SparseChain: Sparse Distributed Blockchain Storage Protocol

Kahina Khacef

SparseChain: Sparse Distributed Blockchain Storage Protocol

Kahina Khacef* Hadrien De March†

May 24, 2022

Abstract

Traditional blockchains such as Bitcoin and Ethereum today contain hundreds of gigabytes of data, and their transactions will only continue to grow in size over time. This paper focuses on block storage scalability. We design a new consensus algorithm that rewards nodes for storing historical data and incentivizes them to store a suitable amount of data. Unlike Arweave, our approach provides efficient storage features making it convenient for lightweight peers. Indeed, the consensus distributes specific blocks to store to miners according to their hash rate and rewards them for the storage with Proof of Random Access. In addition, we provide theoretical results that guarantee the immutability and permanent availability of the data.

Key words. Permissionless Blockchain, Decentralized, Security, Availability, Proof of Random Access.

1 Introduction

Permissionless blockchains, also known as distributed public ledgers, were first proposed as a technical solution [9] for deploying the Bitcoin digital currency and payment system. Permissionless blockchains strive to be a persistent, distributed, consistent, and ever-growing transaction log publicly auditable by anyone who can join or leave the network without permission. Those blockchains work along with two components of consensus to ensure security and consistency: The full blockchain replication that each node records make data tampering impossible and guarantee data tamper-evident, adding on top of that, solving the Proof_of_Work (PoW), this cryptography puzzle is demonstrably secure against a large proportion of participants who wish to disrupt the system and allows a configurable and independent rate of block creation

*LIP6, Śorbonne Universite, kahina.khacef@lip6.fr.
†École Polytechnique, hadrien.de-march@polytechnique.org.

regardless of the size of the system. About the last one, the broadcast primitive relies on Peer_to_Peer (P2P) network properties, without the presence of a trusted third party, despite the malicious behavior of specific nodes. Proof_of_X systems summarize other consensus techniques utilized in blockchain protocols; for example, Ethereum blockchain technology evolves and moves toward a less computationally intensive design. Ethereum [13] popularised the concept of smart contract to implement any application on top of the blockchain or meant actual word certification to token validated by the blockchain. Furthermore, the consensus aims to provide finality, which means that no value inserted into the ledger can ever be withdrawn. Once a value is registered, it is hard to change it. Security is the responsibility of the nodes that maintain the blockchain, the more nodes store the blockchain, the more secure it is. The replication of the blockchain across all nodes that constitute the network is a hugely inefficient use of the memory storage of the system because the node in a blockchain system (such as the Bitcoin system) does not rely on a central organization, and each node stores a full copy of the transactions. This aspect, however, indicates that the size of blockchain transactions is overgrowing. Therefore, the node memory needs to be expanded to support the system running with continuous system operations. One of the issues with blockchain scalability is the growth in storage costs as the number of transactions increases, necessitating substantial memory capacity to store the blockchain and which excludes small nodes from the validation process. The accounting/balance models used in blockchain (UTXO model used in Bitcoin and the account model used in Ethereum) need to be stored for validation, called the validation state. Traditional blockchain maintains a continuously-growing history of ordered information to achieve consensus and provide security to the blockchain. Full nodes replicate all previous validation states to prevent double-spending, used as proof of correct status to keep the system functioning. Full nodes validate new transactions by checking their states (recorded transactions) and then storing them, which requires a lot of storage space. Ethereum does not record all transactions but instead keeps track of the sequence number ("nonce") of the most recent transaction issued from a specific account [3]. Even if the account has no balance, this nonce must be saved and causes storage costs to grow linearly and unintentionally caused Ethereum issues when a smart contract establishes several zero-balance accounts. Furthermore, designing a cryptocurrency with storage costs that scale well with the number of users and transactions is difficult due to various constraints. A long line of research proposes various techniques to make blockchain more scalable. For example, On_chain approaches, i.e., sharding nodes into multiple subsets [7, 8, 15], and Off_chain approaches, i.e., Lightning Network [5]. Partitioning data into separate shards managed by different subsets of nodes reduces performance as more messages are exchanged to build consensus without improving robustness; in such an approach, the data grows linearly with the number of nodes and transactions, the Off_chain scalability solutions can introduce their security threats to the blockchain because the data is not stored directly on the blockchain but rather through third-party protocols. These techniques provide scalability but affect decentralization and result in security vulnerabilities.

The solutions to achieve scalability must not compromise the decentralization and security of

blockchain networks. Therefore, the ability to scale a blockchain lies primarily in improving the foundation of blockchain technology, preventing the hardware shortages triggered by classical Proof_of_Access cryptocurrencies. This paper presents a new blockchain design to address the storage bottleneck. We suggest a storage sharding of the blockchain with sparse distribution over the nodes instead of a full replication to reduce the impact of the ever-growing state in blockchain while providing robustness through data replication.

The main contributions of this paper are summarized as follows:

- The design of the storage sharding: distributing the block storage in a sparse way over nodes while maintaining security of the blockchain. The proposed system is named as SparseChain.
- Algorithm for a smart live adaptation of the storage sharding scheme.
- Method for guaranteeing a positive behaviour of nodes, and perpetual availability of all blocks.

The remainder of this paper is organized as follows. Section 2 presents the motivation, Section 3 discusses the related work, Section 4 provides background on blockchains. Section 5 presents an overview of SparseChain. Section 6 describe the storage sharding mechanism which enhances the scalability of the blockchain, section 7 discusses the protocol economics, and section 8 discusses the parameters for the protocol.

2 Motivation

Each transaction is replicated redundantly across multiple storage nodes. Therefore, the classic blockchain's scalable storage bottleneck is primarily due to the concerns listed below.

- *Storage capacity per full node*: Every full node in the traditional blockchain stores and processes all states and transactions. A full node is a node in a blockchain network that can independently verify all transactions and the current state of the network. This process enhances security and maintains traceability, but limits the scalability because the volume of data that each node must store will continue to grow. The nodes must maintain all states and supply extra hardware and memory capacity to store the massive volumes of data for the blockchain to continue functioning. Off_chain storage solutions, such as IPFS or Arweare [11], allow for scalability but accentuate the centralization of the blockchain because a user must obtain permission and request to access the blockchain history stored on Off_chain.
- *Cost of using blockchain network*: As the number of users and transactions on the blockchain network increases, nodes need to process and store more data with higher fees to maximize their earnings. Since All transaction requests require fees, Miners prioritize transactions that pay higher fees. Therefore, if transactions need to be verified quickly, the user must pay higher fees to get priority. The more a user agrees to pay

high transaction fees, the faster they will be processed. The Bitcoin Cash protocol [1] helps reduce transaction fees and improves transaction speed. It rejects the block size limitation introduced by Bitcoin (limited to 1MB). Blocks in the Bitcoin cash chain correspond to a limit of 32 MB, or approximately 250 transactions per second (compared to 24,000 transactions per second for Visa). As the blocks of the Bitcoin cash are large enough to allow miners to mine all transactions, there is no need to pay transaction fees for the transaction to be mined on the next block. Although, Bitcoin cash raises concerns regarding the difficulty of hosting a full node.

We propose algorithms to reduce blockchain replication without affecting blockchain security and decentralization while allowing scalability. Instead of storing the whole blockchain on each full node, the overall state will only be replicated on a subset of the nodes, and we increase the block size to include more transactions. As a result, the blockchain continues to receive transactions and improves the number of transactions performed by nodes per second.

3 Related Work

A long line of research proposes to make the blockchain more scalable. Executing transactions Off_chain overcoming the scalability limitations of blockchains, i.e., untrusted peers can build direct payment channels on the Lightning network, allowing them to make Off_chain micropayments without committing each transaction to the underlying blockchain. A payment channel is made up of a blockchain-based contract that stores the funds of the peers involved in the transaction.

Sharding blockchain is the closest work to ours. Elastico is the first public sharding blockchain that tolerates byzantine adversaries [8]. Elastico divides the network into shards and assures probabilistic correctness by allocating nodes to committees randomly, with each shard is verified in parallel by a separate committee of nodes. It uses costly PoW to construct committees, with nodes joining committees randomly and running PBFT (Practical Byzantine Fault Tolerance) for intra-committee consensus. Omniledger [7], unlike Elastico, proposes novel methods for assigning nodes to shards with a higher security guarantee and employs both PoW and BFT. In addition, it uses an atomic protocol for across-shard transactions (Atomix) to achieve global synchronization of transactions. The intra-shard consensus protocol uses a variation of ByzCoin [6]. It assumes partially synchronous channels to achieve speedier transactions, resilient against a weakly dynamic adversary that corrupts up to 25% faulty nodes in each committee and 33% malicious nodes tolerated by the network.

OmniLedger restricts the responsibility of verifiers to prevent double-spending attacks. These proposals appear tempting because they lower the costs of storage, bandwidth, and latency, allowing the system to scale throughput arbitrarily. But unfortunately, attackers that control a small portion of the currency can manipulate the systems. Shard replication factors fall as shard size decreases, and as a result, transactions in a specific shard are confirmed by a small number of clients. In addition, an adversary may own a key fraction of a shard's stake,

allowing it to control the entire shard. As a result, these systems necessitate an unfavorable trade-off between scaling and security, limiting the amount of sharding that can be done.

Rapidchain [15] also supports cross-shard transactions using Byzantine consensus protocols but requires strong synchronous communication among shards which is hard to achieve with resilience up to 33% and 50% of committee resiliency. In sharding-based systems, database performance scales linearly with the number of nodes, necessitating the creation of complicated protocols to enable shard connectivity.

Mina [2] developed by O(1) labs is considered the smallest blockchain. It integrates zero-knowledge proofs ("succinct non-interactive argument of knowledge" or "Zk-SNARKs") to validate transactions, which were first used by the Zcash cryptocurrency [1]. The protocol generates a proof at each step to validate a new transaction without consulting the register of all the previous transactions; this proof drastically reduces the size of the blockchain, which is never more than 22 KB. Mina stands out for data privacy. It allows users to use the blockchain while maintaining control over their private information. The objective is to connect to other participants in a secure manner and without revealing any personal data. Although, block producer in Mina actually do store the full state (a block producer must have the current state of the blockchain), while block validators need not store anything. So Mina still has a storage problem.

In Arweave [11], authors propose a blockchain that provides permanent storage by incentive. Numerous blockchains (i.g., Solana [14]) that achieves scalability use Arweave for storing transactions and only keep a few blocks on their On_chain. Solana is a high-throughput blockchain that uses a network timestamp technique called Proof_of_History to achieve sub-second block times and high throughput. If Solana operates at maximum capacity for a year, it is expected to generate 4 petabytes of data. At all times, the full blockchain must be stored and accessible. Arweave introduces a new block structure called "blockweaves" related to two previous blocks, i.e., in addition to a pointer to the last block in the chain, blocks point to another randomly chosen "recall block" of the prior history of the blockchain. When miners solve the problem and find an appropriate hash they can share the new block and recall block with the network. Arweave is constantly growing. Although its blockchain is designed to be archived forever, few nodes are incentivized to store old chain data. This problem gets worse as the blockchain network grows and has a storage problem.

4 Background

4.1 Incentive mechanism in distributed system

Transaction in Bitcoin uses The Unspent Transaction Outputs (UTXO) data model, which can have a single or multiple inputs and outputs. Permissionless blockchains allow any node to join or leave the P2P network without authentication. When a new transaction is broadcast over the network, it is received and verified by a group of nodes to ensure that it is correctly signed and has not been previously recorded in the blockchain, after which they are grouped into valid

transaction blocks by miners competing and executing a consensus to extend the longest chain with blocks they generate.

Traditional blockchain has robust incentive mechanisms to encourage nodes to maintain the network running economically. Miners are rewarded for their labor by getting a fee each time their proposed block is accepted. Besides Bitcoin, file sharing is the most common use of P2P technologies such as BitTorrent. Nodes in Bittorent participate in a game called the 'optimistic tit-for-tat algorithm' [4]. In this game, nodes share data reciprocally with other nodes that share data with them. Network participants use information about favors to calculate peer rankings. The participants then use these rankings to determine how they will share their resources with other network participants preferring those who have higher rankings. Occasionally, nodes share data at random, interacting with each other without taking peer rankings into account. This game leads to a *Nash equilibrium* where all nodes are incentivized to share resources (data) freely at the network's maximum capacity and perform prosocial behaviors. Bitcoin acts similarly to BitTorrent [10]in that it is made up of computers and servers that host full nodes through a P2P network, making it resistant to attacks.

4.2 Arweave

Each node in the Arweave network ranks its peers similarly to Bittorrent. Node rates its peers based on two main criteria, first, the peer's generosity - sending new transactions and blocks - and second, the peer's responsiveness - responding quickly to information requests. Instead, it allows each actor to maintain private, local scores for other peers. Nodes are incentivised through the Adaptive Interacting Incentive Agent (AIIA) meta-game, i.e., nodes with low AIIA social rank risk their messages (including new candidate blocks) being propagated too slowly to be accepted by the network. It mechanism incentivizes nodes to keep data, the miner must include the recall block in the new block, otherwise, it won't be able to produce a new block [12]. The recall block is one of the past historic blocks in the chain of nodes; including this block to mine a new block is proof that the miner keeps data. Transactions of the recall block are hashed alongside those found in the current block to generate the next block. Once mined, the miner sends the new block with the recall block to be verified by the network, even if validators do not have their copy of the recall block. Miners have an incentive to store vast amounts of data to enhance their chances of finding a good recall block, because the selection of recall blocks is random. Rewarding in Arweave is used (i) to encode data into the system and (ii) to reward miners.

Arweave introduces a synchronization block generated once every 12 blocks, containing a full list of the balance of every wallet in the system and a hash of every previous block without the transaction. Synchronization blocks help new participants in the network to bootstrap, and there is no need to download all the last blocks from the genesis block. Instead, when the miner joins the Arweave network, they will download each previous block from the current block to the last synchronization block.

5 Overview

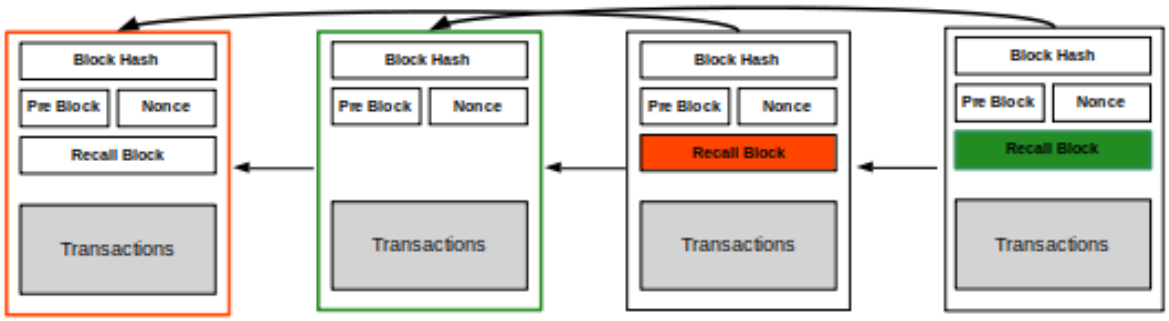

Figure 1: An overview of the block data structure illustrates the link to both the previous block and recall block.

In a nutshell, Sparsechain is a permissionless, Proof_of_Work combined with sparse Proof_of_Access inspired by Arweave, On_chain P2P cryptocurrency that (i) significantly reduce the storage load by minimizing the replication of each block and (ii) ensure availability of all transactions.

5.1 Objectives

The SparseChain system achieve the following main goal:

1. *Efficient Storage*: Distribute the blockchain's storage costs across different network nodes by sharding without sacrificing security. As a result, the node can store a small amount of data while participating in the maintenance of the blockchain.
2. *Availability*: Due to an incentive mechanism for nodes to keep blocks in blockchains, all blockchain transactions are guaranteed to be available (even old blocks). It allows the verification of historical transactions and prevents their falsification.
3. *Transaction per second*: The number of transactions per second should be increased. By freeing up local storage, the system lowers the cost of holding the blockchain; in fact, the block size can be increased to allow more transactions without overburdening the network.

SparseChain rewards the storage of the history of the blockchain through the very consensus algorithm. As for Arweave, the only way to reward storage without giving the possibility for an actor to create multiple nodes in order to get all the storage reward is to intricate the Proof of Random Access with mining through the use of a recall block included in the new blocks for increasing rewards (through decreasing the mining difficulty in the case of using a recall block).

Therefore, there cannot be pure storage nodes that get rewarded for this; the miners need to be the ones in charge of the storage by the protocol.

5.2 Architecture of SparseChain

The most popular blockchains, such as Bitcoin and Ethereum, employ a full replication process that makes it impossible to do big blocks as it would become challenging for miners to store them all. To address this problem, we present SparseChain, a storage sharding protocol that allows scalable storage and provides permanent availability of data by incentive. It combine a block structure, an incentive model, and a consensus rule. (2).
Sharding allows nodes to deal with large ledger sizes. Consider a system with 100 billion transactions and 1 million online nodes as an extreme example. We then have 33 thousand blocks, and each block is held by about 250 nodes by Proposition 7.1, enabling a high degree of availability, the possibility for lightweight nodes to fully participate into the protocol, and a division by 135 of the storage space needed among the network. A group of miners is rewarded for mining blocks as well as for proving that they are storing a subset of the blockchain defined by the algorithms. This subset is designed to have a reasonable size, in order to allow small miners to participate in the process. Nodes are incentivized to store blocks since they increase their chances of mining a new block and receiving rewards.

In order to allow the nodes to validate transactions while only storing the 12 latest blocks, the balance state of each wallet is summarized in a synchronize block that appears in the chain every 12 blocks (see Subsection 4.2). We provide a new data structure, The block is cryptographically related to the last block in the chain, and to an Arweave-like recall block, i.e., a previous block in the chain's history. However, unlike Arweave, the potential candidates to be a recall block are chosen thanks to our novel algorithm described in Section 6 to make the scalability possible through sparsity (see Figure 1).

6 Storage Sharding Protocol

In this section, we describe the algorithm for storing of the blocks that preserves the security and scalability in the blockchain.

6.1 System Model

We adopt an unspent transactions output (UTXO) model. A public key PKI distinguishes a UTXO and its amount related. Furthermore, each public key is linked to the digital signature schema with the uniqueness attribute, allowing users to utilize the public keys (or a hash thereof) of their UTXOs as a reference to them. A user can own many UTXOs at any same time. UTXOs can only be debited once, and once debited, they are no longer valid.

We provide a blockchain protocol that enables storage scalability by reducing block replication across the whole network and ensures security against attacks using the hash link properties.

6.2 Block structure

A block is a data structure that contains valid transactions. When transactions enter the network, they are initially verified. Once validated, the transactions are collected in a pool until they are included in a block and confirmed by the network. Transactions that have been mined by a node and confirmed by the network are withdrawn from the pool.

A block B contains a set of transactions txs, a previous block hash, a nonce that a miner of that block found to calculate the valid block, a timestamp, and the recall block (i.e., the entire contents of the recall block hashed with the previous block hash and current transactions) The transactions are encoded in a Merkle tree (i.e., the Merkle tree is a data structure used for efficiently and securely data storage). The blocks are ordered chronologically. Each block is linked to both previously added block and the recall block in the chain, forming the blockchain.

Formally the block B_i, $i \geq 1$, can be considered to be a quintuplet $(h_i, txs_i, nonce, recallblock|\emptyset)$. h_i is the hash to the previous block, tx_i is the set of transactions, the *nonce* that miner must discover before solving for a block, as it is appended to a hashed block that, when rehashed, meets the difficulty level restrictions, and a *recallblock*

SparseChain encourages nodes to store data; differently from Arweave, nodes are rewarded for mining blocks with a recall block or without the recall block. This method allows all network nodes to participate in mining. The difference between the two types of mines is that miners who contain recall blocks have a high chance of finding the correct nonce and calculating the new block.

Definition 6.1. *A block is a data structure that includes a header, valid transactions, and a recall block. The header contains the previous hash block, timestamp, nonce, Merkle root, and difficulty. A hash function $H(z)$ converts a random data input z into a fixed-length string of bytes, a transaction's hash makes it simple to identify it on the blockchain.*

6.3 Block validity rules

Every block must follow two rules to be considered valid:

6.3.1 Mining

A miner who demonstrates that he has a recall block must check the requirement that the block hash is lower than the threshold Θ_{recall} in order to mine a new block, the hash begins with a number of zero bits. Validating a new block includes verifying this proof. Furthermore, a miner who does not have access to the recall block must ensure that the block hash is lower than Θ_0, with $\Theta_0 < \Theta_{\text{recall}}$, as chows in Figure 2. Using the recall block is considered as proof that the miner stores blocks. Demonstrating whether or not the miner has access to the recall block is part of the block's construction.

PoW difficulty is determined by the average number of blocks mined each hour. The difficulty increases if the block is generated too quickly.

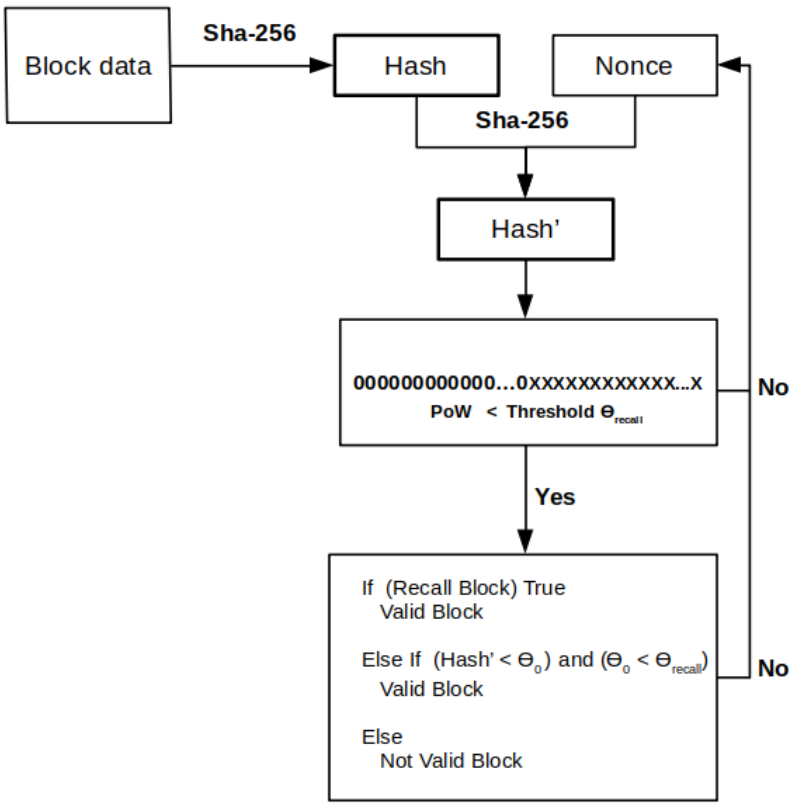

Figure 2: An illustration of mining block in SparseChain

SparseChain protocol incites storage because miners who have access to recall blocks of the blockchain history to mine new blocks have a higher probability of winning. The protocol incentivizes miners to keep data available and ensure security, while reducing block replication.

In order to make sure that storage stays rewarded enough, Θ_{recall} is adjusted in order to have 50% of validating hashes $\leq \Theta_0$. It then also ensures that the new miners that have never mined blocks can never have less than 50% of disadvantage while mining. Thus leaving them a good chance to validate blocks and get storage rewards.

6.3.2 Recall block validity

The goal of SparseChain is to give an incentive to miners to store the history of the blockchain. For this we use the same concept of recall block than Arweave. The difference is that the protocol we define gives a set of recall blocks among which miners can pick. These blocks are given by (1) in Definition 6.2.

Another difference is that each miner gets a selected set of valid recall blocks to store so that, unlike for Arweave, huge miners are not twice advantaged because they have a large computation and a large storage capacity. SparseChain protocol is made so that the number of potential recall blocks for a miner is proportional to its computing power. These blocks are

given by (2) in Definition 6.2.

Let $\lambda > 0$, $0 \leq \alpha \leq 1$, and $1 \leq k_{min} \leq H$ be three parameters depending on the size of H from which we will discuss the value in Section 8.

Definition 6.2. A block B_j for $1 \leq j \leq H$ is a valid recall block for mining if and only if:

$$\frac{Hash_1\left(hash_{B_H}, hash_{B_j}\right)}{hashmax} \leq \frac{\lambda}{\alpha H}. \tag{1}$$

and

$$\frac{Hash_2\left(hash_{B_k}, hash_{B_j}\right)}{hashmax} \leq \alpha, \tag{2}$$

for some $k_{min} \leq k \leq H$ so that the mining node has mined the block B_k.

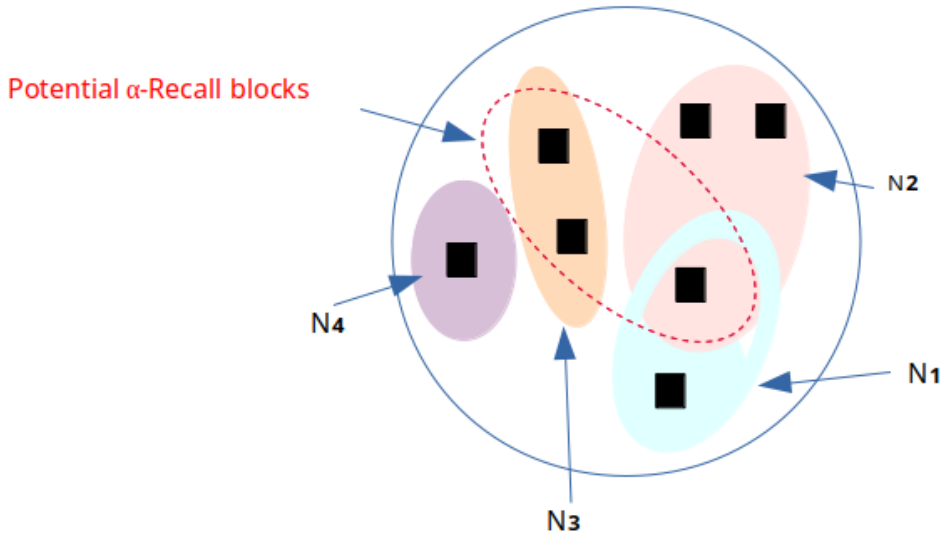

Figure 3: A potential recall blocks.

Figure 3 illustrates the selection of blocks that could be recall blocks. The red circle indicates the blocks that match condition (1), that are the same for all the nodes. Then each node has a specific set of potential recall blocks that they can use from condition (2), represented in the figure by the colored spots. For example, in this figure, the node N_4 has no potential recall block and needs to mine without it before the next block. Node N_1 and N_2 can both use only one block, and it is the same. Finally, node N_3 has two potential recall blocks that happen to not be potential recall blocks for the other nodes represented.

In the rest of the paper we will denote $N := H - k_{min} + 1$ the number of blocks with height higher than k_{min}. This is the number of block which mining gives the possibility to its successful miner to use recall blocks and get higher mining rewards through a higher probability of mining.

7 Protocol Equilibrium

7.1 Block replication

The following Proposition indicates how many, under the protocol, blocks will be replicated, and how many blocks a particular node will have to store.

Proposition 7.1. We assume that all miners has a fraction of the hash rate $\beta \ll \frac{1}{\sqrt{\alpha N}}$. Then we have:

- A block will be replicated $\approx \alpha N$ times among storage nodes.
- A node will store $\approx n\alpha H$ blocks for the protocol, where n is the number of blocks with height higher than k_{min} that the node has mined.

As a preparation to prove Proposition 7.1, we will prove the following Lemma:

Lemma 7.2. We assume that $\beta \ll \frac{1}{\sqrt{\alpha N}}$. Then most miners do not have a block to store from the equation (2) for two different blocks B_k that they have mined.

Proof. If we denote n the number of blocks of height $\geq k_{min}$ mined by the node, the probability that all these n blocks bring different history blocks to store is

$$\prod_{k=0}^{n-1} \frac{H - k\alpha H}{H}$$

as for each new of the n blocks, there remains $H - k\alpha H$ blocks that have not been selected before among the H that will be chosen at random by the hash of (2). Then we have this probability approximately equal to

$$\begin{aligned}
\prod_{k=0}^{n-1} \frac{H - k\alpha H}{H} &= \prod_{k=0}^{n-1} (1 - k\alpha) \\
&= \exp\left(\sum_{k=0}^{n-1} \ln(1 - k\alpha) \right) \\
&\approx \exp\left(-\sum_{k=0}^{n-1} k\alpha \right) \\
&= \exp\left(-\frac{(n-1)n}{2} \alpha \right).
\end{aligned}$$

We can approximate in order of magnitude that $n \approx \beta N$ by looking at the probability of mining each block. Then the assumption that $\beta \ll \frac{1}{\sqrt{\alpha N}}$ makes this probability close to 1, which proves that most miners do not have a block to store from the equation (2) for two different blocks B_k that they have mined. The result is proved. □

Proof of Proposition 7.1 We start with the second point: as each of the n mined blocks from the miner allows to use αH blocks as recall blocks by (2) by the fact that each of the H block has a probability α to verify this equation together with the law of large numbers, we

have that the second point of the Proposition is proved as all these potential recall blocks are different by Lemma 7.3.

The first point stems from the fact that for a given block B_j, each of the N blocks with height $\geq k_{min}$ have a probability α to verify (2), and therefore they are potential recall blocks for αN blocks B_k approximately by the law of large numbers, and these B_k are all from different nodes from Lemma 7.2. □

7.2 Average reward of a miner

Similar to Arweave, the reward given to the miner incentives him to store as many blocks as possible among the blocks he is meant to store, as he may then benefit of a better mining rate. We quantify this reward in this section.

To understand the reward of a miner, we first need to give an intermediary result on the probability distribution of the number of valid potential recall block that a miner has.

Lemma 7.3. Let $n \geq 0$ be the number of blocks that the node has validated. Then the probability for the node to have $k \geq 0$ valid recall blocks is given by

$$\frac{(\lambda n)^k}{k!} e^{-\lambda n}. \tag{3}$$

Proof. At each iteration of the block validation, there are approximately $\lambda(\alpha H)^{-1} H = \lambda \alpha^{-1}$ potential recall blocks that satisfy condition (1). Independently, there will be approximately αH, blocks satisfying the condition (2). As this holds for all the n blocks that the miner has validated in the past, we have by Proposition 7.1 that $\alpha n H$ blocks are satisfying the condition (2) for one of the blocks mined by the miner. We can get the probability to have $k \geq 0$ blocks in the intersection thanks to the binomial law probability:

$$\binom{\lambda \alpha^{-1}}{k} (n\alpha)^k (1 - n\alpha)^{\lambda \alpha^{-1} - k}.$$

Now using the fact that $k \ll \lambda \alpha^{-1}$, we have

$$\binom{\lambda \alpha^{-1}}{k} \approx \frac{(\lambda \alpha^{-1})^k}{k!}.$$

Using that $n\alpha \ll 1$, we have that

$$\begin{aligned} (1 - n\alpha)^{\lambda \alpha^{-1} - k} &= e^{(\lambda \alpha^{-1} - k) \ln(1 - n\alpha)} \\ &\approx e^{-n(\lambda \alpha^{-1} - k)\alpha} \\ &\approx e^{-n\lambda}. \end{aligned}$$

Therefore we have finally

$$\binom{\lambda \alpha^{-1}}{k} (n\alpha)^k (1 - n\alpha)^{\lambda \alpha^{-1} - k} \approx \frac{(\lambda n)^k}{k!} e^{-\lambda n}.$$

The result is proved. □

Theorem 1. If a miner mines under the sharding protocol of Section 6, then its average reward is given by:

$$rwd_{tot}\frac{hashrate_{miner}}{hashrate_{total}}\frac{1}{2}\left(1+\frac{1-e^{-n\lambda}}{\sum_{k=0}^{\infty}(1-e^{-k\lambda})\nu_k}\right), \tag{4}$$

Where rwd_{tot} is the total reward distributed to miners, $hashrate_{miner}$ and $hashrate_{total}$ are the hash rate of the miner and the total hash rate, n is the number of blocks that the miner has already mined, and for all $k \geq 0$, ν_k is the fraction of the hash rate delivered by the miners that have mined k blocks.

Proof.

We use Lemma 7.3 to get that at each new block mining step, if a node has validated n blocks in the past, the probability he has to have no recall blocks is given by $e^{-n\lambda}$. Let β_1 be the mining rate with the hash of the mined block being lower than Θ_0, and let β_2 be the mining rate with the hash of the mined block being higher than Θ_0 (but lower than Θ_{recall}). Let ν_k be the fraction of the computing power of miners that have already mined k blocks. The total win rate of hashing is given by

$$\begin{aligned}&\beta_1\sum_{k=0}^{\infty}\nu_k+\beta_2\sum_{k=0}^{\infty}\nu_k(1-e^{-k\lambda})\\&=\beta_1+\beta_2\sum_{k=0}^{\infty}\nu_k(1-e^{-k\lambda}),\end{aligned}$$

where the second term is obtained as the bonus that miners have if they can have a recall block. From the definition of the protocol, the difficulties are adjusted so that these two win rates are made equal. Hence, we get that $\beta_2 = \frac{\beta_1}{\sum_{k=0}^{\infty}(1-e^{-k\lambda})\nu_k}$. We finally get the average expectation of reward obtained by a miner that is fully storing its attributed recall blocks if he has already mined n blocks by re-injecting this ratio. We then get the result we wanted to prove:

$$rwd_{tot}\frac{hashrate_{miner}}{hashrate_{total}}\frac{1}{2}\left(1+\frac{1-e^{-n\lambda}}{\sum_{k=0}^{\infty}(1-e^{-k\lambda})\nu_k}\right).$$

□

7.3 Optimal behaviour of miners

7.3.1 Storage of their allocated blocks

We see from the reward equation (4) that the miners are rewarded to store the $\approx n\alpha H$ blocks satisfying (2), where we denote by $n \geq 0$ the number of blocks with indices higher than k_{min} validated by the miner. This reward is strictly higher than the reward obtained with a block without recall block, therefore if the miner wants to maximize its reward, he will chose to store its attributed blocks, also knowing the the number of blocks to store should not be too important by design.

7.3.2 Sharing of all the stored blocks

The storage nodes have an incentive to share their blocks, as similar to Arweave, the file sharing system uses the AIIA meta-game, and well behaving nodes receive the new blocks quicker than less well-ranked nodes.

7.3.3 Downloading of all the blocks allocated for storage

It is better for a miner to download all the blocks it is responsible for. Indeed, as $\approx \frac{\lambda}{\alpha H} H = \alpha^{-1}$ potential recall blocks are selected for each new blocks by (1), the miner needs to store as much of these blocks to maximise its likelihood to obtain the reward enhanced by the addition of a recall block.

We could even quantify this level of incentive.

Theorem 2. If a miner has mined $n \geq 1$ blocks with a height higher than k_{min}, then the reward loss of not storing one block from the miner's allocation is given by

$$rwd_{tot} \frac{hashrate_{miner}}{hashrate_{total}} \frac{\lambda}{2\alpha H} \frac{e^{-n\lambda}}{\sum_{k=0}^{\infty}(1-e^{-k\lambda})\nu_k}. \tag{5}$$

Proof. The result is obtained by differentiating n by $-\frac{1}{\alpha H}$ in (4). □

We study in Subsection **??** the best parameter λ to choose.

7.3.4 Downloading in priority rare blocks

As the blocks that satisfy (1) and therefore are potential recall blocks are the same set for every miner, if one of them is rarely spread (because it has been less selected at random, or it has been chosen by nodes that became inactive), there will be less competition for the mining when they are selected, and these rare blocks will be downloaded in priority.

Notice that we could object that this incentive is relatively weak. Notice that we could tweak the protocol to solve this problem by adding a 10% probability to use the Arweave mining protocol, providing this way an incentive to focus on the rare blocks.

7.3.5 Staying an active miner

Recall that we denote N the number of blocks that allows the use of a recall block for mining. As the privilege of being an storage node gives a very large reward increase, and the privilege is short-term as we take $N \ll H$, the nodes that mine blocks tend to stay active.

7.4 Block storage handling by miners

We have proved that the optimal behaviour of miners is to store all the blocks satisfying (2). Now we explain how to do it in practice. For all k and j, the equation $\frac{Hash_2\left(hash_{B_k}, hash_{B_j}\right)}{hashmax} \leq \alpha$ will never become true after having been false, as α decreases over time. Therefore the algorithm that the miner can apply to know which block to store is the following:

- When the miner mines succesfully a new block B_k, he computes $Hash_2\left(hash_{B_k}, hash_{B_j}\right)$ for all $j \geq 1$.
- the miner stores these values for all j so that the equation (2) is satisfied. He can then download the associated block.
- At each new block validation and update of α, the miner goes through all the values of $Hash_2\left(hash_{B_k}, hash_{B_j}\right)$ that he has stored, and delete all these that become higher than the deacreasing α. He may then delete all the associated blocks that he no longer needs to store to secure the history of the blockchain.

This approach is efficient since each block mined only requires H calculations. Notice that including $hash_{B_k}$ in the equation makes the result unpredictable to avoid the miner from "selecting" the moments when to use their computing power. α varies with the height H of the blockchain, as the number of blocks grows, α approaches 0, and old blocks are erased from memory, but they are stored by new successful miners, as showed by Theorem 3.

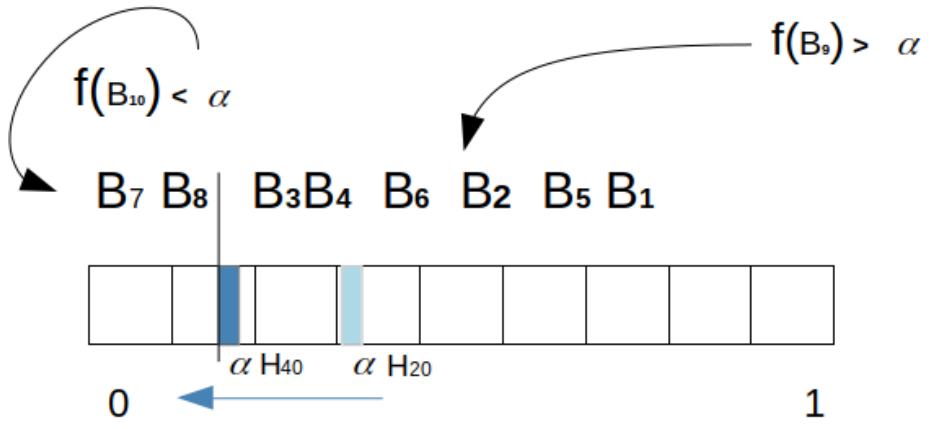

Figure 4: The chain in a node that stores block according to 6.2

Figure 4 shows an example of the inclusion of a block in the storage of a given node that has mined the block B_2. Let B be a block, we denote $f(B) := \frac{Hash_2\left(hash_{B_2}, hash_B\right)}{hashmax}$. Recall that by (2), the node is rewarded for using B_j as a recall block if $f(B_j) \leq \alpha$. The line in the figure represents α, as we can see, α converges to zero as the height H of the blockchain goes to infinity. The node stores the two blocks B_7 and B_8 (blocks to the left of the line), on the other side, blocks B_6, B_2, B_5, and B_1 are not stored by the node (blocks to the right of the line). According to 6.2, the result of this equation for the new block B_9 entering the blockchain is greater than α, so the node does not store it, and for B_{10} the result is less than α, so the node stores it. As α moves to zero while H goes to infinity, we see an evolution in the blocks stored by the node. For example, blocks B_3 and B_4 will provide recall block rewards to the node at $H = 20$, but it will no more be the case when $H = 40$. On the long run, all these blocks will

stop providing recall blocks rewards, but they will be replaced by new blocks B having a small $f(B)$.

Even if a node removes older blocks, all network nodes keep the block hashes.

7.5 Guarantee that no block will be lost

If α decreases too fast, there is a risk that not enough miners are rewarded to store a given block, and an attacker could try to lose the data, resulting in a permanent data loss. Therefore, we provide here a condition on the size of α that guarantees that no block will be lost.

Theorem 3. Let $\gamma > 0$ be the fraction of computing power of misbehaving nodes. Then if for some $\delta > 0$ we have

$$\alpha \geq \frac{2+\delta}{1-\gamma}\frac{\ln(H)}{N}, \tag{6}$$

then with a probability 1, all the history of blocks will stay available for H large enough.

Proof. We denote γN the set of blocks with height higher than k_{min} mined by malicious nodes. Recall that we denote by N the total number of blocks with height higher than k_{min}. Let b be a block, we compute the probability that no mis-behaving node stores b. Recall that a node having mined a block from $N \setminus \gamma N$ stores b with probability α for this node by Proposition 7.1, therefore the probability that none stored b is given by:

$$\begin{aligned}
\mathbb{P}\left(b \notin N \setminus \gamma N\right) &= (1-\alpha)^{(1-\gamma)N} \\
&= e^{\ln(1-\alpha)(1-\gamma)N} \\
&\approx e^{-(1-\gamma)\alpha N}
\end{aligned}$$

Then as the probability of multiple possible events is lower than the sum of their probabilities, we have

$$\begin{aligned}
\mathbb{P}\left(\exists b \notin N \setminus \gamma N\right) &\leq H e^{-(1-\gamma)\alpha N} &(7)\\
&\leq e^{-(1-\gamma)\alpha N + \ln(H)} &(8)
\end{aligned}$$

Then injecting (6) in (7), we have

$$\mathbb{P}\left(\exists b \notin N \setminus \gamma N\right) \leq H^{-(1+\delta)}$$

which is the term of a converging series, therefore by the Borel-Cantelli theorem, the event the all blocks are stored by at least one well-behaving node will be always true for H large enough almost surely. □

8 Choice of parameters for the protocol

Recall that we denote H the height of the blockchain, i.e. the total number of blocks, and we denote by N the number of blocks with height higher than k_{min} allowing their miner to get storage reward. Notice that $N = H - k_{min} + 1$.

8.1 Make sure that no block of the history of the blockchain is lost

In practice, Theorem 3 ensures that the storage of the Sparsechain protocol will be safe against 75% of misbehaving nodes if we set

$$\alpha := \frac{10\ln(H)}{N}. \tag{9}$$

8.2 Make sure storage nodes are renewed enough while still dividing well the memory

In order to satisfy the incentive from subsection 7.3.5, we need to have $N \ll H$. As this can stay very marginal, we can pick

$$N := \frac{H}{\ln(\ln(H))}. \tag{10}$$

Then, by (9), we have

$$\alpha := \frac{10\ln(H)\ln(\ln(H))}{H}. \tag{11}$$

8.3 Trade-off with memory and transactions per seconds

By Proposition 7.1, a node stores $\alpha N = 10\ln(H)$ blocks. From proportional scaling of Bitcoin figures, a year sees the validation of 50000 blocs, and we can have 6000 transactions per second for each Gb in a block. With these numbers, we obtain the following equation:

$$\begin{aligned} \#\text{tx/s} &= \frac{600}{\ln(H)\ln(\ln(H))}\#\text{Gb/node} \\ &\overset{1 \text{ year}}{=} 25\#\text{Gb/node} \end{aligned}$$

Then for example, if we want to reach 5000 transaction per seconds, each storage node needs to store 200Gb per year, which seems ok for retail nodes. Then each block has a size $\frac{200\text{Gb}}{\alpha H} =$ 800Mb.

8.4 Reward per stored block

We have from Theorem 2 that the reward for storing the last block is

$$rwd_{tot}\frac{hashrate_{miner}}{hashrate_{total}}\frac{\lambda}{2\alpha H}\frac{e^{-n\lambda}}{\sum_{k=0}^{\infty}(1-e^{-k\lambda})\nu_k}. \tag{12}$$

In order of magnitude we can do estimations, using the approximation:

$$\beta := \frac{hashrate_{miner}}{hashrate_{total}} \approx \frac{n}{N}, \tag{13}$$

as this is an approximation of the fraction of blocks that the miner would have successfully mined. Notice that here we neglect on purpose the impact of the compounding rewards due to additional storage rewards.

Now if we incorporate (11), (10) and (13) in (4), we get that the reward of the miner is approximately:

$$rwd_{tot}\frac{\beta\lambda N}{20H\ln(H)}\frac{e^{-\beta\lambda N}}{\sum_{k=0}^{\infty}(1-e^{-k\lambda})\nu_k}.$$

We have that $\sum_{k=0}^{\infty}(1-e^{-k\lambda})\nu_k \leq 1$ an d in a mature market it converges to 1, as all important miners will have mined enough to have $1-e^{-k\lambda} \approx 1$. Also, in the worst case, a very strong miner should have $\beta := 1\%$ of the hashing power. To reward enough the miners to incentive them to store we need to have:

$$rwd_{tot}\frac{\beta\lambda N}{20H\ln(H)}e^{-\beta\lambda N} \geq cost_{block}. \tag{14}$$

Notice that the cost to store $1Gb$ during one year is $\approx \$0.06$. From subsection 8.3, we have that a good block size is 800Mb. As the exponential would become overwhelming and prevent any profitability of storing the blocks if $\beta\lambda N \gg 1$, we will do our computation with the assumption that $\beta\lambda N \approx 10$, as we are mostly interested in an order of magnitude. We get that we need to have for one year long (assuming annual mining reward of $\$100m$):

$$\begin{aligned}\lambda &\leq \frac{1}{\beta N}\ln\left(\frac{rwd_{tot}}{2H\ln(H)cost_{block}}\right)\\ &= \frac{1}{\beta N}\ln\left(\frac{100000000}{2\cdot 50000\ln(50000)0.06\cdot 0.8}\right)\\ &\approx \frac{7}{\beta N}.\end{aligned}$$

Then a good option would be taking

$$\lambda := \frac{500}{N}, \tag{15}$$

as the cost of storage would diminish faster than the raise of $H\ln(H)$.

9 Conclusion

SparseChain is a new cryptocurrency design based on Proof_of_work and Proof of Random Access that allows storage scalability without compromising security and guarantees availability. SparseChain achieves its goals using three techniques: (1) Sharding storage, which helps SparseChain securely distribute the storage over nodes and helps nodes with a small capacity to participate in the network and maintain the blockchain; (2) Incentivizes storage, as miners need access to random blocks from the chain,Äôs history to quickly mine new blocks and receive mining rewards; and (3) Valid block rules, SparseChain is made so that the number of potential recall blocks for a miner is proportional to its computing power. The protocol achieves scalability in a decentralized way, availability of data, and security guarantees.

References

[1] Eli Ben-Sasson, Alessandro Chiesa, Christina Garman, Matthew Green, Ian Miers, Eran Tromer, and Madars Virza. Zerocash: Decentralized anonymous payments from bitcoin. *2014 IEEE Symposium on Security and Privacy*, pages 459–474, 2014.

[2] Joseph Bonneau, Izaak Meckler, Vanishree Rao, Evan, and Shapiro. Mina : Decentralized cryptocurrency at scale. 2021.

[3] Vitalik Buterin. A next generation smart contract & decentralized application platform. 2015.

[4] Beth Cohen. Incentives build robustness in bit-torrent. 2003.

[5] Christopher Hannon and Dong Jin. Bitcoin payment-channels for resource limited iot devices. *Proceedings of the International Conference on Omni-Layer Intelligent Systems*, 2019.

[6] Eleftherios Kokoris-Kogias, Philipp Jovanovic, Nicolas Gailly, Ismail Khoffi, Linus Gasser, and Bryan Ford. Enhancing bitcoin security and performance with strong consistency via collective signing. *ArXiv*, abs/1602.06997, 2016.

[7] Eleftherios Kokoris-Kogias, Philipp Jovanovic, Linus Gasser, Nicolas Gailly, Ewa Syta, and Bryan Ford. Omniledger: A secure, scale-out, decentralized ledger via sharding. *2018 IEEE Symposium on Security and Privacy (SP)*, pages 583–598, 2018.

[8] Loi Luu, Viswesh Narayanan, Chaodong Zheng, Kunal Baweja, Seth Gilbert, and P. Saxena. A secure sharding protocol for open blockchains. *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, 2016.

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

[10] Michael Piatek, Tomas Isdal, Thomas E. Anderson, Arvind Krishnamurthy, and Arun Venkataramani. Do incentives build robustness in bit torrent. 2007.

[11] Sam A. Williams, Viktor Diordiiev, and Lev Berman. Arweave: A protocol for economically sustainable information permanence. 2019.

[12] Sam A. Williams and Will Jones. Archain: An open, irrevocable, unforgeable and uncensorable archive for the internet. 2017.

[13] Daniel Davis Wood. Ethereum: A secure decentralised generalised transaction ledger. 2014.

[14] A. T. Yakovenko. Solana : A new architecture for a high performance blockchain v 0 . 8. 2018.

[15] Mahdi Zamani, Mahnush Movahedi, and Mariana Raykova. Rapidchain: A fast blockchain protocol via full sharding. *IACR Cryptol. ePrint Arch.*, 2018:460, 2018.