, the fraction of honest blocks from honest parties is at least F- ideally greater than $\frac{1}{2}$- Being G and F dependent on the security parameter of the protocol, network latency and time between blocks.

2.2 Bitcoin and Proof of Work

Proof of Work (PoW) was introduced by Cynthia Dwork and Moni Naor [17] to deter email spam by requiring users to solve moderately hard cryptographic puzzles, increasing computational costs for spammers. Building on this, Adam Back's Hashcash [18, 19], proposed in 1997 and refined in 2002, used cryptographic puzzles to counter spam and DoS attacks. Users expend computational resources to generate a nonce such that the message's hash (e.g., SHA-256) starts with n leading zeros, where n is the difficulty parameter. This system was efficient, non-interactive, difficulty adjustable, and easily integrated into SMTP servers.

2.2.1 Bitcoin Protocol

As mentioned before, Bitcoin [9] is a permissionless system, thus inheriting the Byzantine Agreement problem and consequent exposure to Sybil Attacks. To mitigate these attacks, Bitcoin uses PoW to tie a participant's voting power to computational power, a nonforgeable resource instead of identities. This prevents Sybil Attacks by making it expensive to generate computational power and gain majority network control.

Bitcoin was a foundation for blockchain technologies and popularized PoW consensus in permissionless systems, most specifically Nakamoto Consensus. Nakamoto Consensus Proof-of-Work is based on Adam Back's Hashcash [19] cryptopuzzle, it consists on finding a nonce n such that when block is hashed, e.g $SHA-256$ the output of $H(p, b, n)$ starts with k leading zeros, being p the hash of the previous block, b the current block including transactions, timestamp, etc. and k the difficulty parameter. Probabilistically the work required to solve the puzzle increases exponentially with the number of zero bits and can be easily verified by executing a single hash. To compensate for hardware speed, for every 2016 blocks, this k is adjusted by the moving average of the number of blocks per hour, and a target block generation time of 10 minutes.

After solving the cryptographic puzzle, a participant (called a miner) broadcasts the newly mined block to the network. Other participants then add the block to their local blockchains. However, it is possible for multiple miners to mine blocks simultaneously, resulting in different valid chains being observed by participants—a situation known as a fork. To resolve these conflicts and maintain consistency, Bitcoin uses a policy known as the longest chain rule. The longest chain rule complements PoW by dictating that nodes should always extend the chain with the highest cumulative computational work. When forks occur, nodes initially extend the first valid block they receive at a given height. If a longer chain later appears, nodes switch to the new longest chain, ensuring eventual convergence across the network. This mechanism achieves consensus by encouraging all nodes to adopt a consistent view of the blockchain as it grows, thus providing finality over time.

Nakamoto [9] concludes that the combination of PoW and the longest chain rule defends against Sybil attacks and double-spending attacks. Modifying a previously confirmed block would require an attacker to redo the PoW for that block and all subsequent blocks while surpassing the computational work of the honest chain. This becomes increasingly infeasible as more blocks are added, assuming the majority of computational power is controlled by honest participants who outpace any competing chain. Following Game Theory, Bitcoin achieves consensus by incentivizing honest behavior by rewarding miners with coins for mining blocks, and transaction fees and penalizing dishonest behavior by requiring an extremely high computation power and consequent electricity to modify the blockchain, encouraging honesty, even if an attacker holds the majority of the power since he would have to choose between following the rules that reward him with more coins or breaking the system and the validity of his own wealth.

2.2.2 Bitcoin Security Analisys

Garay et al. [13] found Nakamoto's assumption of honest majority computational power insufficient for securing transaction ledgers. They highlighted the need for unforgeable digital signatures and faster network synchronization to outpace PoW solution rate, accounting for network delays. Eyal et al. [20] showed that mining pools compromise this assumption by enabling selfish mining, where withheld blocks create private chains, reducing the honest threshold from 50% to 33%, increasing centralization risks. Mining pools, geographic factors, and electricity costs exacerbate these risks.

Bitcoin also faces the scalability, decentralization, and security trilemma, sacrificing scalability for security and decentralization [21]. With a block generation time of 10 minutes and a 1 MB block size cap, Bitcoin processes only $\approx$ 7 transactions per second, far below systems like Visa, which can handle 56,000 transactions. Energy consumption is another concern, with Bitcoin using between 155 and 172 TWh annually, comparable to the energy use of countries like Poland [22]. While PoW secures decentralization, its environmental impact raises questions about its societal value. Alternative solutions, like PoUSp, which uses disk space as a non-forgeable resource, are the main focus of our work.

2.3 Proof of Space

Proof-of-Space (PoSp) was introduced by Dziembowski et al. [23] as an energy-efficient alternative to PoW that relies on storage space instead of computational power as a non-forgeable resource to achieve consensus on a blockchain. In PoS systems participants prove that they allocated some specific amount of data to the network. It consists of a two-phase communication protocol composed of the following:

Initialization Phase: Prover P allocates a certain amount of disk space by pre-computing and storing cryptographic data structures, often called Plots [24]. This process of creating plots is computational/time consuming, but once created, they remain in disk, being a one-time cost, amortized over the system lifetime since the stored data can be reused to continuously produce proofs.

Challenge Phase: Verifier V issues challenges that require the P to demonstrate the possession of the pre-computed data, when challenged, P derives the proof of the stored data.

Verification Phase: P submits the proof to V that should efficiently verify it with minimal computation.

The initial PoSp [23] construction was based on hard-to-pebble graphs, this type of graphs are designed to explore time-memory tradeoffs, meaning that to recompute we always need time and memory/storage, and the less storage used the more time is need to recompute it. This approach uses Direct Acyclic Graphs (DAG), where provers compute a graph where each vertex is labeled with $hash(n, i, p_1, ..., p_x)$, where n is a nonce, i the vertex index and $p_1, ..., p_x$ are the labels of the parent indexes. After the initialization phase and computing all labels, the prover computes a Merkle-Tree [25] (a tree where leaves are the hash of data blocks and non-leaf nodes are the hash of their children) from these labels and sends the Merkle Root to the verifier as a commitment. In the execution phase, the verifier chooses a subset of labels to be opened and the prover builds Merkle-Proofs from them. The verifier compares the Merkle-Roots from the proof and the commitment, allowing for more efficient and compact proofs. Probabilistically provers are incentivized to store all DAG labels due to the time/computation needed to recompute labels if some parent labels are missing from memory and the fast and cheap memory accesses to disk.

A main problem in PoSp is that a cheater could simply not store proofs in the disk and re-initialize them during the execution phase, this way swapping memory storage over time by CPU work when challenged. Dziembowski et al. [23] designed the protocol to make cheating inefficient since, to pass the verification, one needs to either dedicate N bits of memory, resultant from $\Theta(N)$ initialization steps, or recompute $\Theta(N)$ computation steps in each execution phase step. A rational participant should prefer to run a one-time CPU cost initialization phase, storing proofs in the disk at an extremely small cost, rather than being exposed to the CPU cost every time they are challenged, which depending on the challenge configuration can be exponentially hard. Due to the possibility of cheating (re-initialization on challenge), to be secure, a PoSp system honest nodes should possess the majority of the combined resources CPU + Storage space [26]. Essentially PoSp relies on a time-memory tradeoff where rational participants should be incentivized to dedicate more storage rather than CPU time to efficiently produce proofs. This first approach had large proof sizes (several MB instead of a few bytes as PoS and PoW) and these proofs are interactive (a Merkle-root commitment needs to be sent to verifiers), meaning in a Blockchain scene, a participant cannot simply join the network without sending a transaction/message to the network.

Many simpler and efficient schemes based on inverting functions have been proposed, however with no guarantees to Hellman time-memory tradeoffs, also known as Hellman attacks [27]. Hellman showed that a domain of size N can be inverted in T time when provided S bits, whenever $S \cdot T \approx N$ (e.g., $S = T \approx N^{1/2}$), in other words, an attacker would only need $\sqrt{N}$ space and time to construct a proof (inversion of permutation).

To mitigate this, Abusalah et al. [28] construction explore a loophole in Hellman that states "For Hellman's Attack to work, the function needs to be easy to evaluate in the forward direction and for PoSp usefulness its sufficient that the function table is computable in linear time".

PoSp miners perform simple disk operations to generate proofs but remain vulnerable to long-range attacks and costless simulations [29] (discussed in Section 3.1).

2.4 Proof-of-Storage

Proof-of-Space (PoSp) schemes, like the one used in Chia Blockchain [29], often rely on random data as a nonforgeable resource to demonstrate storage commitment. While effective for ensuring decentralized consensus, the allocated storage is underutilized, as the random data serves no purpose beyond the consensus protocol. To address this inefficiency, improvements have been proposed to replace this "useless" space with meaningful storage, such as Proof-of-Storage (PoSt). PoSt schemes provide cryptographic guarantees that enable a verifier v to outsource the storage of specific data D (e.g. Files) to a provider P. Unlike PoSp, which uses random data, PoSt ensures that P maintains and correctly stores the meaningful data D, while allowing V to periodically verify P's commitment to storing D ensuring data integrity, retrievability, and accountability.

PoSt can also be used independently from consensus protocols, thus the use in Cloud Services and L2 (built on top of Layer 1 networks usually via transaction) Blockchain implementations to provide storage guarantees, e.g Storacha [30] and SiaCoin [31]. Three approaches have been proposed:

Proof-of-Data Possession (PDP): PDP [32, 33] is a cryptographic protocol that allows a client to verify that a file stored on an untrusted server remains intact and unaltered without downloading the entire file. The client pre-processes the file, locally stores cryptographic metadata, uploads it to the server, and then safely deletes the local copy. Periodically, the client issues challenges to the server, which must compute proofs and send them back. The client verifies these proofs using the locally stored metadata. While PDP ensures the integrity of the file on the server, it does not address the potential corruption of individual file blocks.

Proof-of-Retrievability (POR): POR [33–35] extends the idea of PDP by verifying the integrity of the data but also ensuring that the entire file can be retrieved from the server even in partial corruption of data blocks. POR achieves this by using techniques such as erasure codes and aims to detect and possibly correct them if the corruption is under a certain threshold. The process works similarly to PDP but also adds cost of pre-processing erasure codes to add that extra functionalities.

Proof-of-Replication: PoRep [36] is a protocol that proves data D has been replicated to its own uniquely dedicated physical storage. By enforcing unique physical copies, PoRep allows a verifier to confirm that a prover is not deduplicating multiple copies of D in the same storage space.

When clients replicate data across multiple storage providers, the above ideas are often combined with Client side encryption. When the client is pre-processing the file, it should generate a unique encoding using a SecretKey Sk that should be unreproducible by any other entity, this way making it impossible for stores to share the same disk space (e.g. Replay Attacks) since each of them is associated to a unique encoding of the replicated file, apart from this pre-processing step, challenge and verification phase are specific to the constructed protocol [31, 36].

2.5 Proof of Time

PoT proves that a sequential function was executed T amount of times, being sequentially the most important aspect here. This means that the prover cannot simply add more machines to make the function execute faster [37]. This time parameter T can be translated into real (wall-clock) time. PoT is implemented using verifiable delay functions (VDF) [6, 38], a deterministic function that guarantees the output is unique and can be verified quickly with a small proof while ensuring that its computation requires a measurable amount of real-world time [39]. This helps preventing costless simulation and long-range attacks as well as making functions hard to evaluate in the forward direction, essential to PoSp schemes.

2.5.1 Wesolowski Verifiable Delay Function Scheme

Wesolowski VDF is based on squaring in a group of unknown order $G \in \mathbb{Z}_N^*$, where $N = p \cdot q$, being p and q the product of two distinct prime numbers and the size of N dependent on a security parameter K. The scheme is specified by the following two algorithm [6, 29]:

- **VDF.solve(*c,t*)** $\to \tau = (\tau.y, \tau.\pi, \tau.c = c, \tau.t = t)$
  Being the input c a challenge $c \in \{0,1\}^n$, and t time parameter/iterations. The output τ consists on a tuple containing the input (for convenience) along with an output of the function $\tau.y$, and $\tau.\pi$ is a proof that $\tau.y$ has been correctly computed. This output can be computed given the equation $y = c^{2^t} \mod N$, this can be calculated by squaring c sequentially t times, e.g. $c \to c^2 \to c^{2^2} \to \cdots \to c^{2^t}$, an important aspect is that there is no shortcut to this computation without knowing the factorization $N = p \cdot q$, thus taking $2T$ exponentiations.
- **VDF.verify**$(\tau) \in \{\text{reject}, \text{accept}\}$. This method returns either `true` or `false` based on whether it successfully verifies the output. It ensures that the output was indeed calculated with the specified delay T in the group G. The verification process involves computing $2^T \mod N$, which is efficient and requires only $\log_2 T$ multiplications.

Along with these two specified algorithms, Wesolowski [6] also specified two security properties:

(i) **Uniqueness:** It is hard to create a false output y and still have a valid proof π. Meaning, for the same input, the output y will always be unique but the proof π can change.

(ii) **Sequentialitiy:** An attacker that makes less than T sequential steps will not find an accepting proof on c. The number of sequential steps bounds the attacker and, even using high parallelism, the VDF output cannot be computed faster than the speed of a single CPU core computing a step of the VDF computation.

An important aspect of this scheme is that being a trapdoor VDF, if an RSA group as above, the security depends on the secrecy of the factorization $N = p \cdot q$, if this factorization is revealed a shortcut to the computation can take place braking the scheme. To deal with this, imaginary quadratic fields as the group of unknown order can be used, which makes it possible to sample a group without revealing its order given a random number [6, 40].

2.5.2 Slow-Timed Hash Function

A Slow-Timed Hash Function (sloth), as the name indicates, was proposed by Arjen Lenstra and Benjamin Wesolowski [41] to be a hash function where its computation is designed to be as slow as desired, following two design criteria:

- It must be possible to choose parameters, such that, computing sloth takes at least ω seconds, independently of the computing resources available.
- The hash verification process should be modest compared to the sloth computation time ω.

This scheme is based on modular square roots. First a prime p, such that, $p \equiv 3 \mod 4$ and $p \geq 2^{2k}$ being k a security parameter related to the bit length of p. The sloth computation is based on modular square roots, $y = x^{\frac{p+1}{4}} \mod p$, along with an invertible permutation, both are mathematically forced to be calculated sequentially. The verification is squaring, thus it's faster, and it's defined by $x = y^2 \mod p$, however, the inverse permutation is applied. Both operations execute t, where T is a time parameter that can be mapped to wall-clock time. Verification is faster because it involves simple modular squaring, which is less computationally intensive than modular exponentiation.

Given this, we can also deduce the main weakness of this algorithm. The process is not asymptotically efficiently verifiable, meaning that verification can only be faster to a constant factor, thus the verification time increases with the input size (p and t). This gap in computation and verification is not big enough to allow the scheme to qualify for a VDF [6]. However, in our work, we aim to use this scheme to create small delays of up to 1-2 seconds, making the verification time negligible, and given the fact that this is not a trapdoor function and doesn´t need trusted setup, a verifiable slow hash function suits perfectly in our construction.

3 Related Work

3.1 Chia Blockchain

Chia [29] is a cryptocurrency implemented on a blockchain system that uses a combination of Proof-of-Space and Proof-of-time for its consensus mechanism. It aimed mainly to enhance PoW sustainability problem, by using disk space as nonforgeable resource instead of computational power, and interactivity problems of previous systems (e.g., Spacemint [24]) based on the Dziembowski et. al. [23] PoS scheme, where provers need first to send a commitment as a blockchain transaction before being able to join the network, meaning they can't simply participate by listening to network and if miners do not accept the transaction, one might never be able to participate. To solve this, Chia PoS is based on Abusalah et. al. [28] approach that is both proof size efficient and non-interactive.

Chia's PoSp [42] implementation nests 6 times (7 tables) the Abusalah et al. [28] structure design, which increases exponentially the computational difficulty of solving the problem, in order words, it makes the scheme exponentially more resistant to Hellman Attacks [27]. This construction also has different heuristics to make it more practical (e.g., a Matching Function M, a collation Function C, etc.). Each table has 2^K entries, being K a security parameter with a minimum value of 32, that generates a table with around 101.4 GiB. Each entry in a table points to entries in the previous table, except for the first table which contains a pair of integers called "x-values." A proof of space is formed by 64 x-values with a specific mathematical relationship. The computational complexity of generating this structure incentivizes honest participants to store proofs in disk instead of computing them on the fly, since probabilistically they could not produce them, constantly, either fast enough or without using an immense amount of resources while honest participants have a one-time cost of generating them and an extremely small cost of storing them on disk over time.

Chia's PoT [42] is based on the Wesolowski Scheme [6], utilizing repeated squaring in classgroups of unknown order, as mentioned in 2.5.1. Unlike RSA-based groups, classgroups with large prime discriminants eliminate the need for trusted setup (e.g., via MPC) but are less optimized and tested in practice. Chia's PoT generates 1024-bit prime discriminants and proofs derived from blockchain randomness. A critical feature of this PoT is its sequential nature, prohibiting parallelization and tying computation speed to a single CPU core, thus limiting the advantage to current hardware, which is limited.

3.1.1 Chia's Consensus Algorithm

There are 2 parties involved in Chia consensus algorithm, a farmer is a participant who contributes with disk space to secure the blockchain and produces PoSp proofs, and Timelords are responsible for creating PoT proofs and ensuring that a real-time delay between blocks occurs. The combination of both will be crucial to maintaining the security of the blockchain.

The **Farmer algorithm** operates in two phases: plotting and farming. In the plotting phase, storage is allocated to create reusable PoSp plots. During farming the farmer responds to network challenges by identifying the highest-quality proof in their plots. Upon finding the best PoSp, the farmer produces and broadcasts a non-finalized block. If the farmer's block has the best quality, it extends the blockchain, earning them rewards and transaction fees. This algorithm protects Chia from Sybil attacks.

The **Timelord Algorithm** finalizes non-finalized blocks by computing a PoT VDF using PoSp as a challenge, adding a sequential delay to securely extend the chain and prevent long-range and costless simulation attacks. Timelords work on the chain with the highest accumulated space, calculating PoT for signage points and block infusion. They run three parallel VDF chains, requiring fast CPU cores for efficiency, and ignore challenges or blocks with less weight to prevent withholding attacks. Timelords ensure decentralization and network security, though they do not earn rewards.

In Chia, farmers and Timelords work together to extend the chain with the highest cumulative space, as determined by the quality of PoSp. Each non-finalized block has its PoSp quality evaluated, and the time required to compute the VDF is inversely proportional to this quality. This means that higher-quality blocks are extended faster by Timelords because they require less time for VDF computation. To maximize their chances of winning a block, farmers are incentivized not only to allocate more space but also to disseminate their proofs quickly. The process relies on a combination of selecting blocks with the highest quality and ensuring they are broadcast efficiently to extend the honest chain with little delay. This dynamic helps mitigate withholding attacks, as timely dissemination ensures that only the best-quality proofs are used to finalize blocks. The probability of winning a block is directly proportional to the amount of space allocated by a farmer. Allocating more space allows the farmer to compute more proofs, increasing the likelihood of finding a proof with a quality value close to the challenge.

3.1.2 Chia's Relevant Attacks and Analisys

Long-range Attacks: As discussed in 2.3, the security of Bitcoin relies on an attacker requiring computational power exceeding the historical average to grow a longer chain, making such attacks impractical. In contrast, Chia's PoSp can be computed rapidly, potentially allowing an attacker with sufficient space to bootstrap a heavier chain in a short time. To address this vulnerability, Chia alternates PoSp with PoT. When a farm generates a PoSp, it serves as input to the timelords PoT computation. The output of this PoT is then used as input for the next PoSp and the process repeats itself. This mechanism introduces an enforced real-time delay between blocks, similar to Bitcoin's mining process. The attacker's chain growth is bound to his single-core hardware speed, which even with the fastest hardware today that advantage is relatively small. This bound delay/growth rate ensures that to outpace the honest network, an attacker would need to expend real-time between blocks, which effectively limits their ability to grow a heavier chain unless they possess both significant storage space and faster timelord hardware, which is highly impractical.

Grinding Attacks: In Bitcoin a miner can work towards solving two crypto-puzzles at the same time, by tweaking, however, this implies that they would need to split the computational resources, offering no advantage. In contrast, Chia PoSp allows malicious miners to generate multiple challenges at no cost (e.g., by tweaking a block timestamp), increasing their success rate. This leads to grinding/digging attacks, where the attacker picks the challenge that increases its odds of winning a block and/or future blocks, allowing him to deviate from honest farming. Inspired by Spacemint [24], Chia splits the chain into a trunk chain, that contains the PoSp and PoT proofs, and the other chain is called foliage that contains a block payload and a signature that binds both chains. This way all the challenges come from previous trunk values, protecting against grinding attacks, since trying multiple values in the foliage changes nothing in the trunk and in the odds of winning blocks.

Double Dipping Attacks: In double dipping instead of trying to extend some block in many ways, an attacker tries to extend multiple blocks, posing the same dangers as the last attack. Penalties have been suggested by Spacemint [24] to address this issue, Chia states that this is not a solution only a deterrent, instead of completely mitigating this, Chia makes this part of the protocol. Farmers and Timelords work on extending the first K blocks at every depth. Cohen et. al. [29] analysis concluded that setting this security parameter to 1 would require the 73% of the network to be honest, which is high compared to bitcoin 50%. Increasing k to 3 allows increasing this honest threshold to $\approx 61.5\%$, at the cost of more effort from participants and increased time to achieve consensus.

One of the main drawbacks of Chia is the weaker assumption of $\approx 61.5\%$ honest participants when compared to Bitcoin, this concern is amplified by the nature of the VDF and PoSp combination. VDFs and classgroups of unknown order with large prime discriminants d are not well tested or optimized in the real world and need further investigation. Combining PoSp with PoT to achieve consensus makes an attacker with a faster VDF "multiply" its space [43].

Chia aims to be a more sustainable solution than Bitcoin, achieving 0.13 TWh energy use, $\approx 99.9\%$ of Bitcoins at the time of this measurement. However, energy efficiency is not the only requirement of sustainability. This protocol is heavy I/O intensive on storage hardware leading to increased electronic waste. Space also poses no other use than to store proofs required for network security, by analyzing Chia dashboard [44], we can notice 18.7 EiB (21 Million Tb) are being used to farm plots. This lack of data utility leads to under-utilization/inefficient usage of Disk Space. This work will study proposed "Bread Pudding" Algorithms that aim to improve this storage inefficiency by allowing the storage of useful data within the cryptographic proofs, allowing for more general purpose and more storage efficient systems.

3.2 Chia Bread Pudding

Chia Bread Pudding was introduced by Mónica Jin [5], aiming to reduce disk waste resulting from the PoSp schemes such as the one used in Chia [29], by allowing the usage of pre-existing data (e.g. user files) within the proof mechanism. This way storing PoSp proofs and useful data simultaneously provides better utilization of space resources.

Being an improved adaptation of the Chia blockchain it maintains some of its main properties. In this protocol, farmers undergo an initialization process to generate proofs before participating in the system. Challenge generation works similarly to Chia in the sense that challenges are derived from the previous block thus benefiting from the blockchain randomness and ensuring precedence relations. While adapting Chia PoSp, Costless Simulation attacks, and Long-Range attacks were also a concern of this work due to the cheap process of proof initialization.

The protocol remained non-interactive meaning farmers can use their local data for the initialization step and are not required to send a commitment to the blockchain, being able to simply join the network by listening to the system.

To determine the winner the protocol specifies a quality function, for each proof, that reflects the amount of space a farmer is allocating to the network, meaning, the probability of a farmer extending a chain is proportional to the amount of space allocated. Similarly to Chia, farmers follow the chain with the most accumulated space/quality, thus the block with the highest quality at a certain depth is chosen as an extension of the chain.

The costless simulation attack approach from Chia was not modified, the proofs of a block are decoupled from the payload into two separate chains.

The main difference of this protocol when compared to Chia PoSp is how it deals with Long-Range Attacks and the proof scheme, instead of alternating between PoT and PoSp, this protocol uses Verifiable Delay Encoders/Decoders to guarantee a delay between blocks and making proof generation more difficult. Here are the modified PoSp algorithms:

Initialization: The first step of this phase is the encryption of the user data using Cypher Block Chaining (CBC), which helps guarantee the user's data privacy and increases the entropy of the proofs. This protocol uses a Vector Commitment (VC) Scheme, in particular, a Merkle Tree [45]. A Merkle Tree is built using the encrypted blocks, where the Hash of these Encrypted blocks becomes tree leaves and iteratively non-leaf nodes are the hash of their children, producing a Merkle Root. To finish this phase every Merkle node is encoded using a Verifiable Delay Encoder (VDE). Similar to a VDF, the VDE encoding computation is sequential, and the encoding process is time-consuming while decoding is fast. By making the initialization step time-consuming honest farmers are incentivized to store proofs in disk and reutilize them having a one-time generation cost, instead of running the re-initialization during the challenge phase, making it impractical to perform a time-memory tradeoff to generate proofs at the go.

Proof Generation: In this phase, farmers receive challenges from the network (e.g. derived from the proof of the previous block). Upon receiving a challenge, farmers search their disk for the Merkle root with the highest quality relative to the challenge. Once the best Merkle tree is identified, the farmer generates a cryptographic proof (e.g., $proof = E3||E4||E12||E5678||E12345678$) composed by a leaf ($E3$) and the corresponding Merkle Path, which includes is a set of intermediate nodes needed to compute the tree's root hash. This path can only be efficiently constructed if all leaf nodes are stored on disk, proving that the farmer is genuinely storing the data. However, a malicious farmer might attempt to store only a single path, falsely claiming to store all leaf nodes. To counter this, the protocol makes use of the blockchain randomness to randomly target a specific leaf, determined by $challenge \mod total\_leaves$. This ensures that different portions of the Merkle trees are probabilistically selected over time, making it impractical for attackers to discard nodes and incentivizing honest storage. Once the proof is generated, the farmer extends the blockchain. This proof generation process is efficient for honest miners, as proofs can be reused over the blockchain's lifespan, effectively amortizing the initialization cost.

Proof Verification: This phase mainly consists of 2 steps, decoding and validation. First, if a block's proof has a quality below a certain difficulty threshold, the block is discarded. If the block passes this filter, it enters the decoding phase where a Verifiable Delay Decoder (VDD) applies the inverse operation of a VDE and decodes the different nodes in the Merkle Tree, obtaining the original Merkle Path that corresponds to the proof's Merkle Root. Finally, in the validation phase, the Merkle Root resultant from iterative hashing of these decoded nodes matches the Merkle Root present in the proof, if it does, the block is deemed valid. To complete this phase, the challenge from the block should belong to the three most recent PoT proofs, and miners given a short time frame (9.375 to 28.125 seconds) to generate and broadcast a valid proof for each challenge, thus adhering to real-time delays as present in chia, preventing long-range attacks. This limited time window also discourages miners from generating data on demand making this process expensive thus incentivizing the initialization and reusal of proofs in disk.

The main part of this protocol is that the user's data encoded within the Merkle Trees can simply decode and traverse all Merkle Leaves and reconstruct the original data, thus allowing for a useful storage scheme. User's data modification is also possible however it is required to run the initialization phase over that data.

There are several parameters to be adjusted, n number of leaves of the Merkle tree, f the fanout of the Merkle tree, m node size, and t VDE execution time, all affecting proof sizes, useful space ratio and initialization time.

This work successfully shows that it's possible to achieve 50% of the useless data can be repurposed with useful data while maintaining most of the security properties present in Chia, thus helping reduce storage waste.

One of the main limitations addressed by the author included the lack of a proper VDE implementation. Our work PoSp scheme will be mainly based on creating a Chia Bread Pudding [5] inspired practical implementation, being inspired by the Merkle Tree proofs, challenges, and time consuming encoding process, however, we will tweak little implementation details important to the security of a real-world system, return the concept of PoT and PoSp alternation from Chia, address the encoding scheme limitation, and possibly increasing the useful space ratio to simply the size of the encoding proof and user's data, reducing the random storage waste even more, all these modifications will be addressed along our work.

3.3 Filecoin Blockchain

Filecoin [46] is a decentralized storage network that turns traditional cloud storage into an algorithmic market. Built on top of the Interplanetary File System [47], it enables participants to rent out unused space, and clients can choose storage providers based on cost, redundancy, and location, making it highly flexible and user-oriented. This market runs on a blockchain with a native token, Filecoin (FIL), which incentivizes storage providers to participate actively and maintain a quality service. Filecoin is essentially an economic layer on top of IPFS, a perfect infrastructure for decentralizing data through content addressing and implementing smart contracts.

There are three participants in this Decentralized Storage Network. **Clients** pay to store and retrieve data. **Storage miners** provide storage by pledging collateral, responding to PUT requests and committing to store data for a specific time, generating Proofs-of-Spacetime (PoSpT) to prove their commitment. If they fail to provide valid proofs, they lose part of their collateral. **Storage miners** also mine new blocks and earn rewards and transaction fees. Retrieval miners help with the initial storage setup and respond to GET requests by delivering data to clients, with storage miners often also acting as retrieval miners.

Filecoin consensus consists of two types of proofs, Proofs of Replication, mentioned in Section 2.3, and Proofs of Spacetime (PoSpT), where miners provide public proof that a given data encoding is being stored continuously over time.

In the PoRep scheme, the miner offers storage sectors (usually 32/64 GB) and agrees with the client on a storage deal through the Storage Market. The data is placed in the miner's sector, and the PoRep lifecycle begins. The sector undergoes a slow, sequential sealing process to transform the data into a replica and generate a proof, which is compressed using zk-SNARKs [48–50] and sent to the network as a commitment. Challenges have a 30-minute time window, and the slow sealing process prevents miners from falsely claiming more storage. To reduce communication complexity, Filecoin introduced PoSpT, where a prover produces recursive sequential PoRep proofs for a specific number of iterations.

Filecoin currently extends PoS consensus by replacing the concept of stake with storage capacity. Through a process known as expected consensus, one or more participants are elected in each round, with their probability of winning a block being proportional to their allocated storage. Because multiple valid blocks can be created in a single round, these blocks are grouped together into a structure called a tipset.

Questions have been raised about the expected consensus mechanism in Filecoin; however,other limitations include its slow data retrieval process. This is due to the time-consuming sealing and unsealing procedures, which can take 5–10 minutes to retrieve a 1 MiB file [51]. As a result, Filecoin is better suited for cold storage scenarios where data retrieval is infrequent or does not require high speed. The complex sealing process also requires more hardware investment to participate in the system.

4 ArchiveMint

4.1 Motivation

Currently, FCCN's Arquivo.pt centralized service faces escalating storage and operational costs due to the accumulation and rapid expansion of digital content to be archived, creating a need for a more scalable solution. Our goal is to scale FCCN and consequently reduce costs by building a new blockchain system that provides distributed storage where clients can safely outsource their data and network participants are incentivized to provide storage services and participate in consensus. Simultaneously, our proposal aims to mitigate both Bitcoin [9] energy and Chia [29] storage waste

4.2 Challenges and solutions

(i) How to create a stable blockchain network where the consensus mechanism relies on energy efficient primitives?

To mitigate Bitcoin [9] energy and Chia [29] storage waste we introduce a Proof-of-useful-space (PoUSp) consensus scheme, where storage space is used as a nonforgeable resource instead of computational power. In this scheme, farmers have an initial computational cost of producing proofs that can be stored in disk at a negligible energy cost over time. Proofs are reused over time amortizing initialization costs. Opposite to Bitcoin, where energy demand (due to PoW difficulty increase) scales with the number of participants, our scheme storage requirements remain constant. These properties allow for a more energy-efficient system compared to Bitcoin.

Chia already introduced a PoSp scheme that enhances energy efficiency. However, its main downside is that it fills storage space with random data, which serves no purpose other than maintaining the blockchain consensus. In contrast, our PoUSp enables the encoding of useful data within the stored proofs, leading to more efficient utilization of space resources.

(ii)How to scale the blockchain network into a fully distributed storage system able to securely serve Arquivo.pt files to clients?

To scale the blockchain into a fully distributed storage system for Arquivo.pt, a verifiable storage mechanism is essential. While PoUSp allows consensus using space as a nonforgeable resource and encode useful data within proofs, it alone cannot guarantee data integrity or storage checks. To address this, we propose a PDP scheme that complements PoUSp by providing verifiable storage. This allows FCCN to securely outsource Arquivo.pt files to blockchain participants.

In our design, farmers allocate storage space to the network. Using the PDP scheme, FCCN can send files to selected farmers for archival and continuously verify their integrity. When archiving a file, the farmer pre-processes it using the PoUSp scheme, encoding the file into PoUSp proofs. To retrieve or verify the file, the farmer simply fetches the file or proof from the PoUSp storage.

When combined, both schemes allow the construction of a distributed storage network, and consequently, FCCN's Arquivo.pt can safely be distributed with guarantees of storage and integrity.

(iii)How to define an incentive model that rewards farmers?

The security and sustainability of our system rely on the incentives/penalties farmers get for providing storage services to the network.

Our construction supports a native cryptocurrency that will serve as an incentive to farmers. Farmers can be rewarded in two main ways, either by maintaining the blockchain and processing its transactions or by providing storage to FCCN. This topic's details are still under study since it relies first on building the blockchain and PDP scheme.

(iv)What is the role off FCCN and how it interacts with the system?

FCCN will have two main roles in our construction. First, FCCN is responsible for securely encoding files and generating cryptographic metadata to verify files. It then creates storage contracts and sends the encrypted files and associated contract details to farmers for storage.

Second, in our PDP scheme, FCCN uses AES symmetric encryption to generate unique cryptographic file encodings. To maintain security, FCCN's secret key, essential for encryption/decryption, must remain confidential. To manage this, FCCN acts as a proxy where it requests farmers archived data from farmers, decrypts the files using its secret key, and securely forwards the decrypted files to the clients.

The cost of FCCN's file distribution role is amortized over time. Once files are archived, FCCN no longer needs to perform additional work related to those files. The cost of serving as a proxy is minimal, especially when compared to current centralized solutions, given today's hardware and software optimizations for AES encryption. By focusing solely on these efficient roles, FCCN significantly reduces storage and operational costs, leading to long-term cost savings.

4.3 ArchiveMint consensus protocol

4.3.1 Proof-of-Useful-Space

Recalling Section 2.3, a PoSp scheme aims to use storage space as a nonforgeable resource to achieve consensus, where a farmer proves to the network they are storing a specific amount of data, and the network should be able to efficiently verify it. It consists of three components: plotting, farming, and verifying.

This scheme tackles two significant issues simultaneously. First, it mitigates the storage inefficiency associated with Chia by enabling the integration of useful data into the proofs required for maintaining blockchain consensus. Second, by solving the first issue, it facilitates the storage of FCCN web files within these proofs, allowing farmers to store meaningful data as part of the consensus mechanism. However, it is crucial to note that in this scheme, farmers can use any type of useful data but does not inherently ensure data integrity or verifiable storage. Later, in Section 4.5.1, it will be shown how to enhance this scheme to allow for verifiable storage of FCCN files.

As mentioned in Section 3.2 our PoUSp Scheme will be inspired by Mónicas Jin's Chia Bread Pudding [5], the main difference being the Merkle Trees encoding algorithm and return to the PoUSp and PoT alternation presented in Chia [29], due to the lack a secure practical implementation and a proper VDE. This way we can maintain Chia properties while improving them with Chia Bread Pudding useful data encodings.

4.3.2 Plotting

In this process, a farmer initializes a specific amount of storage space filling it with proofs. To address the Hellman time-memory tradeoffs, mentioned in Section 2.3, and the usefulness of this scheme, the function should not be easy to evaluate in the forward direction, meaning plotting should be an expensive time-consuming process that incentives rational farmers to store proofs in the disk, amortizing initialization cost over time rather than trying to generate proofs on the go, because probabilistically they could not generate them fast enough or without spending immense amount of resources. The scheme's probability of having a winning proof should also be proportional to the dedicated space.

Being an adaptation of Chia Bread Pudding [5], the plotting process begins by loading the farmer's useful data into memory and splitting it into chunks of size cs. For each chunk, we compute a Merkle Tree, splitting its data into n leaves and a fanout of f. Our Merkle Tree construction is also based on Ralph Merkle's [45] method, with a few additions to prevent second pre-image attacks [52] (which involve producing a specific Merkle Root from different inputs). A byte with the index position of each leaf node is appended to its hash, different hash algorithms are used for leaves and internal nodes, and a fixed Merkle Tree length and block size m are required. This prevents attackers from reconstructing trees from intermediate nodes (Merkle Trees that represent less disk space) as if they were the expected original leaves, thus representing an invalid Merkle Tree. The result of this computation is a Merkle Root Commitment that represents the original useful data. This root can later be used to produce PoUSp proofs and can be easily verified using Merkle Proofs.

As mentioned before, this needs to be a time-consuming process, instead of relying on a VDE to encode a three with a sequential computation, we implemented a SLOTH [41] that allows the verification that a hash function took T real-time to compute. Given a Merkle Root and a farmer public key Pk, the farmer computes $SLOTH(root + Pk)$, and the result of this computation will be a hash that we can compare its quality to a challenge and be later used as proof. Being the proof output dependent on the result of this time-consuming computation, it prevents Hellman attacks since the proof production rate is limited by T, making this process computationally and time-consuming incentivizing the storage and reusal of proofs over time and making it expensive for replotting attacks where farmers dedicate space only when a challenge is received and not over time. The result of SLOTH and the farmer's useful data chunks are then stored on disk and can be used as a PoUSp.

Combining Merkle Trees with SLOTH allows for an even more storage-efficient scheme than the one explored in Chia Bread Pudding because it stores the encoded tree as a whole which includes a lot of metadata. With this adaptation, nodes only need to store the leaves and sloth computation and proof, reducing metadata to only SLOTH proof size (around 256 bits).

4.3.3 Challenge generation and validation

Periodically, the network challenges farmers (e.g., derived from PoT of the last block) to provide PoUSp. This is an opportunity for them to prove that they are dedicating some amount of space to the network and a chance to win rewards from farming the block and transaction fees.

A proof consists of an opening of the VC scheme, in this case, a Merkle Root, where a subset of hashes can be used to prove a specific leaf is present in a tree, this subset is composed of the leaf itself and its corresponding Merkle Path, as explored in Section 3.2.

When a farmer is challenged, it searches the disk for the data that produced the Merkle tree with the SLOTH output with the best quality when evaluated with a quality function (e.g. bitwise difference). Once this Merkle Tree is found the process works similarly to Chia Bread Pudding [5] with slight changes in the proof composition and validation. This PoUSp scheme relies on the storage of data that produced a certain Merkle Tree and sloth, however, to ensure farmers did not discard some leaves/data the protocol randomly selects leaves and corresponding Merkle Paths, this randomness is derived from the challenge itself, and the chosen leaf given by the expression $leaf\_index = challenge \mod n\_leaves$. Finally, the farmer recomputes the Merkle Root and produces the Merkle Proof composed by the selected leaf and its corresponding Merkle Path, additionally, the final PoUSp proof will also contain the SLOTH output, both components can be easily verifiable by the network. After producing the PoUSp proof the farmer extends the blockchain and broadcasts a non-finalized block where timelords will compute a PoT finalizing it.

When a farmer or a timelord receives a block, they will validate its PoUSp by doing the following verifications:

1. Recompute the Merkle Root by sequentially hashing the leaf and Merkle Path and verify if it is equal to the one provided in the proof.
2. Verify if the encoded prefix in the leaf matches the index given by $challenge \mod n\_leaves$. (Preimage Attack Prevention)
3. Verify if $Merkle\_Path.size == \log_2 n\_leaves$ (Preimage Attack Prevention)
4. Verify if $SLOTH.verify(root + farmer\_Pk)$ returns true.
5. verify if the presented leaf has the expected size (e.g. 64 bytes)

If all the presented validations are verified, the proof is valid, and depending on whether the verifier is a timelord or a farmer, they can proceed to their expected protocol algorithm.

4.3.4 Quality Function

The quality function has several purposes, it helps to compare proofs in a deterministic way where better proofs correspond to a higher-quality function output. It also makes farming more efficient, since farmers only need to read the stored Merkle Root and compute the quality function without reading/computing the entire proof, since full proofs are only produced when the best quality Merkle Root is found. The probability of getting matching bits follows uniform random distribution and every bit position has an equal likelihood of matching, this makes this function fair, deterministic, and unbiased, which makes the likelihood of finding a better quality increase with the increase of allocated space. This incentivizes farmers to dedicate more space to the network since the more proofs they store on disk the higher the chance of finding a proof with a quality good enough to produce a winning block. Currently, in our implementation,n this quality function is given by $quality = \frac{bitwise\_difference(challenge, proof.root)}{max\_difference}$ and a difficulty parameter d that works similar to a plot filter in Chia [29], where the first k bits of the challenge should match the first d bits of the proof, where d can be dynamically adjusted to moderate the amount of produced proofs and making it more difficult to perform replotting attacks.

4.4 Relative attacks and countermeasures

The PoUSp data structure is inspired by Chia Bread Pudding [5], however, the protocol participants can either be a Farmer or a Timelord, and their behaviors are inspired by Chia [29] and how they prevent grinding, double dipping, and long-range attacks and how they work on the longest accumulated chain space.

Long-range attacks: Similarly to Chia, in our PoUSp scheme, proofs are computed rapidly, allowing an attacker with sufficient space to bootstrap a heavier chain in a short time, opposite to Bitcoin where an attacker is required to have computational power above the historical average. To mitigate this, we also alternate between PoUSp and PoT, meaning one serves as input to the other alternately. By requiring PoT between PoUSp, an attacker is required to spend a real-time delay between computing the VDF, which makes it impractical to grow a chain larger than the honest chain since the attacker's chain growth rate is bound to his single-core hardware speed, meaning, an attacker would need to compute the VDF significantly faster (along with sufficient space) than the honest participants, which given today's possible hardware advantage it's not possible. To make it even more difficult the time spent in VDF is inversely proportional to the quality of the PoUSp, meaning the better the quality the faster the VDF is computed which prioritizes the chain with the most accumulated space to grow faster.

Grinding attacks: To prevent farmers from modifying the block payload to produce challenges that increase the odds of winning blocks, we follow the Chia proposal where the PoUSp and PoT proofs are decoupled from the payload of the block, meaning that modifying the block payload does not produce changes on the proof's of a block and consequently do not affect consensus odds of winning blocks.

Double Dipping attacks: Farmers can also extend multiple blocks, again, inspired by Chia proposal, we make this part of the protocol instead of using penalties. Farmers and Timelords work on the first K blocks at every depth. Being K

is a security parameter directly affecting the honest threshold, where higher values allow for higher honest thresholds at the cost of more work participants perform.

Farmer Algorithm: After plotting, the farmer starts the main farming loop. The algorithm begins when a farmer receives a finalized(block with a valid VDF and PoUSp), valid (extend an existing block), and fresh (valid but never seen) at the depth i and ignores it if the block VDF is invalid. As aforementioned, farmers work on the first K chains they see, so if they have seen more than K blocks at depth, they ignore the block. Otherwise, it increments the counter of blocks seen at this depth. Then, the output of the VDF is signed and the hash of this signature is used as input to PoUSp. Finally, the new PoUSp is computed, and the new block's payload and it's corresponding signature are generated, broadcasted to the network and the chain is updated (e.g. local view update and dissemination).

Timelord Algorithm: When a timelord receives a non-finalized block that lacks a VDF output but has a valid PoUSp, the block's depth is checked. The new block is ignored if there are already K finalized blocks at the same depth i. Otherwise, the timelord calculates the required number of VDF iterations. The time t spent on the VDF is inversely proportional to the quality of the PoUSp and is defined as $t = \frac{T}{Q}$, where T is a constant regulating the VDF speed to maintain the desired block time, and Q is the quality of the PoUSp of the non-finalized block. Two scenarios then arise:

1. **Fewer than K finalized/running blocks at depth** i: If the number of finalized blocks and blocks with ongoing VDF computations at depth i is less than K, the timelord begins computing the VDF for the received block and increments the counter of blocks at this depth.
2. K **or more finalized/running blocks at depth** i: If there are already K finalized or running blocks at depth i, the timelord compares the VDF time of the slowest running thread at i with the VDF time of the new block. If the new block's VDF time is faster, the timelord aborts the slowest VDF thread and begins computing the VDF for the new block.

By prioritizing VDF computations for blocks with faster VDF times (which correspond to higher PoUSp quality), this algorithm ensures timelords focus on blocks that disseminate quickly and have greater quality. This approach maintains the chain with the highest accumulated quality. Additionally, farmers are incentivized to disseminate blocks quickly. High-quality blocks that arrive late may fail to outpace the slowest VDF thread, discouraging withholding attacks. As a result, this mechanism works to extend the chain with the most accumulated space and minimize delays.

We opted to implement the Wesolowski's VDF [6] scheme mentioned in Section 2.5.1. Our primary distinction compared to Chia is that, in this first version, we rely on a trusted party to generate the secret factorization of N.

4.5 From a blockchain to a distributed storage network

The explored PoUSp and PoT mechanisms allow the construction of a secure blockchain that achieves consensus on a consistent view of the ledger by using space as a nonforgeable resource. However, the PoUSp scheme allows farmers to encode useful data within their proofs and, to outsource the storage of web archives, it lacks a method to verify the integrity and continuous storage of data. We propose implementing a Proof-of-Data-Possession (PDP) scheme that complements PoUSp with verifiable storage. This scheme should address several key challenges:

- **Outsourcing Attacks**: Preventing farmers from pretending to store data by quickly fetching it from other farmers.
- **Efficient Data Retrieval**: Ensuring FCCN can quickly retrieve data.
- **Minimal FCCN Costs**: Reducing FCCN's costs to achieve savings relative to the existing centralized solution.

4.5.1 Proof-of-Data-Possession Scheme

Our PDP scheme will be a central component of our solution and a subject of future research. In this preliminary version, the scheme consists of:

Recalling Section 2.4, in a PDP Scheme, the client preprocesses a file to generate a unique cryptographic encoding and metadata for later verification. A critical feature of this encoding is that it must be reproducible and decodable only by the client. Otherwise, the scheme would be vulnerable to outsourcing attacks, as farmers could generate the encoding by requesting the file from others. In our initial scheme, we use symmetric encryption (e.g., AES-256), with Hash(farmerPk + fileId) as the Salt/IV to produce unique encodings for each file and farmer. To facilitate efficient storage verification, we employ the Merkle Tree [45] data structure, enabling easy verification through Merkle Proofs, similar to our PoUSp scheme.

Storage Phase:

1. Initially a farmer dedicates storage space (e.g 32 GB) to the network, then the farmer responds to FCCN's archival requests until it has no more space to fill.
2. When archiving a file on a farmer, FCCN encrypts the file using the AES-256 algorithm, it's secret key Sk, and Hash(farmerPk + fileId) as the salt. FCCN then computes the Merkle Root for this encrypted file.

3. FCCN sends the encrypted file and a signed contract (FCCNId, farmerPk, Merkle Root) to the farmer, and stores the contract metadata.
4. The farmer processes the file through the plotting mechanism to produce PoUSp proofs. Once completed, the farmer submits the signed contract to the blockchain, enabling the generation of PDP proofs and file accesses, and earning storage rewards over time.

Challenge Phase:

1. Periodically, FCCN or the network (this aspect is still under study) issues challenges to farmers to prove storage of specific parts of the file.
2. Upon receiving a challenge, the farmer extracts the encrypted file from the PoUSp encodings, recomputes the Merkle Root, and generates a Merkle Proof for the targeted leaf. This constitutes the PDP proof.
3. The verifier (e.g. FCCN or the network) reconstructs the Merkle Root from the proof and compares it to the Merkle Root in the signed contract. If the roots match, the proof is deemed valid.

An essential aspect of this scheme is allowing clients access to web archive files. Our initial approach is to use an FCCN Arquivo.pt proxy between clients and farmers. In this setup, the client requests a specific file to the web archive server, which contains information about which farmers are storing the file, along with their IP addresses. The web archive server then requests the encrypted file from the farmer, decrypts it using the secret key and the corresponding salt, and forwards the decrypted file to the client.

This scheme allows for the verification of integrity and continuous storage of web archive files by farmers and can prevent outsourcing attacks, working on top of PoUSp, thus not wasting any storage and extending it to build a system that provides distributed storage. Given today's hardware and software optimizations for AES encryption, encoding and decoding can be executed extremely quickly at a negligible cost by the web archive proxy, reducing current storage and operational costs. Our future work aims to allow clients direct access to farmers, further reducing the operational costs of web archives.

5 Preliminary Results

In this section, we will give an overview of the developed work as well as the preliminary prototype that is currently implemented. The main work incurred on developing a stable blockchain thus in this first phase the study of blockchain foundations and space as a nonforgeable resource as in Chia [29] and Filecoin [46] as well as Chia Bread Pudding [5] re-purposing of Chia wasted storage. Verifiable Delay functions (PoT) were also a hot topic. Given this, our practical implementation so far includes:

- Implementation and understanding of the Mathematics behind Wesolowski's Scheme Proof-of-Time and SLOTH.
- Implementation of the Chia Bread Pudding inspired PoUSp scheme with our modifications and specificities for this use case.
- Implementation of a blockchain prototype that can reach consensus based on PoUSp and PoT alternation. The network supports multiple timelords and farmers who always work on the chain with the highest accumulated quality. So far with two timelord and four farmers setup the network stably finalized 3000 blocks maintaining a consistent view in all participants. Only two CPU cores were used consistently and minimal work when blocks were being mined or broadcasted, causing the system to be energy and computationally efficient.
- Using SLOTH with 256 bits security parameter and PoUSp Merkle Tree parameters set at $f = 2, cs = 2048, n = 64$ the system achieves 0.89% useful space and plotting time defined by the SLOTH delay set to match Chia proof generation rate.

6 Forthcoming Work and Conclusions

Currently, our design and implementation focus on achieving blockchain consensus using PoUSp, which embeds useful data within proofs. However, this alone is insufficient to create a distributed storage system capable of outsourcing web archive storage. To address this, our future work will focus on the following, in chronological order:

- Refining and implementing our PDP scheme;
- Developing a web archive server and API that allows clients to request outsourced files;
- Designing an incentive system;
- Evaluating protocol metrics and security and corresponding testing.
- Evaluate web archive cost reduction

In summary, we believe our current blockchain effectively addresses the energy inefficiency of Bitcoin and the storage waste of Chia by implementing a consensus mechanism based on PoUSp and PoT. Our PoUSp scheme builds on the foundation of Chia Bread Pudding implementation, aiming to enhance storage efficiency and incorporate practical considerations for real-world deployment.

By alternating consensus mechanisms between PoUSp and PoT, similar to Chia, our blockchain retains its benefits while re-purposing unused storage. Our research on PDP has laid a strong foundation to transform our stable blockchain into a distributed storage network where farmers are incentivized to store client's data. Once implemented, we believe our PDP scheme will enable the efficient outsourcing of web archive data to blockchain participants, significantly reducing operational and storage costs.

References

[1] Wayback machine. URL https://web.archive.org/. Accessed: 2025-01-03.

[2] Arquivo.pt, . URL https://arquivo.pt/. Accessed: 2025-01-03.

[3] n.d. URL https://www.fccn.pt/. Accessed: 2025-01-03.

[4] O arquivo.pt em números, . URL https://sobre.arquivo.pt/pt/imprensa/o-arquivo-pt-em-numeros/. Accessed: 2025-01-03.

[5] Monica Chen Jin. Chia bread pudding: A blockchain of useful storage. Master of science thesis, Instituto Superior Técnico, University of Lisbon, Lisbon, Portugal, December 2024.

[6] Benjamin Wesolowski. Efficient verifiable delay functions. In *Advances in Cryptology–EUROCRYPT 2019: 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19–23, 2019, Proceedings, Part III 38*, pages 379–407. Springer, 2019.

[7] Leslie Lamport, Robert Shostak, and Marshall Pease. The byzantine generals problem. In *Concurrency: the works of leslie lamport*, pages 203–226. 2019.

[8] Daniel Burkhardt, Maximilian Werling, and Heiner Lasi. Distributed ledger. In *2018 IEEE International Conference on Engineering, Technology and Innovation (ICE/ITMC)*, pages 1–9, 2018. doi: 10.1109/ICE.2018.8436299.

[9] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Satoshi Nakamoto*, 2008.

[10] Elli Androulaki, Artem Barger, Vita Bortnikov, Christian Cachin, Konstantinos Christidis, Angelo De Caro, David Enyeart, Christopher Ferris, Gennady Laventman, Yacov Manevich, et al. Hyperledger fabric: a distributed operating system for permissioned blockchains. In *Proceedings of the thirteenth EuroSys conference*, pages 1–15, 2018.

[11] Richard Gendal Brown. The corda platform: An introduction. *Retrieved*, 27:2018, 2018.

[12] Hai Wang, Yong Wang, Zigang Cao, Zhen Li, and Gang Xiong. An overview of blockchain security analysis. In *Cyber Security: 15th International Annual Conference, CNCERT 2018, Beijing, China, August 14–16, 2018, Revised Selected Papers 15*, pages 55–72. Springer Singapore, 2019.

[13] Juan Garay, Aggelos Kiayias, and Nikos Leonardos. The bitcoin backbone protocol: Analysis and applications. *Journal of the ACM*, 71(4): 1–49, 2024.

[14] Aggelos Kiayias and Giorgos Panagiotakos. Speed-security tradeoffs in blockchain protocols. *Cryptology ePrint Archive*, 2015.

[15] Aggelos Kiayias, Alexander Russell, Bernardo David, and Roman Oliynykov. Ouroboros: A provably secure proof-of-stake blockchain protocol. In *Annual international cryptology conference*, pages 357–388. Springer, 2017.

[16] Rafael Pass, Lior Seeman, and Abhi Shelat. Analysis of the blockchain protocol in asynchronous networks. In *Annual international conference on the theory and applications of cryptographic techniques*, pages 643–673. Springer, 2017.

[17] Cynthia Dwork and Moni Naor. Pricing via processing or combatting junk mail. In *Annual international cryptology conference*, pages 139–147. Springer, 1992.

[18] Adam Back. Hashcash - a denial of service counter-measure, 1997. URL http://www.cypherspace.org/hashcash/. Accessed: 2024-11-29.

[19] Adam Back et al. Hashcash-a denial of service counter-measure. 2002.

[20] Ittay Eyal and Emin Gün Sirer. Majority is not enough: Bitcoin mining is vulnerable. *Communications of the ACM*, 61(7):95–102, 2018.

[21] Jan Werth, Mohammad Hajian Berenjestanaki, Hamid R Barzegar, Nabil El Ioini, and Claus Pahl. A review of blockchain platforms based on the scalability, security and decentralization trilemma. *ICEIS (1)*, pages 146–155, 2023.

[22] Polytechnique Insights. Bitcoin electricity consumption comparable to that of poland, 2024. URL https://www.polytechnique-insights.com/en/columns/energy/bitcoin-electricity-consumption-comparable-to-that-of-poland/. Accessed: 2024-12-02.

[23] Stefan Dziembowski, Sebastian Faust, Vladimir Kolmogorov, and Krzysztof Pietrzak. Proofs of space. In *Annual Cryptology Conference*, pages 585–605. Springer, 2015.

[24] Sunoo Park, Albert Kwon, Georg Fuchsbauer, Peter Gaži, Joël Alwen, and Krzysztof Pietrzak. Spacemint: A cryptocurrency based on proofs of space. In *Financial Cryptography and Data Security: 22nd International Conference, FC 2018, Nieuwpoort, Curaçao, February 26–March 2, 2018, Revised Selected Papers 22*, pages 480–499. Springer, 2018.

[25] Ralph C Merkle. A digital signature based on a conventional encryption function. In *Conference on the theory and application of cryptographic techniques*, pages 369–378. Springer, 1987.

[26] Tal Moran and Ilan Orlov. Simple proofs of space-time and rational proofs of storage. In *Advances in Cryptology–CRYPTO 2019: 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18–22, 2019, Proceedings, Part I 39*, pages 381–409. Springer, 2019.

[27] Martin Hellman. A cryptanalytic time-memory trade-off. *IEEE transactions on Information Theory*, 26(4):401–406, 1980.

[28] Hamza Abusalah, Joël Alwen, Bram Cohen, Danylo Khilko, Krzysztof Pietrzak, and Leonid Reyzin. Beyond hellman's time-memory tradeoffs with applications to proofs of space. In *Advances in Cryptology–ASIACRYPT 2017: 23rd International Conference on the Theory and Applications of Cryptology and Information Security, Hong Kong, China, December 3-7, 2017, Proceedings, Part II 23*, pages 357–379. Springer, 2017.

[29] Bram Cohen and Krzysztof Pietrzak. The chia network blockchain. *White Paper, Chia. net*, 9, 2019.

[30] Filecoin Blog Team. Filecoin and storacha: Spicing up decentralized hot storage like never before. February 2024. URL https://filecoin.io/blog/posts/filecoin-and-storacha-spicing-up-decentralized-hot-storage-like-never-before/. Accessed: 2024-12-11.

[31] David Vorick and Luke Champine. Sia: Simple decentralized storage. *Retrieved May*, 8:2018, 2014.

[32] Giuseppe Ateniese, Randal Burns, Reza Curtmola, Joseph Herring, Lea Kissner, Zachary Peterson, and Dawn Song. Provable data possession at untrusted stores. In *Proceedings of the 14th ACM conference on Computer and communications security*, pages 598–609, 2007.

[33] Ieuan Walker, Chaminda Hewage, and Ambikesh Jayal. Provable data possession (pdp) and proofs of retrievability (por) of current big user data. *SN Computer Science*, 3(1):83, 2022.

[34] Ari Juels and Burton S Kaliski Jr. Pors: Proofs of retrievability for large files. In *Proceedings of the 14th ACM conference on Computer and communications security*, pages 584–597, 2007.

[35] Hovav Shacham and Brent Waters. Compact proofs of retrievability. *Journal of cryptology*, 26(3):442–483, 2013.

[36] Juan Benet, David Dalrymple, and Nicola Greco. Proof of replication. *Protocol Labs, July*, 27:20, 2017.

[37] Chia Network. Proof of time documentation, 2024. URL https://docs.chia.net/proof-of-time/. Accessed: 2024-12-07.

[38] Krzysztof Pietrzak. Simple verifiable delay functions. In *10th innovations in theoretical computer science conference (itcs 2019)*. Schloss-Dagstuhl-Leibniz Zentrum für Informatik, 2019.

[39] Dan Boneh, Joseph Bonneau, Benedikt Bünz, and Ben Fisch. Verifiable delay functions. In *Annual international cryptology conference*, pages 757–788. Springer, 2018.

[40] Dan Boneh, Benedikt Bünz, and Ben Fisch. A survey of two verifiable delay functions. Cryptology ePrint Archive, Paper 2018/712, 2018. URL https://eprint.iacr.org/2018/712.

[41] Arjen K Lenstra and Benjamin Wesolowski. A random zoo: sloth, unicorn, and trx. *Cryptology ePrint Archive*, 2015.

[42] Bram Cohen and Krzysztof Pietrzak. The chia network blockchain. Technical report, Chia Network, July 2019. URL https://www.chia.net/wp-content/uploads/2022/09/Chia-New-Consensus-0.9.pdf.

[43] Chia Network. Chia proof of space construction v1.1, 2022. URL https://www.chia.net/wp-content/uploads/2022/09/Chia_Proof_of_Space_Construction_v1.1.pdf. Accessed: 2024-12-13.

[44] Chia Network Inc. Chia blockchain state dashboard, 2024. URL https://dashboard.chia.net/d/CL1X4UWnk/blockchain-state?orgId=1&from=now-6h&to=now&timezone=browser&var-network=mainnet. Accessed: 2024-12-16.

[45] Ralph C Merkle. Method of providing digital signatures, January 5 1982. US Patent 4,309,569.

[46] Protocol Labs. Filecoin: A decentralized storage network, 2017. URL https://filecoin.io/filecoin.pdf. Version 1.0.0.

[47] Juan Benet. Ipfs-content addressed, versioned, p2p file system. *arXiv preprint arXiv:1407.3561*, 2014.

[48] Rosario Gennaro, Craig Gentry, Bryan Parno, and Mariana Raykova. Quadratic span programs and succinct nizks without pcps. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 626–645. Springer, 2013.

[49] Nir Bitansky, Alessandro Chiesa, and Yuval Ishai. Succinct non-interactive arguments via linear interactive proofs. In *Springer*, 2013.

[50] Eli Ben-Sasson, Alessandro Chiesa, Daniel Genkin, Eran Tromer, and Madars Virza. Snarks for c: Verifying program executions succinctly and in zero knowledge. In *Advances in Cryptology–CRYPTO 2013*, pages 90–108. Springer, 2013.

[51] Protocol Labs. Network performance - filecoin docs, 2024. URL https://docs.filecoin.io/networks/mainnet/network-performance. Accessed: 2024-12-29.

[52] Ben Laurie, Adam Langley, and Emilia Kasper. Certificate Transparency. RFC 6962, June 2013. URL https://www.rfc-editor.org/info/rfc6962.