Chia Bread Pudding

A Blockchain of Useful Storage

Mónica Chen Jin

Thesis to obtain the Master of Science Degree in

Computer Science and Engineering

Supervisors: Prof. Miguel Ângelo Marques de Matos
Prof. João Pedro Faria Mendonça Barreto

Examination Committee

Chairperson: Prof. David Manuel Martins de Matos
Supervisor: Prof. Miguel Ângelo Marques de Matos
Member of the Committee: Prof. Bernardo Luís da Silva Ferreira

November 2023

Declaration
I declare that this document is an original work of my own authorship and that it fulfills all the requirements of the Code of Conduct and Good Practices of the Universidade de Lisboa.

Acknowledgments

First and foremost, I want to express my gratitude to my supervisors for guiding me through this research. To Prof. Miguel Matos, for his expert insights and engaging brainstorming sessions that served as a source of inspiration during the challenging phases of this work. To Prof. João Barreto, for his extensive knowledge and assistance throughout the entirety of this research endeavor. This thesis stands as a testament to their collaborative efforts and unwavering support.

I would like to extend my appreciation to those who have accompanied me during my university studies. To Jay and Cris, for being my trailblazing companions at IST and for the numerous sleepovers and movie nights brimming with unending conversations. To Maggs and Afonso, for the idyll moments shared during academic projects, game nights, and countless malarkey. To Bia, my housemate, for preventing me from getting lost at the hospital and for enduring all the highs and lows of this adventure alongside me.

Last but not least, to the dearest people in my life. To André, for allowing me to become a part of your world, for keeping my sanity intact, and for sharing countless moments of laughter and tears with me. To my family, Mom, Dad, and Francisco, for providing me with everything I needed. Thank you for being my unwavering support system and for all the sacrifices you have made to ensure I receive the best education and the freedom to explore and flourish.

This work was supported by FCT - Fundação para a Ciência e a Tecnologia, through projects UIDB/50021/2020 and Ainur (PTDC/CCI-COM/4485/2021), and developed within the scope of project no. 51 “BLOCKCHAIN.PT - Agenda Descentralizar Portugal com Blockchain”, financed by European funds, namely “Plano de Recuperação e Resiliência - Componente 5: Agendas Mobilizadoras para a Inovação Empresarial”, included in the NextGenerationEU funding program.

Abstract

Permissionless blockchains, such as Bitcoin, are secured using Proof-of-Work (PoW) consensus algorithms. While effective, PoW has a significant energy footprint, leading to the exploration of alternative approaches like Proof-of-Space (PoSp) blockchains. PoSp blockchains offer a promising alternative, relying on disk space rather than computing power to secure the system. Current PoSp techniques generate large quantities of random data for proof generation, leading to initial computational costs and wasted disk space. This thesis introduces Chia Bread Pudding, a protocol designed to employ pre-existing data, such as user files, for proof generation within the framework of PoSp systems. Our protocol demonstrates the capacity to curtail storage wastage in PoSp blockchains by at least 50%, all while upholding system performance. This research empowers greater decentralization by broadening miner participation in the blockchain ecosystem.

Keywords

Blockchain; Proof-of-Space; Proof-of-Storage; Useful Storage.

Resumo

As blockchains sem permissão, como o Bitcoin, utilizam provas de trabalho (em inglês *proof-of-work*, ou PoW) para garantir a segurança do sistema. Embora eficaz, PoW tem uma pegada energética significativa, o que tem levado à investigação de abordagens alternativas como as provas de espaço (em inglês *proof-of-space*, ou PoSp). As blockchains baseadas em PoSp oferecem uma alternativa promissora, baseando-se em espaço de disco em vez do poder de processamento para garantir a segurança do sistema. As técnicas PoSp atuais geram grandes quantidades de dados aleatórios para a geração de prova, resultando num desperdício de espaço em disco. Este trabalho apresenta o Chia Bread Pudding, um protocolo projetado para utilizar dados já existentes, como ficheiros de utilizadores, para a geração de provas no âmbito dos sistemas PoSp. O nosso protocolo demonstra a capacidade de reduzir o desperdício de armazenamento nas blockchains PoSp em pelo menos 50%, mantendo ao mesmo tempo o desempenho do sistema. Esta investigação promove uma maior descentralização ao ampliar a participação de mineiros no sistema da blockchain.

Palavras Chave

Blockchain; Prova de Espaço; Prova de Armazenamento; Espaço Útil.

Contents

List of Figures

List of Tables

List of Algorithms

Acronyms

CBC	Cipher Block Chaining
PDP	Provable Data Possession
PoRep	Proof-of-Replication
PoRet	Proof-of-Retrievability
PoSpT	Proof-of-Spacetime
PoSp	Proof-of-Space
PoSt	Proof-of-Storage
PoS	Proof-of-Stake
PoT	Proof-of-Time
PoW	Proof-of-Work
VC	Vector Commitment
VDD	Verifiable Delay Decoder
VDE	Verifiable Delay Encoder
VDF	Verifiable Delay Function
zk-SNARK	Zero-Knowledge Succinct Non-Interactive Argument of Knowledge
WORM	Write Once Read Many

1 Introduction

Contents

Blockchains are amongst the most popular technologies of recent years. A blockchain is an implementation of a distributed ledger used to record transactions across a network of nodes. It allows multiple parties to reach consensus on a single version of a record, without the need for a central authority. The use of blockchains has the potential to increase security and transparency in a wide range of industries, including finance [2], supply chain management [3], and identity verification [4].

Permissionless blockchains, also known as public blockchains, are a type of blockchain that allows anyone to participate in the system and validate transactions. These blockchains are decentralized and open to anyone, and they do not require permission to join or participate. Examples of permissionless blockchains include Bitcoin [5] and Ethereum [6]. This type of blockchain enables a fully decentralized distributed system, where no single central entity exists, but rather multiple entities work together in a protocol that enables participants to trust the protocol itself, even if they do not trust each other. This means that if any single entity fails, the system can still function as intended. However, this type of blockchain is more susceptible to Sybil attacks [7], in which a malicious actor creates multiple fake identities in order to gain disproportionate influence on the network. If the attacker is able to create enough fake identities, they may be able to gain control of a significant portion of the network and influence the outcome of the consensus process. This can lead to the attacker gaining control of the blockchain and potentially altering existing records.

Prominent blockchains like Bitcoin [8] mitigate the risk of these attacks by using Proof-of-Work (PoW), which requires participants to expend a significant amount of computing power in order to add a new block to the chain. This makes it more difficult for an attacker to create fake identities and gain influence on the system. Unfortunately, this Sybil-proof mechanism comes at the cost of high energy consumption. The energy consumption of PoW blockchains is a source of concern, as it can contribute to greenhouse gas emissions and other environmental impacts [9]. In addition to the environmental impact, the high energy consumption of PoW can also make it more expensive to operate a blockchain. Participants, also known as miners, must invest in powerful computing hardware and pay for the energy required to operate it, which can drive up the cost of participating in the network.

This issue led to the research on more energy-efficient Sybil-proof mechanisms such as Proof-of-Stake (PoS) and Proof-of-Space (PoSp). In a PoS system, the miner of the next block is chosen based on their stake in the network, rather than their computational power. Recently, Ethereum [6] replaced its former PoW protocol with PoS. This decision was made in order to address the high energy consumption [10].

PoSp is another promising approach that relies on disk space, rather than computing power, to ensure the security of the system. In a PoSp system, miners must prove that they have a certain amount of storage space available by allocating a portion of it to the blockchain. For this, existing PoSp techniques rely on the generation of large amounts of random data to build proofs. The Chia Network [1] is a blockchain project based on PoSp that was launched in 2021. Chia has gained significant attention and adoption since its launch, with many people interested in its energy-efficient approach to securing the network. One concern with PoSp is that it may result in a waste of storage space. In order to participate in PoSp blockchains, miners allocate disk space by generating random data with specific cryptographic properties. This not only implies an initial substantial computational cost (which is amortized over the life of the system) but also wastes the space on disk with randomly generated data. Currently, in Chia, all miners provide around 32 billion gigabytes of storage that has no purpose other than securing the blockchain.

One potential way to address this useless space issue is to implement storage-sharing schemes, in which multiple users can share the same storage space and participate in the consensus process. This could help to reduce the amount of unused storage space and make the use of PoSp more efficient. Recent proposals such as Filecoin [11] and Subspace [12] address this useless space issue by repurposing the storage allocated with random data used to build the PoSp proofs. For this, they use

Proof-of-Storage (PoSt) [13–17] to generate proofs. PoSt allows a process to prove that it is correctly storing a piece of data previously provided. Filecoin [11] reuses this space, allowing clients to store useful data there. Filecoin is a blockchain that implements a decentralized storage network where clients can outsource the storage of their data to miners. These miners participate in the blockchain by providing PoSt proofs that they are storing clients' data. Subspace [12] proposes to repurpose this space by archiving the blockchain's history. Subspace is a blockchain where miners can participate by storing the blockchain's history and providing PoSt proofs of it. These previous approaches find ways to provide data to miners and secure its storage and retrieval. However, these additional requirements increase the complexity of their systems, which impacts their network performance. There are a number of limitations of the current state of the art in PoSt. PoSt blockchains are still a relatively new development and have not yet seen widespread adoption. As a result, it is difficult to accurately assess their long-term viability and effectiveness in reducing space waste. There is a lack of data and experience with these blockchains, which makes it difficult to understand their full potential and limitations. It is not yet clear how successful PoSt blockchains will be in the long term, and the trade-offs between performance and security for these protocols have not yet been thoroughly studied.

Our work focuses on a simpler way to reduce the waste of disk space of PoSp blockchains without the data storage and retrieval requirements of current proposals.

1.1 Objectives

The goal of this dissertation is to research techniques and methods that can preserve the guarantees of PoSp but take advantage of data the user already has (such as user files) for proof generation, resulting in a Proof-of-Useful-space system that reduces the waste of disk space inherent to existing approaches.

In particular, we aim to address the following research questions:

RQ1: To what extent can we replace Proof-of-Space blockchain's random data with useful data?

RQ2: What is the impact of repurposing the data used in Proof-of-Space blockchains?

To achieve this goal, we propose Chia Bread Pudding, a novel protocol that uses local preexisting data for the proof generation of PoSp blockchains. Chia Bread Pudding aims to change the way miners demonstrate their commitment of storage space for the blockchain. We propose using an encoding method on the miners' existing useful data, which will enable them to compute the required proof. This approach is built and tested by modifying Chia's PoSp protocol with a PoSt protocol. Our results show that this method allows us to reduce the storage waste of Chia by at least 50% without having a significant impact on the blockchain's overall performance.

The work described in this thesis was partially presented in the following peer-reviewed publication:

M. Jin, M. Matos and J. Barreto. ”Pudim de Pão e Chia: uma blockchain de espaço útil”, in Atas do décimo quarto Simpósio de Informática (Inforum), Portugal, September 2023.

1.2 Outline

This thesis is organized as follows: Chapter 2 presents background and related work about the main concepts relevant to our work. Chapter 3 describes our solution proposal. Chapter 4 outlines the evaluation methodologies employed and presents the results obtained. Finally, Section Chapter 5 concludes this dissertation.

2 Background and Related Work

Contents

In this chapter, we present some fundamental background on blockchains and consensus mechanisms and summarize the related work that has been done so far in the areas of blockchains with useful work consensus. We first describe distributed ledgers and the properties that they must ensure. Then we enumerate the decentralized consensus mechanisms that are used in permissionless blockchains. Lastly, we describe the concept of useful work in the context of consensus and its current applications.

2.1 Distributed Ledgers and Blockchains

We consider a ledger to be an ordered list of transactions. The current state of a system (e.g. the balance of users) can be derived through the entire history of transactions stored in the ledger. The ledger can be modified by appending new transactions to the end of the ledger. These transactions can consist of any relevant transfer of information. For example, in the case of cryptocurrencies (digital currencies), a transaction is a transfer of currency value.

A distributed ledger is a ledger spread across multiple nodes on a peer-to-peer network, where each node stores a replica and agrees on a consistent view of the ledger. However, achieving consensus in a decentralized network is not trivial. These properties of distributed ledgers allow their usage in various sectors, including decentralized currencies and smart contracts. We now focus on blockchains, an implementation of a distributed ledger.

Blockchains were proposed by Satoshi Nakamoto for a decentralized digital currency named Bitcoin [8]. A blockchain consists of a list of blocks that are linked together using cryptographic functions. Each block holds important data, such as the ledger's transactions and a reference to its previous block, which is computed by using a cryptographic hash function on the previous block's data. The state of a blockchain, or simply chain, can be derived by traversing the linked list of blocks. Following a consensus protocol, new blocks can be added to the chain in an append-only fashion. A consensus protocol is a fault-tolerant mechanism used in a system to achieve agreement upon a consistent view of the network. Having nodes agree on a consistent view of the network is crucial to its security. For example, consensus prevents double-spending in cryptocurrencies, i.e., when the same currency unit is used more than once. In blockchain's setting, a consensus protocol dictates which blocks are to be added at the end of the chain. Each block's reference to its predecessor ensures that the history of transactions is not tampered with since altering one block will change its hash, invalidating all of its successor references. This scheme makes it difficult for records held on the blockchain to be altered or deleted, which explains its popular usage in cryptocurrencies.

Blockchains can be classified as **permissioned** or **permissionless**.

Permissioned blockchains are limited to designated participants that were previously approved by a trusted party. This constraint means that the system knows all its participants, enabling the use of classical consensus protocols such as PBFT [18]. This category provides higher throughput because the transactions are stable the moment they are added to a participant's chain. However, having to trust an entity to manage memberships limits this blockchain's fault tolerance.

Permissionless blockchains allow anyone to participate in the system at any time. In this work, we focus on this type of blockchain, where trust is primarily established through the protocol. The flexibility of permissionless blockchains leads to issues such as Sybil attacks, where anyone can create an arbitrary number of identities that can compromise quorum-based solutions. Sybil attacks are a threat to blockchain security since they make it harder for the network to reach consensus. If the blockchain's protocol is not secure, malicious users can perform a Sybil attack by impersonating multiple entities in order to gain advantage of quorum-based consensus protocols. Sybil attacks can be prevented by requiring each participant to spend a non-counterfeitable resource, discouraging the impersonation of multiple entities. Furthermore, as the system does not have any prior information about its participants, classic consensus protocols can not be used to maintain the distributed ledger.

Bitcoin [5] is the first cryptocurrency to implement a permissionless blockchain. This protocol laid the foundation for the protocols of today's decentralized cryptocurrencies. By analyzing Bitcoin's core protocol Gary et al. [8] identified two core properties of a robust public transaction ledger. The authors also imply that Bitcoin only ensures these properties under some additional assumptions described in Section 2.2. We informally state these properties below.

Persistence: once an honest player reports a transaction deep enough in the ledger, all other honest nodes will report that transaction whenever asked and at exactly the same position in the ledger. This property is essential to ensure that a transaction is stable after a specific time.

Liveness: if the transaction comes from an honest node and is sent to all honest nodes of the network, then, after a certain period of time, this transaction will be included in the ledgers of all honest nodes.

Gary et al. [8] and Kiayias et al. [19] also show that the previous properties are ensured if the honest nodes hold the majority of the non-counterfeitable resource, and the blockchain has the following properties:

Common prefix: if an honest participant discards k blocks from the end of its local chain, the probability that the resulting chain will not be a prefix of another honest party's chain drops exponentially in the security parameter (difficulty of the cryptographic puzzle). This property is related to the persistence property since it ensures that honest participants have a consistent past amongst their ledgers.

Chain quality: the ratio of blocks proposed by honest participants in an honest participants' chain is no less than a threshold. A blockchain maintained by an honest participant is guaranteed, with high probability, to have some blocks created by honest participants.

Chain growth: given a growth rate τ, after r consecutive rounds, it is highly probable that the blockchain of any honest participant has grown at least $\tau \cdot r$. This property is linked to the liveness property and enables one to abstract the blockchain feature of being able to grow unhindered by dishonest participants.

2.2 Proof-of-Work Blockchains

As mentioned in Section 2.1, blockchains are an implementation of a distributed ledger suitable for cryptocurrencies due to their decentralization and tamperproof properties. However, the data structure of blockchains does provide not, by itself, the security needed for a distributed ledger. Distributed ledgers require all replicas to agree on a consistent view of the ledger, and the history of old transactions should not be changed. To ensure these properties, blockchains use a consensus protocol that dictates which node can add a new block to the chain. For this consensus protocol to be secure, it needs to have a Sybil-proof mechanism, where all participants are required to spend a non-counterfeitable resource. In this section, we focus on blockchains that use computational power as the non-counterfeitable resource. The expenditure of this resource is proven using PoW.

PoW has been proposed for multiple security goals, including protection against spamming and other denial-of-service attacks [20, 21]. PoW, as described by Jacobsson et al. [22], is a protocol in which one party solves a puzzle that demonstrates to others that a computational effort was expended. This expenditure can be verified by any party with minimal effort.

We now describe the first successful PoW blockchain protocol, Bitcoin, which laid the foundation for other blockchains of this type. Bitcoin [5] is a cryptocurrency that consists of the blockchain technology described above.

Bitcoin uses a consensus protocol known as Nakamoto consensus. The Nakamoto consensus is based on a PoW mechanism and a chain selection rule. First, any participant who wants to add a new block to the blockchain needs to perform a cryptographic proof known as PoW. In Nakamoto consensus, PoW serves as a Sybil-proof mechanism because the voting power of a node is proportional to the computational power spent on the cryptographic puzzle, which cannot be counterfeit without breaking the cryptographic function of the puzzle.

The Bitcoin implementation of PoW is based on Hashcash [23]. Its cryptographic puzzle consists of scanning for a nonce n such that $H(h, b, n) < D$, where H is a hash function (e.g. SHA-256) h is the hash of the previous block, b is the current block that includes a set of transactions, and D is the difficulty parameter. The average work required to solve this puzzle is exponential in the number of zero bits with which the difficulty parameter begins. This parameter is periodically tuned (every 2016 blocks) to keep the average block generation time close to 10 minutes.

In Bitcoin, a miner who successfully adds a new block to the blockchain creates a coin that is owned by that miner. The participant can also charge fees for including transactions in this block. This adds an incentive for participants to support the network and provides a way to initially distribute coins.

When participants receive a new block with a valid PoW, they add it to their local blockchain and try to extend this updated blockchain. The puzzle can have 1) multiple solutions and 2) more than one participant solving it. Due to this, multiple blocks can be added at the same chain height. Such event, where two different valid chains are proposed, is called a fork.

Second, to choose between multiple chains, participants use the chain selection rule. This rule requires participants to choose the chain with the biggest expenditure of computation, i.e. the sum of all the difficulties of each block of a chain.

Nakamoto [5] concludes that if honest nodes hold the majority of the CPU power, the honest chain will grow faster than any competing chain, making it difficult for an attacker to perform a double spending attack as they would need to 1) redo the PoW of the previous block to modify, 2) redo the PoW of the successors of the block to modify and 3) surpass the work of all honest nodes. Gary et al. [8] prove that the majority assumption is not enough to ensure the properties of a roubust public ledger. In their analysis [8], Bitcoin provides a robust public ledger under the following assumptions: 1) the adversary controls less than half of the total hashing power, 2) the network synchronizes much faster relative to the PoW solution rate, and 3) digital signatures cannot be forged. Furthermore, Eyal et al. [24] also demonstrate Nakamoto's conclusion is not correct due to subtle mining attacks. They demonstrate that mining pools,i.e. groups of participants that contribute to the same puzzle solution and share the rewards proportionally to their contributions, can mine on a private chain that is only revealed when its length surpasses the honest chain. Once this longer chain is revealed, some honest transactions on the honest chain will be discarded due to the chain selection rule, leading to a waste of honest resources. Eyal et al. argue that honest nodes will be incentivized to join these pools to avoid wasteful work.

Blockchains based on Nakamoto consensus have several limitations. Transaction confirmation times are long (in the case of Bitcoin 1 to 1.5 hours) since this type of blockchain only considers transactions as stable (i.e., they will prevail in the ledger) after a certain number of successor blocks are added. Transaction throughput is limited, for instance, Bitcoin has a throughput of 7 transactions per second, due to the block size limit of 1MB.

Furthermore, another concerning drawback of the Nakamoto consensus is the consumption of vast amounts of energy to maintain the ledger [25, 26]. At the time of writing, the estimated electrical energy to maintain the cryptocurrency is 111.18 TWh. This demand is comparable to the energy consumption of entire countries such as the Netherlands [27]. This energy is solely used to manage and secure the blockchain, prompting widespread concern about waste. Bitcoin maintainers address this issue by considering that "Spending energy to secure and operate a payment system is hardly a waste. Like any other payment service, the use of Bitcoin entails processing costs. Services necessary for the operation of currently widespread monetary systems, such as banks, credit cards, and armored vehicles, also use a lot of energy[1]".

Alternatives to this consensus mechanism have been proposed to address these issues. They mostly follow one of three different approaches: 1) improving the protocol whilst still using PoW; 2) using a clas-

[1] Accessed 25 Nov. 2022 at https://bitcoin.org/en/faq#isnt-bitcoin-mining-a-waste-of-energy

sical byzantine consensus; 3) replacing the Sybil-proof mechanism using another non-counterfeitable resource.

We focus on the latter approach which exploits the usage of a different kind of resource instead of computational power.

2.3 Proof-of-Stake Blockchains

In this section, we focus on PoS Blockchains.

PoS proves stake ownership, that is, this protocol proves how much currency a participant holds in the system. The design of PoS protocols was first initiated in online forums and subsequently considered by the academic community [28–32]. PoS is used as a Sybil-proof mechanism that uses the currency in the system as a resource. The probability that a stakeholder is selected to add the next block is proportional to their wealth in the system. This approach has the advantage of consuming less energy because the mining process does not require intensive computation. However, PoS blockchains are at risk for costless simulation (also known as nothing-at-stake) and long range attacks [32].

Costless simulation attacks consist of malicious miners computing many more proofs than specified by the protocol due to the deliberately cheap mining process. This is a problem because miners can compute proofs for multiple blocks and check which one has a higher chance of being included in the chain. For example, if the proofs depend on the content of blocks miners can change the order of transactions and generate different proofs. This also compromises the dynamic of the blockchain, since one of the purposes of this Sybil-proof mechanism is energy efficiency. Furthermore, an attacker can also try to extend multiple chains to persist their block in the valid chain. In the presence of a fork (i.e., two alternative chains), rational miners would add new blocks to both of the chains to maximize their expected reward. This slows down consensus and can make the blockchain never converge (i.e., nodes might never agree on a valid chain).

Long-range attacks[2] compromise the ledger's integrity. In other words, the history of previous transactions is not secured. In these attacks, a group of nodes can rewrite the blockchain history in their favor if they hold the majority of the currency in the system. When a group of nodes has such a high stake in the system, they are capable of creating a new chain that surpasses the honest one, directing the state of the blockchain to their advantage. This is also a problem with PoW blockchains, but this threat is aggravated in PoS blockchains due to the existence of costless simulation.

A variety of PoS-based blockchain protocols have been proposed, with different properties and different ways to solve the aforementioned problems such as Snow White [33], Ouroboros [34] and Algorand [35]. For example, Snow White discourages costless simulation attacks by reducing the reward of

[2]These attacks are a specialization of costless simulation attacks, but we state them separately due to their unique purpose.

miners who sign multiple blocks for the same depth, and periodically creates checkpoints that show the state of the system. Discrepancies between the data stored in these checkpoints and the chain indicate long-range attacks.

One of the most popular cryptocurrencies, Ethereum [6], fully replaced PoW with PoS in September 2022. Ethereum maintainers support this decision with the energy efficiency of this Sybil-proof mechanism. According to Ethereum's analysis[3], this transition lead to a reduction in energy consumption for the maintenance of the blockchain of around 99%. This indicates a trend of cryptocurrencies toward energy-efficient protocols, while also acknowledging the waste of resources solely to maintain a blockchain.

One of the biggest drawbacks of blockchains that rely on PoS is the concern of the rich-get-richer issue. This issue is centered around the argument of whether rich miners (those who possess more stakes) will obtain more rewards and further increase their income in the future. Huang et al. [36] analyze the fairness of PoS incentive protocols compared to PoW protocols. They conclude that only Ethereum is capable of achieving the same levels of fairness as PoW models under specific settings.

2.4 Proof-of-Space Blockchains

PoSp is a technique to prove that a certain amount of memory or disk space was allocated. Dziembowski et al. [37] first introduce this protocol as a more ecological alternative to PoW.

PoSp is a protocol between a prover P and a verifier V that has two distinct phases: initialization and proof execution. During the initialization phase, P stores some data and V stores a small piece of information. After the initialization, comes the proof execution phase. This phase starts whenever V challenges P. The latter needs to prove that it is allocating disk space using the data that were stored during initialization. The phase ends with V either accepting or rejecting this proof. There are two significant approaches to this type of proof.

The first approach [37] takes advantage of hard-to-pebble graphs and Merkle hash trees[4]. During initialization, V sends the description of a hash function to P, who then labels each vertex of a hard-to-pebble graph[5] by hashing the labels of its predecessor vertices. P then generates a Merkle tree using the previously computed labels as leaves and sends the root value of this tree to V. In the proof execution phase, V simply asks P to open the labels corresponding to some randomly chosen nodes. This is done by having V send a challenge consisting of a leaf value c of the Merkle tree to P requesting him to provide the values of the sibling nodes that lie on the path from c to the root. V can then compute the

[3]Accessed 2 Dec. 2022 at https://ethereum.org/en/energy-consumption/

[4]Tree structure in which each leaf node is a hash of a block of data, and each non-leaf node is a hash of its children.

[5]A directed acyclic graph where the labels of a vertex are the hash value of its parent vertices. A vertex can only be labeled if all its parent vertices are also labeled.

root of the Merkle tree using these values and compare it to the root that was committed by P during the initialization phase. If these two roots match, V accepts the proof provided by P; otherwise, V rejects it.

Dziembowski et al. [37] prove that an attacker uses $\Omega(N)$ space to store proofs or makes $\Omega(N)$ invocations to the hash function. Hence, a rational user will always choose the first option since it is cheaper and faster. This variation of PoSp provides the best security guarantees for this type of proof but it is hard to put into practice due to its complex construction and this kind of proof cannot be made fully non-interactive when used in blockchains (i.e., participants are not able to join anytime without any preparation).

Abusalah et al. [38] propose a second approach to PoSp. This approach is based on the inversion of random functions. Dziembowski et al. [37] raise a naive version of this approach, but it does not provide any meaningful security guarantees due to time-memory trade-offs to invert random functions. Hellman [39] showed that any function over a domain of size N can be inverted with constant probability in time $\Theta(N^{2/3})$ using $\Theta(N^{2/3})$ storage.

Abusalah et al. show that, in the context of PoSp, the previously mentioned time-memory trade-offs for inverting functions can be overcome if one relaxes the requirement that the function is efficiently computable and just asks for the function table to be computed in (quasi)linear time. The basic construction for this is built using a random function oracle $g : [N] \times [N] \rightarrow [N]$ and a random permutation oracle $f : [N] \rightarrow [N]$. The random function that serves as the proof is $h(x) = g(x, x')$, such that, $f(x) = \pi(f(x'))$ with π being any involution without a fixed point[6]. If f is a function instead of a permutation, the involution can be left out, i.e. the condition becomes $f(x) = f(x')$. We describe the PoSp protocol for f as a random function. During the initialization phase, verifier V sends to the prover P the description of $h : [N] \rightarrow [N]$. P computes the function table of h, that is, stores a table with (x, x') and its corresponding $g(x, x')$. During the proof execution phase, V challenges P by sending a random $y \in [N]$ and. If P replies to V with a tuple (x, x') such that $f(x) = f(x')$ and $y = g(x, x')$, V accepts, otherwise, V rejects.

2.4.1 SpaceMint

Adapting PoSp to enable its use in a blockchain is not trivial. The mining process consists of consulting the proofs generated during the initialization step and fetching the one that corresponds to the challenge provided by the last block of the chain. The deliberately cheap mining process of PoSp blockchains also suffers from costless simulation and long-range attacks (described in Section 2.3). Park et al. propose SpaceMint [40] as the first cryptocurrency with a blockchain using PoSp in its consensus protocol. They identify the following list of challenges when replacing PoW with PoSp.

Interactivity: Permissionless distributed ledgers allow participants to join and start mining a block at

[6] Function that is equal to its inverse, e.g. flipping all the bits

any time without requiring any prior preparation. When SpaceMint was published, the only PoSp known was interactive [37], since it involved an initial commitment before the mining process. SpaceMint solves this problem with two techniques. First, it uses the Fiat-Shamir paradigm, which is a standard technique for replacing a public-coin challenge with a hash of the previous message, already used to adapt PoW for Bitcoin. Second, it creates a new type of transaction to record the commitment to its space that a prover needs to send to the verifier in the initialization phase of PoSp.

Determine the winner: In Bitcoin the probability of a miner successfully extending a new block is proportional to its hashing power. In a proof system like PoSp, where the proof is deliberately easy to compute, there will be multiple participants eligible to extend the chain. However, the protocol needs to specify who ”wins” and when to continue with the next block. The winning probability of a miner should be proportional to the space it dedicates. For this, SpaceMint defines a quality function, which quantifies the quality of each PoSp proof. This function can be computed locally and it is designed such that the probability of a miner having the highest quality proof is proportional to the space that was previously dedicated.

Costless simulation: When replacing PoW with proofs that are computationally easy to generate such as PoSp, blockchains become vulnerable to costless simulation attacks. The intensive computation performed during the Bitcoin [5] mining process ensures that all participants are incentivized to extend a single chain, leading to consensus. However, by using PoSp, it becomes computationally cheap to extend multiple blocks with a single proof by slightly altering the contents of the block (e.g., by using a different order of transactions) before choosing the most favorable one to announce. This slows down consensus and is a potential security issue. To prevent these attacks, SpaceMint splits the chain by having one chain store the proofs and another chain store the transactions. This prevents attackers from cherry-picking transactions in order to generate the most promising proof. In order to disincentivize the extension of multiple chains, SpaceMint resorts to penalties that punish participants that try to mine on more than one block. Whenever a participant identifies this type of attack, half of the block's reward is destroyed, and the other half is given to the accuser.

Long-range attacks: Another critical issue that arises with cheap proof computation is the potential double-spending by generating a chain that surpasses the honest chain. Recall that in Bitcoin [5] participants choose the chain that required the most hashing power to generate. The chain selection rule can be adapted for PoSp by choosing the chain that reflects the most space, i.e. the one that has the highest sum of PoSp qualities. This approach is delicate because the content of the previous blocks influences the PoSp qualities of the future blocks. A malicious miner can craft initial blocks with low quality proofs so that later in the sequence he can create proofs with high quality. This makes it possible to generate a new chain with higher quality than the honestly generated one but with less amount of space dedicated to mining. SpaceMint addresses this issue by placing more weight on recent blocks when calculating the quality of the chain.

SpaceMint's system loses one of the nice properties of permissionless blockchains: the node's ability to join the mining protocol just by listening to the network. In SpaceMint, miners who want to start mining are required to send out a commitment transaction and other miners need to include this transaction in the blockchain. This means that if all miners stop accepting transactions, new nodes cannot join the protocol.

Lastly, all the data stored in memory or disk is solely used to solve a puzzle that verifies that a certain amount of space was indeed allocated. These data have no other utility other than to secure the blockchain.

To the best of our knowledge, SpaceMint was never deployed.

2.4.2 Chia

Chia [1] is another cryptocurrency that uses PoSp in its consensus mechanism. It was launched successfully in 2021. Chia proposes a solution to the aforementioned drawbacks of SpaceMint [40] and deploys a more practical protocol using PoSp that has a mining dynamic similar to Bitcoin. This protocol uses PoSp in conjunction with Proof-of-Time (PoT). Chia uses the second approach of PoSp proposed by Abusal et al. [38] (Section 2.4).

Although Chia uses a different PoSp protocol from SpaceMint which solves the interactivity problem identified by Park et al. [40], it still suffers from the rest of the problems.

Determine the winner: Just like SpaceMint, Chia also assigns a quality to each PoSp proof and the protocol specifies that the proof of the best quality announced should be considered as the extension of the chain. This quality function measures how close each PoSp value is to the challenge value of the block that the miners are trying to extend. The closer a PoSp value is to the challenge value, the higher the quality. The winning chain is the one with the highest accumulated quality. The probability of a miner successfully extending the chain is proportional to the space allocated for PoSp. We can look at the winner selection process as a lottery game. The initial allocation performed by each miner fills the space with lottery tickets. The block the miner intends to extend has a random value, and the closer a lottery ticket number is to this selected value, the higher the chance of the miner winning the lottery (i.e., extending the block). Therefore, a miner can increase their chance of winning by storing more lottery tickets (i.e., allocating more data).

Costless simulation: Chia keeps SpaceMint's idea of a chain split, decoupling the transactions and the proofs into two separate chains. However, instead of using a policy-based solution to discourage attackers from mining multiple blocks, Chia resorts to a technical solution. Chia mitigates this issue by making this attack part of the protocol, i.e., nodes mine on top of the first k blocks they see at each height. This attack becomes less profitable as k increases. The drawback of increasing k is the time it

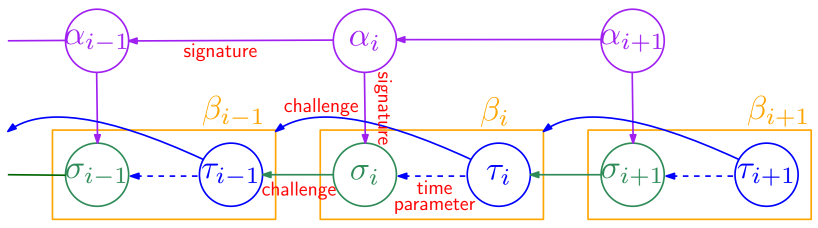

Figure 2.1: Illustration of the Chia blockchain,from [1]. Each block's data α is decoupled from the proofs (PoSp σ and PoT τ).

takes to achieve consensus.

Long-range attacks: Recall that another issue related to PoSp is that an attacker can rewrite the chain's history by privately growing a chain that surpasses the honest chain. SpaceMint mitigates this attack by using a chain weight function that gives more weight to recent block qualities. Chia addresses this by alternating PoSp with PoT. PoT shows that a certain amount of time was spent using VDF [41,42]. A VDF is a non-parallelizable function that takes a predetermined amount of time to compute. For every input to the function, there is only one output and it can be quickly verified. Since every block needs to spend some time to complete, the simulation will not be free. Every block in Chia is composed of a PoSp followed by a PoT. The delay of a VDF is inversely proportional to the quality of the PoSp. PoT is similar to PoW in the sense that they both slow down the block creation and make it harder for attackers to bootstrap a heavy chain in a short amount of time. However, VDFs achieve these properties without wasting massive computing power as required by PoW.

Because our solution is based on Chia, we describe the protocol in more detail next. In Chia, there are two types of miners: PoSp miners, referred to as Farmers, and PoT miners, referred to as Timelords. Every completed block has a PoSp θ followed by a PoT τ, as shown in Figure 2.1.

A Farmer first dedicates an amount of space in a process called plotting. Only then can it begin to mine new blocks. The plotting process stores PoSp proofs in the dedicated space. The PoSp proof construction is based on the basic construction of Abusalah et al.'s [38] approach, but it is nested 6 times and contains other heuristics to make it more practical.

After the plotting process, the Farmer can start mining. To extend a chain, a Farmer first computes the challenge by hashing the last VDFs output. The Farmer then searches their plot by searching for a proof that is closest to this challenge value. Once a good proof is found, the Farmer extends its local chain with this proof and broadcasts it to the network.

Then, a Timelord picks up this non-finalized block (i.e., block with only a PoSp proof) and computes its VDF using the PoSp proof. After completing the VDF execution, the Timelord finalizes the block by extending it with a PoT proof.

Once a block is finalized, other Farmers can try and extend it. The blockchain is extended by alternating PoSp and PoT proofs.

Chia [1] analyzes the security of its blockchain and concludes that at least 73.1% of the nodes should be honest when k = 1 (recall that k is the number of parallel chains), and 61.5% if k = 3 (the current k in production). Chia requires a higher percentage of honest nodes than other PoW blockchains such as Bitcoin, which requires 50%.

Another drawback of Chia is the fact that VDFs are still a recent topic of investigation and require carefully selected parameters when used in blockchain environments. Thus, VDFs may need further study to be fully trusted in a blockchain such as Chia.

Although Chia provides a less energy-consuming blockchain compared to Bitcoin, it still has some waste associated with it. Just like SpaceMint, the waste in question is related to the proof initialization step. During this step, computation is used to generate data with specific cryptographic properties which fills the disk space. These data can only be used to potentially generate PoSp proofs. This introduces computational and storage waste. In a successful network like Chia's, there are 28 exbibytes (32 billion gigabytes) of space solely serving to maintain the blockchain.

2.5 Proofs-of-Storage Blockchains

PoSt is a PoSp scheme, but instead of showing that space is allocated with seemingly random data, it shows that the allocated space is correctly storing a piece of data previously provided. In a PoSt scheme, a verifier V outsources the storage of data D to a prover P and then repeatedly checks if P is still storing D.

There are three types of PoSt: 1) Provable Data Possession (PDP); 2) Proof-of-Retrievability (PoRet); 3) Proof-of-Replication (PoRep).

Provable Data Possession [13] allows a verifier V that has outsourced data storage to an untrusted prover P to verify that P possesses the original data without retrieving them.

Proof-of-Retrievability [14–16] is similar to PDP, but enables the extraction of the data. In this scheme, a verifier V that has outsourced the storage of data D to an untrusted prover P can check if P is still storing D and retrieve it. This is accomplished by making the proofs themselves leak pieces of D so that V can verify the possession of D a number of times and then reconstruct D from the proofs.

Proof-of-Replication [17] is similar to the previous types of PoSt, but additionally ensures that P is dedicating a unique physical space to store D. In other words, attackers can not pretend to store D twice by deduplicating the storage. All storage providers store each replica of D independently.

PoSt has great potential for archival purposes because these schemes allow a client to store a file on an untrusted server and verify the file's integrity. PoSt only proves that data is stored at the moment the proof is generated. However, clients typically outsource data storage for a period of time. Thus, PoSt schemes are frequently paired with Proof-of-Spacetime (PoSpT) schemes.

Proof-of-Spacetime [43] allows the prover P to convince the verifier V that P has allocated some space over a period of time. This scheme does not prove the storage of specific data provided by V. Instead, it proves that P spent some disk space over a period of time.

PoSt blockchains are popular in Decentralized Storage Networks, since they allow users to share and store data without relying on a third-party storage provider. When PoSt schemes are used on a permissionless blockchain as a Sybil-proof mechanism, they provide an additional property to its consensus protocol known as usefulness. The work done by the miners in a consensus protocol is considered useful if the outcome of the computation is valuable to the network beyond securing the blockchain. This useful work idea is described in the early work of PoW due to the concern about the energy waste of the protocol.

The first idea of repurposing this energy for some useful purpose was sketched by Dwork and Naor in their use of PoW for the antispamming technique [20]. They write that ”[One] possible scenario would be that in order to send a user a letter, some computation that is useful to the recipient must be done. We currently have no candidates for such useful computation”.

Jakobsson and Juels introduce the idea of bread pudding protocol [22]. Bread pudding protocol is a scheme that reuses PoW such that the computational effort invested in the proof may be reused by the verifier to achieve a separate, useful, and verifiably correct computation. As an example of a bread pudding protocol, Jakobsson and Juels consider the highly computationally intensive operation of minting in the MicroMint scheme [44]. MicroMint is a micropayment system developed by Rivest and Shamir where its security is based on the hardness of finding collisions of the hash function. The proposed example of a bread pudding protocol is how the computationally intensive task of minting MicroMint can be partitioned into a collection of PoWs, enabling minting to be distributed among a collection of untrusted entities. These PoWs can not only serve in their own right for security protocols but can also be harvested in order to outsource MicroMinting process to a large group of devices.

Later works [45] also propose bread pudding protocols by repurposing PoW computation for something useful (e.g., Primecoin [46] is a cryptocurrency that reuses miners’ computational power to find new prime numbers). Furthermore, other schemes such as PoSt blockchains [11, 12] also focus on repurposing the disk space that is filled with random data in PoSp.

2.5.1 Permacoin

Permacoin [47] is a cryptocurrency that also serves as archival storage using PoRet with a novel variant of PoW. Miller et al. present Permacoin as a way to repurpose the energy wasted in Bitcoin’s mining process [5] in order to achieve a useful goal: distributed storage of archival data.

For this, the PoW scheme used in Permacoin requires miners to generate PoRet proofs of a file in order to compute PoW. Thus, Permacoin not only requires miners to spend computation power but also storage space.

Permacoin network stores a large, publicly valuable digital archive by having each miner store a fragment of it in order to participate in the mining process. It is assumed that a file F is already processed into n segments by erasure coding[7] and a Merkle root computed using all the n segments is known to all participants. This Merkle root is later used to check the validity of each PoRet proof provided in the blocks.

Each participant initially generates a public-private key pair (pk, sk), chooses the number of segments l it wants to store, and fetches those many segments. The specific set S of segments is determined by the hash of its pk. In other words, $S = \forall_{i \in [l]} : H(pk||i) \mod n$. Then, each participant generates the PoRet proof for each of these segments, which consists of the segment i and its respective Merkle proof π_i. All the segments and their respective proofs are stored on disk.

During the mining process, each tentative of a PoW proof includes a set of segments and its respective PoRet proof. For this, the miner is required to compute the hash of the segment and other information and then sign the resulting hash with their private key. The next segment of the subset is chosen based on the node's private key and the signature to ensure it cannot be pre-fetched. By tying the PoW solution to the private key, Permacoin discourages the miners from outsourcing the storage of segments and proofs to remote storage (e.g., the cloud). Furthermore, since the selection of each segment is sequential and random, the set of segments cannot be pre-computed ahead of time. Thus, if the data are stored remotely and the computation is performed locally, many round trips must be incurred during each PoW attempt, reducing the miner's chance of successfully extending the blockchain. The alternative would be to reveal their private key to the cloud, but since the key controls their payments it is unlikely to be revealed.

In Permacoin, nodes must store their segments of the dataset correctly in order to have any chance of a reward. They verify their own PoRet proofs in order to ensure they gain a reward and thus have their proofs publicly verified.

Permacoin is still fundamentally a PoW-based scheme. The computational power used in the mining process of Permacoin is similar to Bitcoin's and, to the best of our knowledge, it has not been deployed.

2.5.2 Filecoin

Filecoin [11] is a Decentralized Storage Network that turns cloud storage into an algorithmic market. This market runs on its native token FIL, which miners earn by providing storage to clients. Filecoin's decentralized storage network relies on a PoRep and PoSpT blockchain that follows a useful work con-

[7]erasure coding turns a dataset of f blocks into rf $(r > 1)$ segments such that the dataset can be recovered from any f segments

sensus.

Clients pay to store data and retrieve data, and miners provide storage to the Decentralized Storage Network. Before offering disk space, miners must pledge their storage by depositing a proportional amount of collateral. Miners can fill this pledged space with clients' requests and receive their payments. Whenever they fail to prove the client's data is being stored, they lose part of their deposited collateral. Miners can also mine new blocks, and in doing so, they receive the mining reward for creating a block and the transaction fees for the transactions included in the block.

Filecoin's demand and supply of storage meet in verifiable exchange markets. Clients and miners set prices for the services they request or provide by submitting to these markets. All of these market exchanges are submitted to the blockchain, and any participant can verify them.

The storage offered by a miner is divided into sectors that are filled with clients' data. To compute a PoRep proof, the miner first needs to seal the filled sector, i.e. unit of storage defined by the system. Sealing is a slow, sequential operation that transforms the data of a sector into a unique physical copy by generating a pseudo-random permutation of the data unique to the miner's public key. This PoRep proof is committed to the blockchain and ensures that the Storage Miner has a physically unique copy of data. To ensure that they are continuing to dedicate storage space to that same data over time, Storage Miners are required to repeatedly commit to the blockchain PoSpT proofs. These proofs are computed by generating PoRep proofs in sequence, using the output of a proof as an input of the other for a specified amount of iterations t (time parameter). If miners fail a PoSpT proof at any point during the contract, part of their collateral is lost. In order to reduce the size of the proofs, they are compressed using Zero-Knowledge Succinct Non-Interactive Arguments of Knowledge (zk-SNARKs) [48–50]. zk-SNARK proves that a proof has been done correctly without having to reveal the details of the proof itself or the underlying data on which it is based.

Benet et al. [11] describe Filecoin's consensus as a Useful Work consensus because the outcome of the computation performed to secure the blockchain is also used to store user data. In Filecoin's consensus, the probability that the network elects a miner to create a new block is proportional to the relation between their storage currently in use and the storage of the rest of the network. This protocol is designed such that miners would rather invest in storage than in computing power. Miners offer storage, and the computation that is used for the proofs is repurposed to participate in the consensus.

The authors describe this consensus protocol as ”extending existing (and future) Proof-of-Stake consensus protocols, where stake is replaced with assigned storage”. In other words, to extend the blockchain, miners do not use the amount of currency they hold in the blockchain but the storage they rented to clients. The amount of storage in use of each miner is publicly verifiable by checking all the PoSpT proofs generated by each.

A miner can add a new block to the chain if the following condition is met:

$$H(\langle t \parallel sign_{M_i}(rand(t))\rangle)/2^L \leq \frac{p^t}{\sum_j p_j^t}$$

Here, rand(t) is a public randomness that can be extracted from the blockchain at epoch t, p_i^t the sum of storage assignments of miner M_i, H is a secure cryptographic hash function which output's length is L and $sign_{M_i}(m)$ to be a message signed by M_i.

The expected number of miners that meet this condition per epoch is 1, but some epochs may have zero or many leaders. In the case of zero leaders, an empty block is added to the chain. In the case of many leaders, the chain can be extended by multiple blocks. This creates forks in the chain and miners choose the chain that reflects the most space rented by the miners. This means that, in cases of multiple leaders at an epoch, transactions introduced into the blockchain take more time to stabilize. Users need to wait for more epochs until they have enough certainty that their transaction persists in the chain.

Filecoin launched in 2020, but there are still open questions about its protocol. One of them is the search for a better leader election for the consensus protocol, which gives exactly one elected leader per epoch.

2.5.3 Subspace

Subspace [12] is a PoSt blockchain proposal that repurposes the space used in PoSp blockchains to archive the history of the blockchain. Wagstaff [12] argues that all current PoSp blockchain constructions are not incentive compatible and suffer from what he calls the farmer's dilemma. In brief, the farmer's dilemma states that in any PoSp blockchain farmers (i.e. miners) need to choose between either maintaining the blockchain state and history or maximizing the amount of space they pledge towards consensus. Rational farmers always choose the latter since it maximizes their rewards. According to Wagstaff, this is a threat to blockchain's decentralized nature since all of the chain's history might be centralized in altruistic archival nodes.

Subspace proposes to solve this dilemma by incentivizing farmers to store the history of the blockchain. For this, farmers who want to participate in the blockchain are required to submit PoRep [17] proofs of the chain's history. In Subspace, farmers store replicas of the partial blockchain history. A farmer first creates and stores these unique replicas by running a PoRep [17] protocol on a constant-sized piece of the confirmed history (i.e., the prefix of the blockchain). To forge a new block, a farmer is required to respond to periodic storage audits with a PoRep proof. These audit challenges are derived from previous blocks of the chain. Miners have specific history pieces assigned to them and they can be determined by hashing their self-assigned ID (i.e. public key) which maps to a position in a hash ring [51].

This approach solves the interactivity issue identified by Park et al. [40] since PoRep can be made fully non-interactive. Wagstaff proposes the following approaches to solve the issues presented by Park et al.

Determine the winner: After the proof initialization process, farmers respond to a challenge provided by storage audits in order to add a new block to the blockchain. Each audit response consists of a valid PoRep proof that has a quality assigned such that the proof value nearest to the challenge value has the highest quality. Subspace does not specify the quality function, but it requires it to be proportional to the pieces of blockchain history stored.

Costless simulation: To solve costless simulation problems, Subspace follows the same solutions proposed by Spacemint [40]. It splits the chain by having one chain store the proofs and another chain store the transactions. Furthermore, the same challenge is used over several consecutive audits.

Long range attacks: In order to prevent attackers from generating a private chain that surpasses the honest one, the chain history stored by each farmer is constantly updated with the new confirmed chain history. The history pieces assigned to each miner are periodically updated and miners are required to generate new proofs for them.

Subspace's concrete solution design is still in progress and details such as the type of PoRep and the consensus protocol are still unknown. Although the concept of ”the farmer's dilemma” in PoSp blockchains has not been thoroughly studied, current statistical evidence from PoSp blockchains such as Chia [1] suggests that it is a valid concern. Currently, the Chia blockchain history occupies 79 Gb of storage space, which represents more than 72% of the minimum storage requirement for Chia's PoSp proofs (108.9 Gb).

2.5.4 Discussion

In this section, we discuss the previously presented work, compare each blockchain's approach trade-offs, and how their mechanisms can assist in achieving our goals. Table 2.1 presents the main aspects of each blockchain.

PoW is a commonly used to prevent Sybil attacks, but it has a large energy consumption which makes it not environmentally friendly and sustainable. Alternative Sybil-proof mechanisms try to solve this energy efficiency issue by using another non-counterfeitable resource. We described in depth PoS, PoSp, and its variants of PoSt. However, there are other alternatives of PoW such as Proof-of-Elapsed-Time [52], which shows that an amount of time was spent using specific trusted hardware.

Proof-of-Stake uses the amount of currency a node holds in the system as non-counterfeitable resource. PoSp uses disk space as a resource. Both are energy efficient and, when applied to blockchains, the mining process is deliberately cheap to compute. This leads to costless simulation issues, which can compromise the blockchain's security either by slowing down consensus or rewriting the chain's history. Therefore, both types of blockchains need to have additional mechanisms to prevent these issues.

PoS blockchains have the additional rich-get-richer problem, which can lead to the centralization of the blockchain into a small group of nodes that hold the majority of the system's currency.

PoSp blockchains, such as SpaceMint [40] and Chia [1], have been proven to be more energy efficient than PoW blockchains and are presented as a more ecological alternative. SpaceMint uses a type of PoSp [37] that has strong security properties but this requires nodes to first register a special transaction in the chain before they start mining. This is a drawback since a node's ability to participate in the blockchain depends on having other participants including his special transaction into their chain.

Chia uses a PoSp [38] with weaker security guarantees but strong enough for its usage in blockchains. Chia does not have the interactivity problem of SpaceMint and uses Verifiable-Delay-Functions to overcome the costless simulation issue. Both of these cryptocurrencies need a higher percentage of honest nodes to maintain the system's security, SpaceMint needs at least 55% and Chia needs at least 63%. Another drawback is that their allocated space is filled with random data which is only used to mine proofs. This is a huge waste of space, especially in Chia, which currently has 32 billion gigabytes of space solely used to maintain the blockchain.

Our work focuses on trying to solve this useless space issue by taking into consideration the ideas of PoSt blockchains such as Permacoin, Filecoin, and Subspace.

The idea of PoSt is to show that a specific data has been stored. This allows this protocol to be used not only as a Sybil-proof mechanism in a blockchain but also for the storage and distribution of data throughout a decentralized network.

Permacoin repurposes the computation used in the Bitcoin mining process to achieve distributed storage of large archival data. Filecoin reuses the proofs computed by the storage providers to prove the storage of their client's data by using them in the mining process. Subspace proposes a way to repurpose the wasted space in PoSp blockchains by archiving the history of the blockchain. Our approach follows the idea introduced in bread pudding protocols that repurposes the energy expenditure in PoW, but in the context of disk space expenditure. In other words, we look for new ways to repurpose the space allocated in blockchains such as Chia into some useful storage utility.

System	Proof	Non-counterfeitable Resource	Useful Work
Bitcoin [5]	PoW	computation	none
Ethereum [6]	PoS	stake	none
SpaceMint [40]	PoSp	space	none
Chia [1]	PoSp & PoT	space & time	none
Permacoin [47]	PoW & PoRet	computation & space	storage of archival data
Filecoin [11]	PoRep & PoSpT	space rented	storage of users' data
Subspace [12]	PoRep	space	storage of blockchain's history
Chia Bread Pudding	**PoRep* & PoT**	**space**	**storage of local data**

Table 2.1: Comparison of the main characteristics of blockchain implementations.

3

Chia Bread Pudding

Contents

In this chapter, we tackle the critical challenge of optimizing storage efficiency in PoSp blockchains, driven by the motivation to unlock the latent potential of storage resources in these systems. We begin our exploration by elucidating the key challenges intrinsic to our vision. In response to these challenges, we unveil our solution, Chia Bread Pudding—an abstract design that reimagines storage optimization within PoSp blockchains. This section also delves into the technical intricacies, architectural components, and integration of Chia Bread Pudding with existing PoSp blockchain-Chia.

3.1 Motivation

Our aim is to optimize the utilization of storage resources within PoSp blockchains by introducing the concept of ”bread pudding protocols” into these systems. Originating from the work of Jakobsson et al. [22], these protocols are methods to repurpose the computational effort inherent in PoW systems, allowing verifiers to engage in distinct, constructive computations. In a similar vein, we adapt the principles of ”bread pudding protocols” to the context of PoSp systems, offering a solution to repurpose the storage space invested in proof generation for practical utility.

The main goal of our work is to reduce the storage waste of PoSp blockchain protocols. In other words, we want to minimize the amount of random data that is generated during proof initialization phase of PoSp blockchains and stored in miners' disks. For this, we allow the miners to use their preexisting useful data as a non-counterfeitable resource. This approach provides the additional property of usefulness, as detailed in Section 2.5. The utilization of preexisting data is deemed useful to miners beyond just securing the blockchain. Under this vision, the system's Sybil-proof mechanism still relies on the concept of PoSp, where the more space a miner allocates, the higher the chance of extending the blockchain. However, instead of generating random data to allocate disk space, miners use local data that is already stored on disk to prove that some amount of storage has been allocated. For this, we design novel bread pudding protocols [22] to efficiently fulfill the above requirements.

3.2 Challenges and Solutions

Using miners' local files as proofs of space is not straightforward. To materialize our vision described in Section 3.1, we must address the following challenges:

1) **Unknown File Structure:** PoSp [37, 38] demonstrates the allocation of space by storing data with specific cryptographic properties. This data is generated based on cryptographic structures and incurs computational and temporal costs. User file structures are unpredictable and do not adhere to the constructions of space proofs, lacking the required cryptographic properties. In addition to the absence of these properties, it is crucial that the file structure remains preserved (i.e., the file content remains unaltered and can be directly read) or recoverable (i.e., the content is modified but can be recovered and read). Miners dedicating their files to the blockchain must be able to retrieve them for other purposes. In other words, using miners' files is only useful if their original state is maintained or recoverable.

2) **Low Entropy:** The data used in the creation of a PoSp proof is generated using hash functions. If the input to these hash functions is repetitive, it will produce repetitive results. The diversity of results is what enables miners to obtain various different proofs and increase their chances of

extending the blockchain. Archived files typically exhibit low entropy since various sections contain repetitive or redundant data [53].

For the aforementioned challenges we propose the following solutions:

1) **Unknown File Structure:** To deal with the unknown file structure we replace the PoSp with a proof technique that allows us to use existing data to prove that specific storage resources were allocated for it. For this, we adapt the cryptographic techniques used in PoRep [17] to suit our requirements. PoRep guarantees the accurate replication of data across locations while confirming the allocation of equivalent storage resources. The replicated data remains retrievable, ensuring the possibility of recovering the original information from this duplicate. Consequently, PoRep functions as PoSp, validating the allocation of a designated space with useful data and assuring its recoverability. Our adaptation of PoRep to the context of allocating miners' files for the blockchain maintains the methods for verifying the space allocated by the files, but it does not encompass integrity verification. This is attributed to each miner's ownership of their files, eliminating the necessity for integrity verification. Furthermore, by not verifying the content of the files we allow each miner to choose to either use preexisting data or generate random data for the blockchain. Miners are incentivized to use existing data since they do not need to allocate more resources for random data. By allowing this dynamic, we expand the domain of potential participants of our system whilst also reducing the storage waste associated with traditional PoSp blockchains. The specific algorithm used for the miners' file allocation to the blockchain is described in Section 3.3.2.

2) **Low Entropy:** To address this challenge, we propose encrypting the original files. When data is encrypted, it undergoes a process where it is converted into ciphertext using an encryption algorithm and a specific key. This process obscures the original information, by ensuring the underlying patterns of data remain unpredictable. This obfuscation increases the entropy of the data and ensures the privacy of miners' data is protected. Using a cipher in Cipher Block Chaining (CBC) [54] mode increases entropy and ensures a certain level of privacy for miners' data. Furthermore, the cipher allows for sequential encryption and parallelizable decryption. Thus, the proof initialization process remains sequential while the recovery of the original data is efficient.

3.3 Chia Bread Pudding

In this section, we describe our protocol by integrating the solutions elucidated in Section 3.2. Our exposition begins with an overarching view of our blockchain, followed by an in-depth exploration of the pivotal algorithms integral to our protocol's functionality.

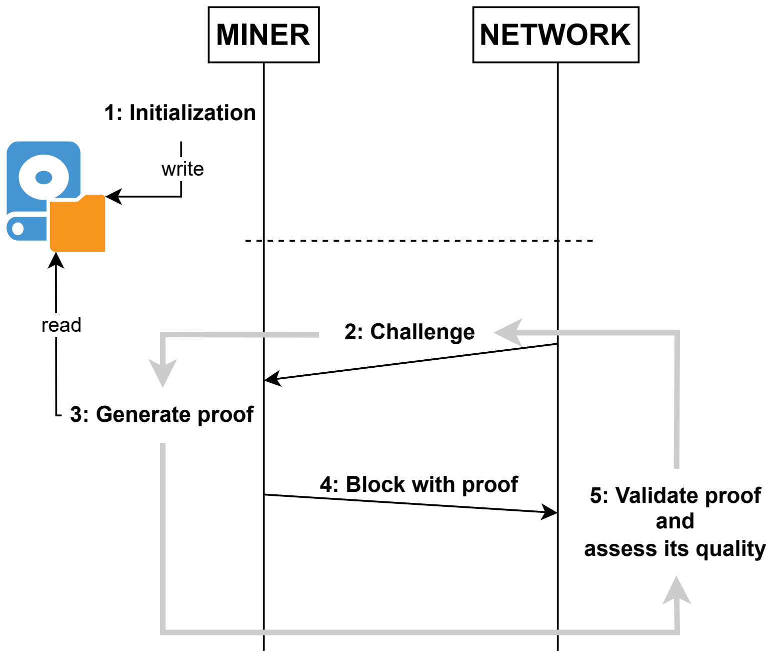

Figure 3.1: Base protocol

3.3.1 Base protocol - Chia

Chia Bread Pudding redefines blockchain by focusing on the concept of useful storage allocation. Despite advances, challenges raised by Park et al. [40], such as costless simulation and long-range attacks, continue to be prominent concerns for our blockchain ecosystem. To effectively tackle these challenges head-on, we embrace Chia's protocol [1], incorporating its proven solutions to these issues as the foundational framework of our system, thereby bolstering our defenses against potential threats. Figure 3.1 illustrates the abstract blueprint of a PoSp blockchain, including Chia. Within our protocol, miners start by undergoing an initialization process before actively participating. Upon completion of this initialization phase, miners are equipped to engage in the mining process. They receive challenges from the last block in the blockchain, generate proofs based on data stored during initialization, and extend the chain by incorporating newly generated proofs in the new blocks. Subsequently, the network verifies proofs submitted by multiple miners, ultimately selecting the block with the highest-quality proof.

When adapting a PoSp blockchain like Chia, we need to ensure that the cheap process of creating a proof from the stored initial data does not lead to common risks mentioned in Section 2.3 and Section 2.4. These issues identified by Park et al. [40] are addressed as follows:

Interactivity: Miners use their own local data to compute the initialization step. This step of our adapted PoRep does not require an initial commitment to the blockchain. Miners are able to participate in the blockchain once they finish the initialization step.

Determine the winner: Just like Chia's original protocol, our protocol specifies a quality for each

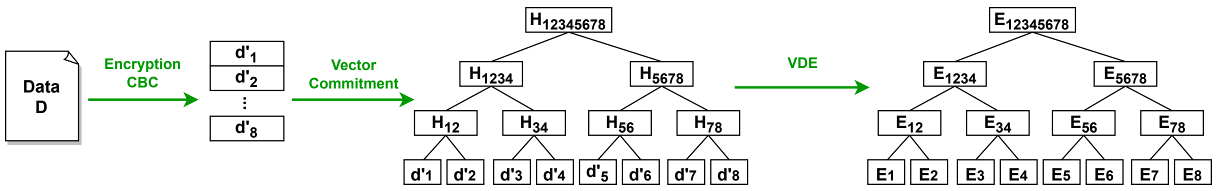

Figure 3.2: Initialization process

proof. The proof with the best announced quality is considered to be the extension of the chain. Similarly to Chia, the probability of a miner successfully extending the chain is proportional to the space allocated, which is measured using our own proof instead of a PoSp. The honest chain is determined by the highest accumulated quality of a chain.

Costless simulation: Our proposal keeps Chia's original methods for dealing with costless simulations. The transactions and the proofs of a block are decoupled into two separate chains.

Long-range attacks: Chia's original method for preventing attackers from rewriting the blockchain's history is kept in our proposal. However, instead of alternating between PoSp and PoT proofs, our proposal replaces PoSp with our adaptation of PoRep. The PoT proofs are computed using Verifiable Delay Functions (VDFs) that take a predetermined time to compute. The time spent in VDFs is defined by the quality level of each space proof. Higher-quality proofs result in smaller computing time in the VDF.

Our protocol modifies Chia's protocol by altering the following algorithms:

1) Initialization (Figure 3.2);

2) Proof Generation (Figure 3.3)

3) Proof Validation (Figure 3.4);

4) Proof Quality (Equation 3.1).

3.3.2 Initialization

Prior to a miner's engagement in the process of block creation to extend the blockchain, a mandatory initialization phase must be executed. The initialization process, as depicted in Figure 3.2, is a multi-faceted operation that transforms a given data (e.g. a set of files stored in the miner's local File System) into a structure possessing specific cryptographic attributes, and it is executed in three distinctive steps.

The first step involves dividing the data into blocks of a uniform and predetermined size m, followed by their encryption through the utilization of a cipher in CBC mode. This encryption serves several pivotal purposes within our protocol. Firstly, it introduces a layer of obfuscation and randomization to the original data, thereby preserving data privacy and elevating data entropy. Additionally, this encryption methodology ensures the recovery of the data, either in its original form or through a systematic recovery process. Notably, the use of block-chaining techniques results in sequential encryption, yet decryption is inherently parallelizable, enhancing the overall efficiency of data recovery and subsequent blockchain operations.

In the second step, we employ a Vector Commitment (VC) protocol [55] to secure the encrypted blocks. A VC is a cryptographic technique that enables a party to commit to a vector of values while allowing for later individual value revelations, all while providing cryptographic proofs to verify that the revealed values correspond to the committed ones. This commitment protocol offers efficiency in both commitment, which involves creating a commitment to the entire vector, and opening, where individual values within the vector are revealed. Importantly, both commitment and opening operations incur minimal computational overhead. In our implementation, we opt for a Merkle tree [56] as our chosen VC scheme. In this scheme, each encrypted block serves as a leaf in the Merkle tree. These leaves are grouped together by concatenating a fixed number, denoted as fanout f, of blocks in each group. A cryptographic hash function is then applied to each group, producing a fixed-size hash value unique to the content of the data blocks within that group. These hashed values become parent nodes to those associated with the hashed group. This grouping and hashing process is applied recursively to the parent nodes until only one hash value remains, which represents the Merkle root.

In the third step, we apply a Verifiable Delay Encoder (VDE) [17] to encode every node within the Merkle tree. This encoding process is deliberately time-consuming and its duration is proportional to both the data size and the VDE time. Notably, VDEs are slow in encoding but fast in decoding. The encoding time is adjustable as it is parameterizable, while the decoding time remains relatively constant. The encoding/decoding scheme relies on VDFs [41,42]. A VDF is a non-parallelizable function that takes a predetermined amount of time to compute. The encoding process is executed through a sequential and chained technique, wherein the encoding of one node is intricately tied to the encoding of other nodes, akin to the encryption process used in block-chaining cipher modes. Algorithm 3.1 describes this chain encoding in the Enconde_Tree function. The process starts by encoding the root node using a VDE in line 2. Then, the hash of the encoded root is used to encode each of its child nodes in lines 3 to 5. All the rest of the nodes are encoded using the encodings of their parents' sibling nodes in lines 6 to 10. Note that all of the nodes that are not directly related to the root of the Merkle tree go through the same encoding process using the function Encode described in Algorithm 3.1. This function takes as input the node with children to be encoded and a key. The key is the hash of all of the node's sibling nodes (the nodes that share the same parent with the input node) concatenated together. Employing this key, each child's data is subjected to XOR operations with the key, followed by VDE execution. This iterative process continues for each subsequent child node of the input node until the leaf nodes are successfully encoded.

Algorithm 3.1: Encode_Tree

```
Function Enconde_Tree($root$):
    $root.data \leftarrow$ VDEncode($root.data$)
    for $child$ in $root.children$ do
        $child.data \leftarrow$ VDEncode($child.data \oplus$ Hash($root.data$))
    end
    for $child$ in $root.children$ do
        $siblings \leftarrow root.children \backslash \{child\}$
        $key \leftarrow$ Hash($sibling_1.data||...||sibling_{f-1}.data$)
        Enconde($child$, $key$)
    end

Function Enconde($node$, $key$):
    if $node$ is not $null$ then
        for $child$ in $node.children$ do
            $child.data \leftarrow$ VDEncode($child.data \oplus key$)
        end
        for $child$ in $node.children$ do
            $siblings \leftarrow node.children \backslash \{child\}$
            $key \leftarrow$ Hash($sibling_1.data||...||sibling_{f-1}.data$)
            Enconde($child$, $key$)
        end
    end
```

This initialization process can be repeated for various files. Each file is transformed into an encoded Merkle tree. Using more files during initialization increases a miner's chances of extending the blockchain due to a greater data diversity enabling more proofs.

The initialization process is intentionally configured to be time-consuming, serving as a strategic incentive for miners to retain and preserve the data generated during this crucial phase. The substantial time commitment associated with initialization predominantly stems from the amalgamation of the sequential and chained technique with VDE. This deliberate choice of creating a time-intensive initialization process discourages miners from discarding the valuable data produced during this phase. By compelling miners to dedicate time and effort to the data generation process, our protocol reinforces a sturdy foundation for the blockchain's security, ensuring the enduring relevance of the generated data throughout the blockchain's lifetime.

3.3.3 Proof Generation

Proof generation is a crucial process that enables miners to demonstrate the completion of the initialization process. It involves the creation of a proof that validates the storage of data during the initialization. Miners undertake this process post-initialization to actively participate in the blockchain.

A proof in our protocol is essentially an opening of the selected VC protocol, implying the disclosure

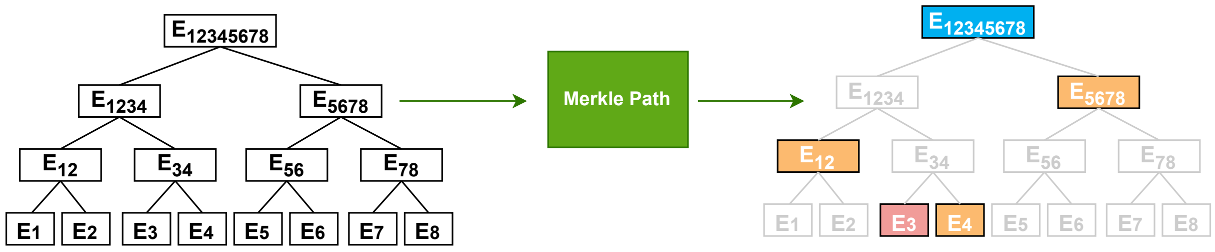

Figure 3.3: Proof Generation

of a subset of the data stored during the initialization process. In the context of a Merkle tree, an opening corresponds to the selection of specific nodes within the tree. This selected subset of nodes includes a leaf and its corresponding Merkle path, which constitutes the minimal number of additional nodes in the tree necessary for the computation of the root hash. Each leaf of the Merkle tree possesses its corresponding Merkle path. To ensure that miners do not merely store a single valid Merkle path, we enforce a random selection of the leaf node. To achieve this, we leverage the data from the most recent block of the blockchain, facilitating the determination of the leaf node for the proof.

The proof generation process begins when a miner receives a blockchain challenge (e.g. the PoT of the latest block, represented as τ in Figure 2.1), which consists of a 256-bit number. The miner selects the Merkle tree with the best quality and creates a proof using the tree nodes. This quality is defined later, in Section 3.3.5. Proof generation involves selecting the leaf indexed by the modulus of the challenge with the total number of leaves (in Figure 3.3, it's leaf E3). Then, we create a Merkle tree path from the selected leaf (shown in orange in Figure 3.3). Finally, the selected leaf is concatenated with the Merkle tree path and the root to generate the proof (in Figure 3.3, $proof = E_3||E_4||E_{12}||E_{5678}||E_{12345678}$). Once a miner concludes the proof generation, he extends the blockchain with a block containing the new proof.

The proof generation is designed to be efficient for honest miners (those who completed the initialization phase and stored the data). This process is intended to be used multiple times after the initialization. Consequently, over time, the computational effort invested during initialization is amortized, given that the data can be leveraged to generate multiple proofs over the lifespan of the blockchain.

3.3.4 Proof Validation

The process of proof validation assesses whether the presented proof and its associated challenge fulfill the necessary criteria for acceptance within the blockchain. This validation consists of multiple verifications, each serving a specific purpose to the blockchain's security. This validation encompasses the following steps:

1. verify cryptographic proof;

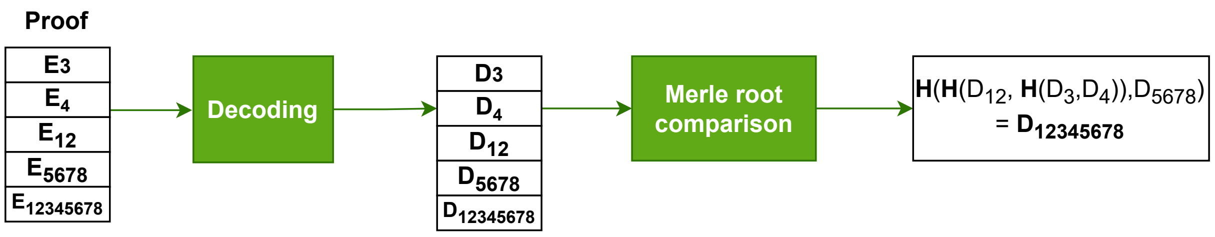

Figure 3.4: Proof Validation

2. verify challenge;

3. verify minimum proof quality.

The first verification validates the proof using cryptography methods. This step ensures that the proof creator is committing a specific amount of data to the blockchain, confirming that the proof was indeed generated using the data stored after the initialization process. This verification, depicted in Figure 3.4, consists of two key steps: 1) Decoding, and 2) Merkle root computation and comparison. In the decoding step, the receiving miner decodes each node with a Verifiable Delay Decoder (VDD) (the inverse of VDE). Algorithm 3.2 describes the decoding process of a proof with p tree nodes. Each node is decoded using the hash value of its following encoded nodes. Algorithm 3.2 is the inverse of Algorithm 3.1 applied to a single proof. Every node except the root node is decoded using a key value, as described in lines 19 to 34. This key value is the hash output of a subset of encoded nodes from the proof. This subset of nodes is composed of the ones used in the encoding process, therefore not every node has the same key value. In Algorithm 3.2 lines 3 to 6, the leaf nodes are decoded by first XORing each leaf with the key and then executing a VDD on the resulting value. The same process goes for the decoding of the intermediate nodes in lines 7 to 12, and the nodes directly related to the root node in lines 13 to 15. Finally, the root is only decoded with the VDD in line 16. Once the nodes are decoded, the process proceeds to the computation and comparison of the Merkle root. This step involves computing a new root from the decoded nodes and comparing it with the root provided in the proof. Computing a Merkle root from the decoded nodes involves an iterative hashing process, where the leaf node is hashed with a node from the Merkle path, and the result of this hash is subsequently hashed with the next node from the path. This hashing process continues until it reaches the last node of the Merkle path. The resulting output of the final hash represents the newly computed Merkle root. Finally, the computed root is compared with the root provided in the proof. The proof is deemed valid if the two roots are a match.

The second verification validates the challenge that is associated with the proof. In this regard, we adhere to the original Chia approach for validation. Specifically, a challenge is deemed valid if it falls within the category of the three most recent PoT proofs. Consequently, miners are allotted a window of

Algorithm 3.2: Decode

```
Function Decode($encoded[p]$):
    $decoded \leftarrow \emptyset$
    for $i = 1; i \leq f; i+ = f - 1$ do
        $key \leftarrow$Compute_Key($encoded, i$)
        $decoded[i] \leftarrow$ VDDecode$(encoded[i] \oplus key)$
    end
    for $i = f + 1; i \leq p - f; i+ = f - 1$ do
        $key \leftarrow$Compute_Key($encoded, i$)
        for $j = 0; j < f; j+ = 1$ do
            $decoded[i + j] \leftarrow$ VDDecode$(encoded[i + j] \oplus key)$
        end
    end
    for $i = p - f + 1; i < p; i+ = 1$ do
        $decoded[i] \leftarrow$ VDDecode$(encoded[i] \oplus$Compute_Key($encoded, i$)$)$
    end
    $decoded[p] =$ VDDecode$(encoded[p])$
    return $decoded$

Function Compute_Key($encoded[p], index$):
    $nodes \leftarrow \emptyset$
    if $index \leq f$ then
        for $i = 1; i \leq f - 1; i+ = 1$ do
            $nodes \leftarrow nodes||encoded[f + i]$
        end
    end
    else if $index \leq p - f$ then
        for $i = f - 1; i < 2f - 2; i+ = 1$ do
            $nodes \leftarrow nodes||encoded[index + i]$
        end
    end
    else
        $nodes \leftarrow encoded[p]$
    end
    return Hash$(nodes)$;
```

9.375-28.125 seconds to generate and share a proof for each challenge. This validation of challenges ensures that the data created during the initialization process remains stored and utilized throughout the blockchain's lifespan. Due to the time-intensive initialization process and the limited validity period of each challenge, miners are unable to generate data on the fly and thus are compelled to retain the data generated during initialization.

Finally, and similarly to the original Chia protocol, a proof is deemed valid only if its quality value falls below the changing $difficulty$ value, which evolves over the course of the blockchain's existence. Further elaboration on the proof quality function is provided in the next subsection.

3.3.5 Proof Quality

Proof quality refers to a function that assigns a value to a given proof, serving the dual purpose of proof validation and ranking the proofs submitted by miners for inclusion in the blockchain. Notably, a lower output value of this function corresponds to a superior proof. Our adaptation of Chia's initial quality function results in the following calculation for the quality value:

$$quality = \frac{difficulty * sha256(merkle\_root||challenge)}{plot\_constant} \tag{3.1}$$

Both the $difficulty$ and $plot\_constant$ parameters are defined by the original Chia protocol and remain integral to our solution. The $difficulty$ is a variable that is updated periodically to moderate the rate of proofs that are accepted by the blockchain. The $plot\_constant$ parameter is a constant that indicates the size of the plot, i.e. the size of the data generated in the initialization process. In Chia, it is possible to generate plots of different sizes, but in our case, we only allow for one universal size.

The quality function depends on the root of the Merkle tree given by the proof. This allows for an efficient search for proof with quality without having to look up every tree node. Furthermore, having the quality be dependent on the Merkle root encourages miners to allocate more files to the blockchain, as each set of files generates a new root.

3.4 Practical Considerations

In this segment, we delve into the practical intricacies of our protocol, focusing on their usability for miners. Specifically, we outline the process of recovering and utilizing the data allocated to the blockchain, elucidating its impact on the overall blockchain protocol.

3.4.1 Original Data Retrieval

For our protocol's storage resource allocation to be deemed useful, the ability to retrieve the original data from the stored information during the initialization phase (discussed in Section 3.3.2) is crucial. When a miner with files allocated to the blockchain seeks to access them, the original state of the data can be generated by decoding and decrypting a portion of the Merkle trees associated with the specific files. The decoding process mirrors that described in Algorithm 3.2, albeit applied to all the leaf nodes of the tree rather than a subset included in a proof. Post-decoding, the miner can generate the original data by decrypting all the data of these leaves.

The process of data retrieval involves generating the original data using the encoded data, without modifying the associated Merkle trees, enabling their continued use in generating proofs for the blockchain. The retrieval process does not alter the data designated for the blockchain, thus it does not impede miners' participation in the blockchain. The process of data retrieval adds a certain level of computational overhead due to the necessity of decoding and decrypting data. Nevertheless, this overhead has a negligible effect on the performance, as the decoding and decryption procedures are designed to ensure efficiency and parallelizability.

3.4.2 Data Modification and Adaptation

Following the allocation of files to the blockchain, miners may find the need to modify and render the earlier unaltered data irrelevant. To achieve this, the miner retrieves the original data (described in the preceding section), makes modifications, removes the previous data, and reallocates the updated data to the blockchain via the initialization process. The proofs generated from the deleted data that have already been integrated into the blockchain remain valid. Once the initialization process for the new files is completed, the miner can proceed to utilize them for proof generation.

Furthermore, it is worth noting that the process of modifying files and subsequently reinitializing them is not expected to occur frequently. This is because write operations are generally less frequent than read operations, which corresponds with the prevalent Write Once Read Many (WORM) paradigm observed in numerous storage systems. One such example is the prevalence of WORM storage systems [57–59], often employed for long-term archival objectives. As a result, the tendency for the data allocated to the blockchain to be modified will also be infrequent, aligning with the overall usage pattern of such storage systems.

3.4.3 Optimizing Storage Allocation: Useful and Useless Data

As expounded upon in Section 3.2, Chia Bread Pudding introduces a novel approach that allows miners to allocate both useful and useless data to the blockchain. The protocol primarily focuses on the amount of storage resources designated for the blockchain, without delving into the specifics of the stored content. The initialization process described in Section 3.3.2 can be used either on useful data or useless data. By affording miners the discretion to choose between useful and useless data, they gain the flexibility to tailor their allocated blockchain data in accordance with their individual requirements.

For instance, consider a scenario where a miner possesses a storage disk of 250GB, containing 100GB of data. In this case, the miner can designate 100GB of useful data to the blockchain, while generating additional useless data to occupy the remaining storage space, which can also be allocated to the blockchain. The generation of useless data can be as straightforward as creating random data, such as storing consecutive byte values starting from a selected nonce. Subsequently, as the miner generates more useful data and seeks to incorporate it into the blockchain, the previously designated useless data can be removed and replaced with the useful data via the initialization process.

This distinctive design choice facilitates the optimization of storage resources, accommodating the diverse contextual needs of individual miners.

3.5 Public Parameters

Our proposed protocol incorporates several adjustable public parameters that influence its performance and implementation decisions:

1. Block and node size, denoted as m.
2. Number of Merkle tree leaves, denoted as n.
3. Merkle tree fanout, denoted as f.
4. VDE execution time, denoted as t.

The tuning of these parameters has a substantial impact on the system's efficiency and implementation choices. It is worth noting that these parameter definitions also hold sway over crucial considerations concerning Hash Function selection and VDE output sizing, with a particular emphasis on ensuring alignment with the specified value of m. In Section 4.3, we delve into a comprehensive analysis of the trade-offs associated with these parameters.

3.6 Summary

This chapter focuses on addressing the challenge of optimizing storage efficiency in PoSp blockchains. It introduces the concept of Chia Bread Pudding, a protocol designed to repurpose miners' existing data for blockchain participation. Drawing inspiration from the concept of ”bread pudding protocols” and building upon the Chia blockchain protocol, this solution aims to minimize storage waste and redefine the PoSp concept. The primary motivation behind this research is to enable miners to utilize their local files as non-counterfeitable resources, thereby enhancing the overall efficiency of PoSp systems.

The chapter outlines the critical challenges associated with using miners' local files as proofs of space and presents effective solutions. It emphasizes the importance of preserving and securing miners' files, addressing low entropy concerns through encryption of original data. By proposing the use of PoRep over PoSp and implementing CBC for encryption, the study aims to enable miners to repurpose their existing data for blockchain participation. Additionally, it highlights the significance of proof quality in the validation and selection process within the Chia Bread Pudding protocol. Lastly, it delves into practical considerations governing the design of the protocol, particularly emphasizing the usability of the data allocated to the blockchain.

The proposed Chia Bread Pudding protocol serves as a comprehensive framework that leverages the Chia blockchain's foundations. By replacing PoSp with an adaptation of PoRep, the protocol ensures efficient storage allocation and offers a detailed analysis of its initialization, proof generation, and validation algorithms. The chapter also emphasizes the importance of adjustable public parameters, such as block and node size, Merkle tree parameters, and VDE execution time, in shaping the protocol's overall performance and operational efficiency.

4

Evaluation

Contents

In this chapter, we undertake the evaluation of Chia Bread Pudding. We start by elucidating the research questions that serve as the guiding compass for our investigation. Following this, we expound upon our evaluation methodology, delineating the chosen metrics. Finally, we encompass a comprehensive examination comprising both analytical and experimental evaluations.

4.1 Research Questions

The goal of this dissertation is to study techniques that repurpose the data used in PoSp blockchains for some useful purpose besides securing the blockchain. To this end, we identify the following two relevant research questions:

RQ1: To what extent can we replace Proof-of-Space blockchain's random data with useful data?

RQ2: What is the impact of repurposing the data used in Proof-of-Space blockchains?

To answer these research questions, we 1) define the evaluation metrics for all RQs; 2) collect the results on these metrics to evaluate our proposed solution.

Our evaluation consists of two parts: 1) analytical evaluation and 2) experimental evaluation. In the analytical evaluation, we analyze the impact of each public parameter of the protocol listed in Section 3.5 to find the best configuration. In the experimental evaluation, we run the proposed protocol with specific public parameters and collect the results.

4.2 Metrics Selection

To assess our proposed solution and address our research questions (RQs), we introduce the following set of quantitative metrics:

- **Useful Space Ratio:** Proportion of useful space in relation to total space used for proofs. Allows evaluation of the space optimization by indicating how much space can be utilized effectively.
- **Proof size:** The size of the proofs submitted to the blockchain directly affects the amount of data that needs to be stored and transmitted. This allows us to evaluate the performance of our system.
- **Initialization time:** The time it takes to generate the data used for future proofs during the initialization phase. This metric measures the efficiency of our solution in terms of time.
- **Proof Entropy:** Diversity of the generated proofs. This allows us to assess if our solution is able to produce a high variety of proofs.
- **Throughput:** The number of valid transactions per second that our system is capable of handling. It allows us to measure the efficiency and capability of our system.

4.3 Analytical Evaluation

In the analytical evaluation section, we aim to gain an understanding of how individual public parameters affect the overall protocol. Through analysis and mathematical modeling, we deduce expressions

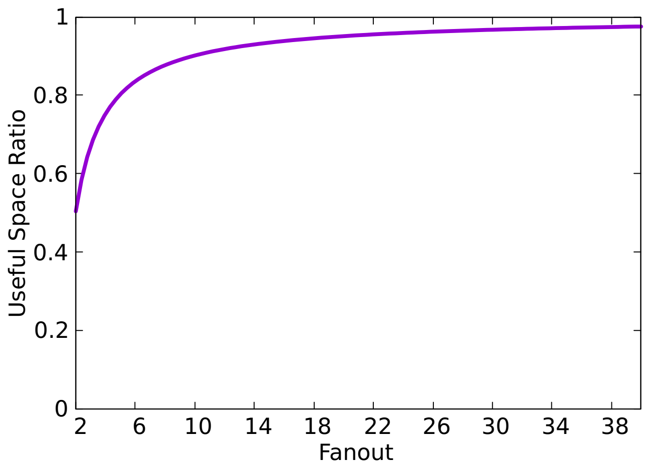

Figure 4.1: Useful space ratio as a function of the Merkle tree fanout for n = {64, 128, 256, 512, 1024}.

that capture how these parameters influence the protocol's efficiency and effectiveness in repurposing storage. This evaluation studies the relationship between individual public parameters within the protocol and its performance metrics, including useful space ratio, proof size, and initialization time. These mathematical insights provide guidance for parameter tuning, contributing to the refinement and enhancement of our blockchain solution.

4.3.1 Useful Space Ratio

The useful space ratio depends on the following parameters:

- n : the number of leaves of the Merkle tree;
- f : the fanout of the Merkle tree.

The ratio, defined as Ratio $= \frac{\text{Useful Space}}{\text{Total Space}}$, is expressed by the expression:

$$R(n, f) = \frac{n(f-1)}{fn-1}$$

To break down this concept further, let's consider its components:

- Total Space: This encompasses the entire space occupied by the Merkle tree, including its nodes.
- Useful Space: This specifically pertains to the space occupied by the leaves of the Merkle tree.

The space allocated by the leaves is determined by the product of m (block size) and n (number of leaves). Meanwhile, the total space is calculated by multiplying m by the number of nodes in a complete Merkle tree with n leaves and a fanout f.

Calculating the number of nodes in the Merkle tree involves understanding that the tree's levels form a geometric sequence, with the root being the first level (f^0), the second level having f nodes (f^1), the third level having f^2 nodes, and so on until the last level with n nodes (leaves). The total number of nodes is obtained by summing the nodes across all levels.

This summation can be expressed using the formula for the sum of a geometric series, which yields:

$$sum = \frac{1 - f^d}{1 - f}$$

Here, d represents the number of layers in the tree, which is $\log_f n + 1$.

Substituting this value for d, we can simplify the expression to:

$$sum = \frac{1 - f^d}{1 - f} = \frac{1 - f^{\log_f n + 1}}{1 - f} = \frac{1 - fn}{1 - f}$$

Finally, the Ratio is calculated by

$$Ratio = \frac{Useful\_Space}{Total\_Space} = \frac{mn}{m\frac{1-fn}{1-f}} = \frac{n(f-1)}{fn-1}$$

Examining this expression and its graphical representation in Figure 4.1, we observe that higher fanout values lead to an increased useful space ratio. This phenomenon is due to the reduction in the number of intermediate nodes in the Merkle trees as the fanout increases. Since these intermediate nodes are not considered part of the useful space, their reduction effectively decreases the total space, ultimately enhancing the overall ratio. Notably, the useful space ratio experiences a significant boost for fanout values ranging from 2 to 10. Consequently, in practical applications, the fanout parameter should be adjusted within this range, as the subsequent intervals yield only marginal increases in the useful space ratio.

4.3.2 Proof size

The proof size depends on the following parameters:

- n : the number of leaves of the Merkle tree;
- f : the fanout of the Merkle tree;
- m : block/node size.

The proof size is given by the expression

$$S(m, n, f) = m((\log_f n)(f - 1) + 2)$$

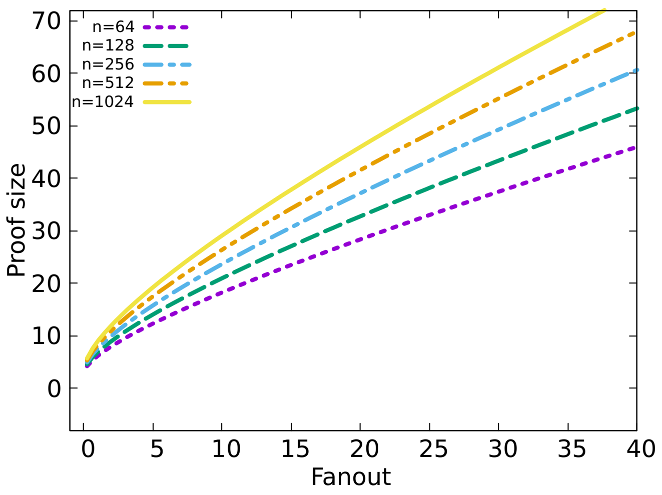

Figure 4.2: Proof size (multiple of m) as a function of the Merkle tree fanout for n = {64, 128, 256, 512, 1024}.

To elaborate, the proof comprises the nodes along a Merkle Tree path, including the root node. Consequently, the proof's size is computed as the product of m with the number of nodes aforementioned. The number of nodes forming a Merkle path is derived from $(\log_f n)(f-1)+1$.

Analyzing the proof size expression and the accompanying graphical representation in Figure 4.2, it becomes apparent that the proof size escalates with both the fanout (horizontal axis in the figure) and the number of tree leaves (as indicated by the various functions in the figure). In practice, the proof size should be tuned in accordance with the blockchain's block size and the respective performance requirements.

4.3.3 Initialization time

The initialization time depends on the following parameters:

- $v(m,n,f)$: the vector commitment execution time;
- $e(m,n,f)$: the chained encoding execution time;
- t : VDE execution time;
- $c(m,n)$: the cipher execution time.

The initialization time is given by the expression:

$$T(n,f,t) = c(m,n) + v(m,n,f) + t \times e(m,n,f)$$

This process's execution time is the cumulative sum of the encryption, vector commitment, and encoding steps. The encryption duration is influenced by the initial data size. The vector commitment phase involves the construction of the Merkle tree, thereby depending on the data size and the tree's fanout. The encoding phase is influenced by the VDE execution time and the chained encoding execution time. Moreover, the encoding time is directly proportional to the VDE execution time. It is essential to note that the execution times are subject to the algorithm's implementation and the underlying hardware configuration.

The expression for the initialization time emphasizes its augmentation with the increase in VDE execution time and the data's size, i.e., m and n. In practical scenarios, the execution time should be adjusted to be time-consuming, motivating miners to store the data produced during this process instead of computing it dynamically.

4.3.4 Parameter tuning

Parameter tuning represents a critical aspect in shaping both the performance and security of the system.

For optimal system performance, we aim to minimize proof size and increase the useful space ratio. The proof size influences the system's complexity, impacting both block transmission within the chain and blockchain storage by miners. The useful space ratio directly reflects the utility of the data being stored by miners. If this ratio is too small, the system approaches current PoSp blockchains where stored data serves no purpose beyond system security. Therefore, the public parameters should be tuned for a high ratio and low proof size to provide more utility to the stored data and establish a more sustainable dynamic for blockchains. For this, f should be high enough to provide more useful space but not high enough that it generates a proof size that allocates a great portion of the block.

Parameter tuning is a critical aspect influencing the system's overall security. The temporal and spatial resources invested in the initialization phase hold a direct bearing on the protocol's security robustness. When the time and space allocation during the initialization step are not cost-prohibitive, miners lack the necessary incentives to adhere to the original protocol, where data is processed and stored on disk for subsequent reuse. Additionally, the scale of data utilized for initialization must be sufficiently substantial to deter any attempts at parallelization. The data employed for each initialization process remains independent of the data utilized in other initialization processes. Thus, if this data is not on a significant scale, miners can potentially parallelize multiple initialization processes, resulting in reduced overall time spent on each individual process.

We discuss the concrete values used in our experimental evaluation in the next section.

4.4 Experimental Evaluation

In the experimental evaluation, we aim to evaluate our blockchain solution through empirical observations. In this section, we tune our protocol's public parameters in order to achieve similar performance results as Chia's original protocol. We present the results obtained for each evaluation metric from running both our developed protocol and the original Chia protocol on the same setup.

4.4.1 Experimental Setup

To experimentally evaluate the developed protocol, we ran the implementation of our protocol alongside the original Chia initialization protocol [60], which served as our baseline reference.

All the experiments were run in a machine with an Intel(R) Xeon(R) Silver 4116 CPU 2.10GHz, with 64GB of RAM.

User data was generated using the DEDISbench generator [61] in "Personal Files" mode, which follows the distribution of real user storage systems.

The public parameters used in the evaluation were as follows:

- $m = 256$ bits;
- $n = 2^{18} = 262144;$
- $f = 2;$
- $t = 0.5s.$

These parameter settings were selected for the purpose of the simulation, enabling us to assess the system's performance without the need to store extensive data batches during the initialization process.

4.4.2 Prototype Implementation

In order to evaluate the effectiveness of our solution, we developed two prototypes to assess different metrics. The first prototype focused on implementing the various components associated with the cryptographic proof, including the initialization, proof generation, and verification processes. Conversely, the second prototype aimed to simulate a blockchain environment.

For the cryptographic components prototype, we employed C++ programming, adhering closely to the description provided in Chapter 3. Within this implementation, we utilized the OpenSSL library [62] for both the Hash function and the CBC encryption. Our chosen hash function was SHA256, while the AES-256 CBC algorithm was adopted for data encryption. Our prototype employs a simulated version of a VDE. Due to the unavailability of an actual VDE implementation, we resorted to using a slow-timed hash function, SLOTH [63]. Specifically, we leveraged the SLOTH implementation [64] was configured

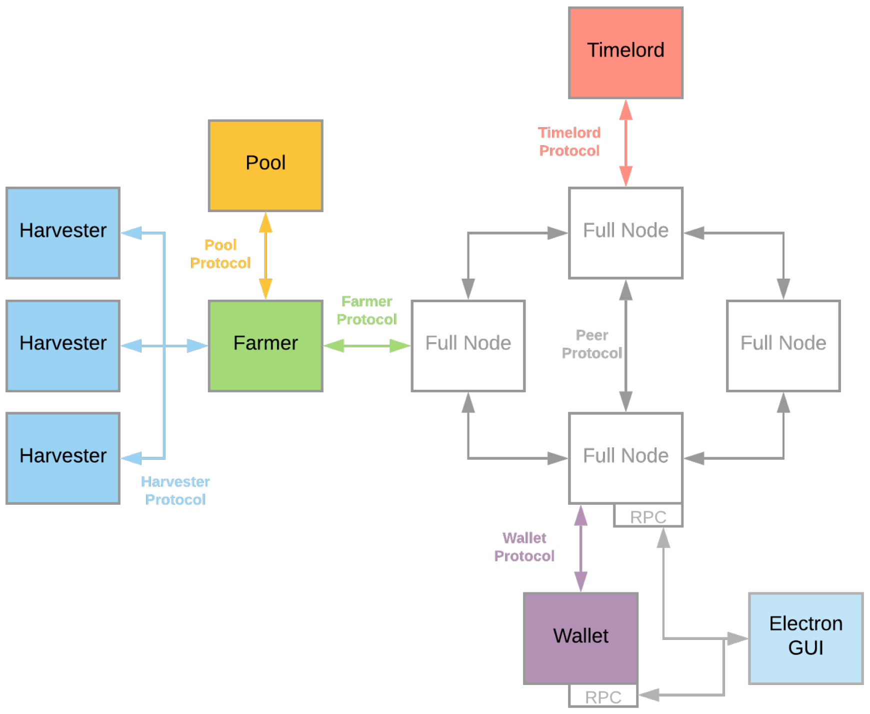

Figure 4.3: Chia Network Architecture Overview

to operate at the desired pace. This SLOTH function serves as a stand-in for emulating the execution and verification time associated with the VDE process in our prototype.

Our second prototype, which simulated the blockchain environment, involved modifying the existing Python code of Chia's blockchain [65] implemented in Python. Considering that conducting experiments with a substantial number of miners would necessitate a significant amount of storage space, we adapted Chia's implementation to circumvent the generation of large data volumes. This was achieved by disabling the Plotting process (i.e. the initialization process), along with the Harvester and Farmer protocols (additional processes facilitating the search and generation of proofs), as depicted in Figure 4.3. Specifically, the modifications ensured that the Full Node utilized a predetermined set of cryptographic proofs during its operation.

In order to establish a connection between our blockchain prototype and the functions implemented in the cryptographic components prototype, we utilized the Pybind11 [66] library for creating Python bindings of the existing C++ code.

Table 4.1: Execution time in minutes for the initialization process for 6, 12, 25, 51, 105 GB.

Protocol / Size of generated data(GB)	6	12	25	51	105
Chia	68.9	130.3	252.5	514.3	1352.2
Chia Bread Pudding	72.8	142.5	273.1	528.6	1171.8

4.4.3 Useful Space Ratio

The useful space ratio in our experimental setup can be computed using the formula outlined in Section 4.3. For the given values of $m = 256$ and $f = 2$, the resulting useful space ratio stands at 50%. This marks a substantial enhancement when compared to Chia's original protocol, which achieves a 0% useful space ratio. Moreover, this improvement translates into a 50% reduction in the free space required on the miner's side, effectively decreasing storage demands.

4.4.4 Proof size

The proof size for our protocol with the defined parameters is 640 bytes. Chia's proof size is 256 bytes. Our proof is 2.5 times larger than Chia's.

Having a larger proof leads to more data being transmitted and stored in the network since a larger proof leads to a larger block in our blockchain. But considering that the maximum block size in Chia is 400KB, our protocol's proof size remains practical, as it is only 0.16% of a block.

4.4.5 Initialization time

To analyze the execution time of the initialization process, we ran our prototype with user data of multiple sizes. Table 4.1 lists the initialization execution time in minutes for data of various sizes (6, 12, 25, 51, and 105 GB). These sizes were selected through Chia's space parameter k, which controls the size of each proof and the allocated disk space.

The execution times for each protocol are similar for the defined public parameters. Our solution has slightly higher execution times than Chia. However since the initialization process is only run once for each initial data, the difference in initialization times does not impact the security or the blockchain's performance

4.4.6 Proof Entropy

To analyze the entropy of the proofs, we ran the initialization process with 105BG of user data and compared the resulting proofs distribution with a uniform distribution.

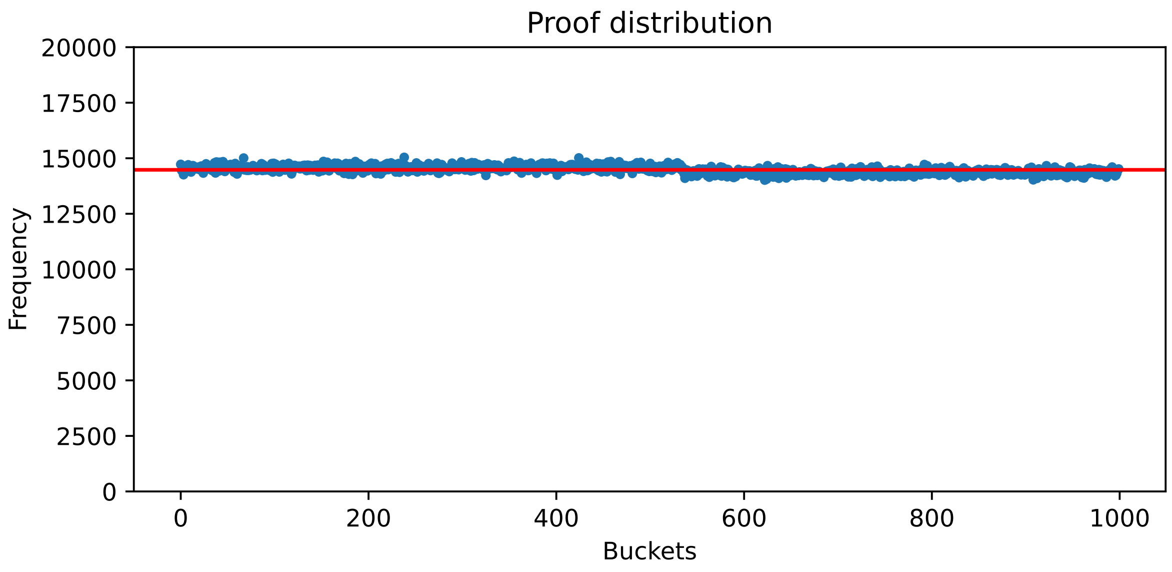

Figure 4.4: Proof distribution of 105GB of data generated during the initialization process(in blue) and the uniform distribution(in red).

To obtain the distribution of the proofs, we considered each proof as a hexadecimal number categorized from 0 to 1000 and recorded the frequency of each category (Figure 4.4). Applying the Pearson chi-squared test to the proof distribution, we obtained a p-value of 6.75^{-57}, indicating that the proofs generated by our protocol are uniformly distributed. This implies that the generated proofs have high entropy.

Having proofs of high entropy is important for the security of the protocol because it means that no proof is more likely to occur than others. Furthermore, this incentivizes miners to allocate more data for the initialization process in order to obtain more diverse proofs.

4.4.7 Throughput

To measure the throughput of our proposed blockchain compared with Chia's we ran private a simulation using our blockchain prototype. This simulation had 25 nodes operating as miners, alongside 1 Timelord, a specialized node responsible for supplying PoT proofs. In the context of our simulation, introducing multiple Timelords did not yield any discernible alteration in the blockchain's performance. This phenomenon is attributable to the fact that, within our system and in Chia's, only the first PoT received by each miner is utilized. Consequently, any additional proofs furnished by other Timelords are disregarded, having no impact on the overall performance.

Conducting assessments with a substantial miner count necessitates an extensive storage capacity, an impractical feat in our experimental context. Consequently, in our simulation, each miner possesses

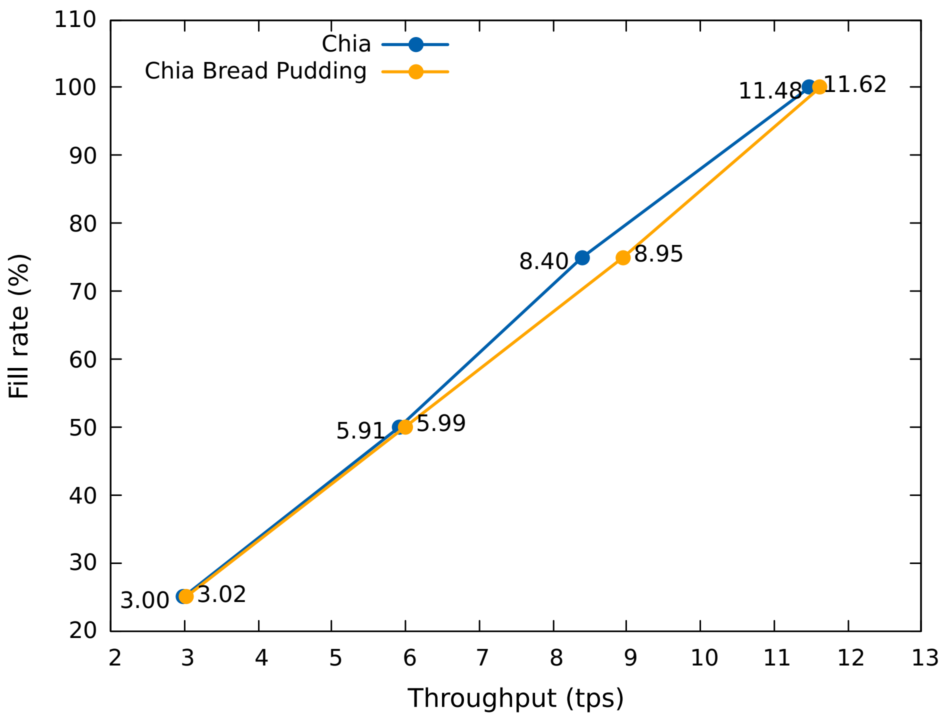

Figure 4.5: Throughput for different fill rates.

an individual proof exclusively designated for the generation of all blocks.

In our workload scenario, synthetic transactions were employed, with each node generating a specific quantity of these transactions, all slated for inclusion in each block. We conducted numerous simulations, each lasting an hour, with variations in the transaction volume. Notably, Chia's protocol prescribes a maximum limit of 1010 transactions per block. For our evaluation, we introduce the concept of "fill rate", representing the proportion of transactions within a block relative to this predetermined upper limit (i.e., 1010 transactions).

In Figure 4.5, we can observe the transactions per second achieved at various fill rates: 25%, 50%, 75%, and 100%. Interestingly, the throughputs attained in both Chia and our protocol exhibit remarkable similarity. This outcome aligns with our expectations, given that the Timelord predominantly governs the throughput of both protocols. The Timelord finalizes each block generated by miners with a PoT proof, a process that consumes a fixed amount of time for generation. Furthermore, the periodic release of challenges to miners by the Timelord ensures that the overall system's throughput remains relatively constant. Consequently, the observed throughput figures closely mirror each other in both Chia and our protocol.

4.5 Discussion

In this section, we discuss the results of our evaluations by answering our RQs.

Answer to Q1: To what extent can we replace Proof-of-Space blockchain's random data with useful data? Our findings affirm that replacing random data within PoS blockchains with useful data is indeed feasible. However, the extent of this substitution is circumscribed by the metadata generated during the initialization of the useful data. These metadata serve dual functions: facilitating the recovery of the original data and verifying the allocated space.

In our protocol, the magnitude of the metadata is intrinsically linked to the designated fanout value. At a minimum, our framework permits a 50% ratio of useful data. Notably, this ratio can be amplified by opting for higher fanout values, as elucidated in Figure 4.1. It is imperative to recognize that augmenting the fanout also induces an enlargement in proof size. Therefore, the fine-tuning of this parameter necessitates judicious consideration to maintain these metrics within practical bounds.

Answer to Q2: What is the impact of repurposing the data used in Proof-of-Space blockchains? The utilization of useful data for the creation of proofs enables miners with limited storage capacity to actively participate in the blockchain, thereby expanding the pool of miners and fostering greater decentralization within the system. Moreover, by repurposing data in PoSp blockchains, we mitigate the inherent storage resource waste associated with these systems. Nonetheless, it is worth noting that the size of PoRep proofs often surpasses that of practical PoSp proofs. This increase in proof size can lead to longer block transmission times within the system and greater blockchain storage requirements. However, it is important to emphasize that the public parameters of our protocol can be fine-tuned to mitigate impractical block transmission times and storage overhead. In practical terms, the increase in proof size remains marginal when compared with the Chia block size.

4.5.1 Threats to validity

We recognize the following threats to the validity of our evaluation:

1. potential implementation errors of the protocol that could lead to incorrect results and analysis;
2. potential inaccurate approximations of the VDE implementation, affecting the simulated initialization and proof verification times;
3. the choice of the number of nodes used in the simulation, impacting the system's performance within the experimental setup;
4. differences in real-world user behavior or data usage that might not be fully captured by the synthetic transaction generation, leading to discrepancies between the simulation and actual performance.

Despite these potential threats to validity, our comprehensive analysis and experimental results have laid a solid groundwork for the reliability of our findings. The research conducted in this evaluation chapter not only highlights the latent storage potential in PoSp systems but also underscores the significance of repurposing miner data for blockchain participation, opening up promising avenues for future exploration in this field.

5 Conclusion

Contents

5.1 Conclusions

In PoSp blockchains, miners fill their storage space by generating cryptographic data randomly, solely for the purpose of ensuring blockchain security. These techniques result in storage wastage, as the stored data serves no purpose for the miner other than generating cryptographic proofs.

This work proposes an approach to reduce the storage waste in PoSp blockchains by utilizing local data (such as user files) to generate proofs. The solution is based on the Chia protocol and adapts it by replacing random data in the proofs with useful data.

Our results show that at least 50% of the random data in PoSp proofs can be replaced with useful data. Our proposed solution does not compromise the performance of the system but fosters greater decentralization within it by expanding the pool of miners who are able to participate in it.

5.2 Future Work

It is crucial to have a formal security analysis of our protocol. While our protocol deviates from the complete adherence to PoRep algorithms, we have tailored the encoding method to accommodate specific performance requirements. In our forthcoming research, we plan to conduct an in-depth examination of the dependency graph that emerges during the encoding process, aiming to establish the proof of correctness for Chia Bread Pudding.

Furthermore, our focus will encompass an expansion of the encryption procedure during the initialization phase, incorporating the encryption capabilities offered by contemporary storage filesystems. This proposed extension holds the potential to enhance the protocol's overall efficiency, considering that modern file systems already integrate data encryption capabilities that our protocol can capitalize on.

Bibliography

[1] B. Cohen and K. Pietrzak, “The chia network blockchain,” *vol*, vol. 1, pp. 1–44, 2019.

[2] O. Ali, M. Ally, Y. Dwivedi *et al.*, “The state of play of blockchain technology in the financial services sector: A systematic literature review,” *International Journal of Information Management*, vol. 54, p. 102199, 2020.

[3] P. Dutta, T.-M. Choi, S. Somani, and R. Butala, “Blockchain technology in supply chain operations: Applications, challenges and research opportunities,” *Transportation research part e: Logistics and transportation review*, vol. 142, p. 102067, 2020.

[4] Y. Liu, D. He, M. S. Obaidat, N. Kumar, M. K. Khan, and K.-K. R. Choo, “Blockchain-based identity management systems: A review,” *Journal of network and computer applications*, vol. 166, p. 102731, 2020.

[5] S. Nakamoto, “Bitcoin: A peer-to-peer electronic cash system,” *Decentralized Business Review*, p. 21260, 2008.

[6] “The merge,” 2022. [Online]. Available: https://ethereum.org/en/upgrades/merge/

[7] J. R. Douceur, “The sybil attack,” in *Peer-to-Peer Systems: First InternationalWorkshop, IPTPS 2002 Cambridge, MA, USA, March 7–8, 2002 Revised Papers 1*. Springer, 2002, pp. 251–260.

[8] J. Garay, A. Kiayias, and N. Leonardos, “The bitcoin backbone protocol: Analysis and applications,” in *Annual international conference on the theory and applications of cryptographic techniques*. Springer, 2015, pp. 281–310.

[9] K. J. O’Dwyer and D. Malone, “Bitcoin mining and its energy footprint,” 2014.

[10] E. Kapengut and B. Mizrach, “An event study of the ethereum transition to proof-of-stake,” *Commodities*, vol. 2, no. 2, pp. 96–110, 2023. [Online]. Available: https://www.mdpi.com/2813-2432/2/2/6

[11] J. Benet and N. Greco, “Filecoin: A decentralized storage network,” *Protoc. Labs*, pp. 1–36, 2018.

[12] J. Wagstaff. [Online]. Available: https://subspace.network/technology

[13] 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*, 2007, pp. 598–609.

[14] H. Shacham and B. Waters, "Compact proofs of retrievability," in *International conference on the theory and application of cryptology and information security*. Springer, 2008, pp. 90–107.

[15] 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*, 2007, pp. 584–597.

[16] K. D. Bowers, A. Juels, and A. Oprea, "Proofs of retrievability: Theory and implementation," in *Proceedings of the 2009 ACM workshop on Cloud computing security*, 2009, pp. 43–54.

[17] B. Fisch, "Poreps: Proofs of space on useful data," *Cryptology ePrint Archive*, 2018.

[18] M. Castro and B. Liskov, "Practical byzantine fault tolerance," in *Proceedings of the Third Symposium on Operating Systems Design and Implementation*, ser. OSDI '99. Berkeley, CA, USA: USENIX Association, 1999, pp. 173–186. [Online]. Available: http://dl.acm.org/citation.cfm?id=296806.296824

[19] A. Kiayias and G. Panagiotakos, "Speed-security tradeoffs in blockchain protocols," *Cryptology ePrint Archive*, 2015.

[20] C. Dwork and M. Naor, "Pricing via processing or combatting junk mail," in *Annual international cryptology conference*. Springer, 1992, pp. 139–147.

[21] A. Jules and J. Brainard, "Client-puzzles: a cryptographic defense against connection depletion," in *Proc. Network and Distributed System Security Symp.(NDSS'99)*, 1999, pp. 151–165.

[22] M. Jakobsson and A. Juels, "Proofs of work and bread pudding protocols," in *Secure information networks*. Springer, 1999, pp. 258–272.

[23] A. Back *et al.*, "Hashcash-a denial of service counter-measure," 2002.

[24] I. Eyal and E. G. Sirer, "Majority is not enough: Bitcoin mining is vulnerable," *Communications of the ACM*, vol. 61, no. 7, pp. 95–102, 2018.

[25] S. Valfells and J. H. Egilsson, "Minting money with megawatts [point of view]," *Proceedings of the IEEE*, vol. 104, no. 9, pp. 1674–1678, 2016.

[26] U. Gallersdörfer, L. Klaaßen, and C. Stoll, "Energy consumption of cryptocurrencies beyond bitcoin," *Joule*, vol. 4, no. 9, pp. 1843–1846, 2020.

[27] "Bitcoin energy consumption index," Apr 2022. [Online]. Available: https://digiconomist.net/bitcoin-energy-consumption

[28] U. "cunicula" and M. Rosenfeld, Aug 2011. [Online]. Available: https://bitcointalk.org/index.php?topic=37194.0

[29] I. Bentov, C. Lee, A. Mizrahi, and M. Rosenfeld, "Proof of activity: Extending bitcoin's proof of work via proof of stake [extended abstract] y," *ACM SIGMETRICS Performance Evaluation Review*, vol. 42, no. 3, pp. 34–37, 2014.

[30] I. Bentov, A. Gabizon, and A. Mizrahi, "Cryptocurrencies without proof of work," in *International conference on financial cryptography and data security*. Springer, 2016, pp. 142–157.

[31] S. King and S. Nadal, "Peercoin — the pioneer of proof-of-stake," 2012. [Online]. Available: https://www.peercoin.net/read/papers/peercoin-paper.pdf

[32] A. Poelstra *et al.*, "Distributed consensus from proof of stake is impossible," *Self-published Paper*, 2014.

[33] P. Daian, R. Pass, and E. Shi, "Snow white: Robustly reconfigurable consensus and applications to provably secure proof of stake," in *International Conference on Financial Cryptography and Data Security*. Springer, 2019, pp. 23–41.

[34] A. Kiayias, A. Russell, B. David, and R. Oliynykov, "Ouroboros: A provably secure proof-of-stake blockchain protocol," in *Annual international cryptology conference*. Springer, 2017, pp. 357–388.

[35] Y. Gilad, R. Hemo, S. Micali, G. Vlachos, and N. Zeldovich, "Algorand: Scaling byzantine agreements for cryptocurrencies," in *Proceedings of the 26th symposium on operating systems principles*, 2017, pp. 51–68.

[36] Y. Huang, J. Tang, Q. Cong, A. Lim, and J. Xu, "Do the rich get richer? fairness analysis for blockchain incentives," in *Proceedings of the 2021 International Conference on Management of Data*, 2021, pp. 790–803.

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

[38] 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," in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2017, pp. 357–379.

[39] M. Hellman, "A cryptanalytic time-memory trade-off," *IEEE transactions on Information Theory*, vol. 26, no. 4, pp. 401–406, 1980.

[40] S. Park, A. Kwon, G. Fuchsbauer, P. Gaži, J. Alwen, and K. Pietrzak, “Spacemint: A cryptocurrency based on proofs of space,” *Cryptology ePrint Archive*, 2015.

[41] K. Pietrzak, “Simple verifiable delay functions,” in *10th innovations in theoretical computer science conference (itcs 2019)*. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2018.

[42] B. Wesolowski, “Efficient verifiable delay functions,” in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2019, pp. 379–407.

[43] T. Moran and I. Orlov, “Simple proofs of space-time and rational proofs of storage,” in *Annual International Cryptology Conference*. Springer, 2019, pp. 381–409.

[44] R. L. Rivest and A. Shamir, “Payword and micromint: Two simple micropayment schemes,” in *International workshop on security protocols*. Springer, 1996, pp. 69–87.

[45] M. Ball, A. Rosen, M. Sabin, and P. N. Vasudevan, “Proofs of useful work,” *Cryptology ePrint Archive*, 2017.

[46] S. King, “Primecoin: Cryptocurrency with prime number proof-of-work,” *July 7th*, vol. 1, no. 6, 2013.

[47] A. Miller, A. Juels, E. Shi, B. Parno, and J. Katz, “Permacoin: Repurposing bitcoin work for data preservation,” in *2014 IEEE Symposium on Security and Privacy*. IEEE, 2014, pp. 475–490.

[48] R. Gennaro, C. Gentry, B. Parno, and M. Raykova, “Quadratic span programs and succinct nizks without pcps,” in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2013, pp. 626–645.

[49] N. Bitansky, A. Chiesa, Y. Ishai, O. Paneth, and R. Ostrovsky, “Succinct non-interactive arguments via linear interactive proofs,” in *Theory of Cryptography Conference*. Springer, 2013, pp. 315–333.

[50] E. Ben-Sasson, A. Chiesa, D. Genkin, E. Tromer, and M. Virza, “Snarks for c: Verifying program executions succinctly and in zero knowledge,” in *Annual cryptology conference*. Springer, 2013, pp. 90–108.

[51] D. Karger, E. Lehman, T. Leighton, R. Panigrahy, M. Levine, and D. Lewin, “Consistent hashing and random trees: Distributed caching protocols for relieving hot spots on the world wide web,” in *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, 1997, pp. 654–663.

[52] L. Chen, L. Xu, N. Shah, Z. Gao, Y. Lu, and W. Shi, “On security analysis of proof-of-elapsed-time (poet),” in *International Symposium on Stabilization, Safety, and Security of Distributed Systems*. Springer, 2017, pp. 282–297.

[53] C. Policroniades and I. Pratt, “Alternatives for detecting redundancy in storage systems data,” in *2004 USENIX Annual Technical Conference (USENIX ATC 04)*. Boston, MA: USENIX Association, Jun. 2004. [Online]. Available: https://www.usenix.org/conference/2004-usenix-annual-technical-conference/alternatives-detecting-redundancy-storage-systems

[54] M. Bellare, J. Kilian, and P. Rogaway, “The security of cipher block chaining,” in *Annual International Cryptology Conference*. Springer, 1994, pp. 341–358.

[55] D. Catalano and D. Fiore, “Vector commitments and their applications,” in *Public-Key Cryptography – PKC 2013*, K. Kurosawa and G. Hanaoka, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 55–72.

[56] M. Szydlo, “Merkle tree traversal in log space and time,” in *Advances in Cryptology-EUROCRYPT 2004: International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004. Proceedings 23*. Springer, 2004, pp. 541–554.

[57] K. Fu, M. F. Kaashoek, and D. Mazieres, “Fast and secure distributed read-only file system,” *ACM Transactions on Computer Systems (TOCS)*, vol. 20, no. 1, pp. 1–24, 2002.

[58] Y. Wang and Y. Zheng, “Fast and secure magnetic worm storage systems,” in *Second IEEE International Security in Storage Workshop*, 2003, pp. 11–11.

[59] R. Sion, “Strong worm,” in *2008 The 28th International Conference on Distributed Computing Systems*, 2008, pp. 69–76.

[60] Chia-Network, “Chia-network/chiapos: Chia proof of space library.” [Online]. Available: https://github.com/Chia-Network/chiapos

[61] J. Paulo, P. Reis, J. Pereira, and A. Sousa, “Dedisbench: A benchmark for deduplicated storage systems,” in *OTM Confederated International Conferences” On the Move to Meaningful Internet Systems”*. Springer, 2012, pp. 584–601.

[62] Openssl, “Openssl/openssl: Tls/ssl and crypto library.” [Online]. Available: https://github.com/openssl/openssl.git

[63] A. K. Lenstra and B. Wesolowski, “A random zoo: sloth, unicorn, and trx,” *Cryptology ePrint Archive*, 2015.

[64] Randomchain, “Randomchain/pysloth: Python wrapper around the sloth algorithm by benjamin wesolowski in a random zoo: Sloth, unicorn, and trx.” [Online]. Available: https://github.com/randomchain/pysloth

[65] Chia-Network, “Chia-network/chia-blockchain: Chia blockchain python implementation (full node, farmer, harvester, timelord, and wallet).” [Online]. Available: https://github.com/Chia-Network/chia-blockchain.git

[66] Pybind, “Pybind/pybind11: Seamless operability between c++11 and python.” [Online]. Available: https://github.com/pybind/pybind11.git