SparseChain: Sparse Distributed Blockchain Storage Protocol

Kahina Khacef

SparseChain: Sparse Distributed Blockchain Storage Protocol

Kahina Khacef
Sorbonne Université, LIP6
Paris, France
kahina.khacef@lip6.fr

***Abstract*—Storage in the blockchain is crucial because full nodes maintain vast amounts of information to decentralize the validation system. The validation system consists of approving new transactions that basically follow two steps: verifying the chain's state and recording the transactions into a block. Verification is processed in many different ways, according to the data model used by the consensus. The most known data models used are UTXO and balance models. UTXO model stores individual transaction receipts in chronological order to check the balance for accounting, the set of all transaction outputs is the global state. The balance model incorporates accounts and keeps track of all balances as a global state; each transaction has a public nonce attached that is incremented by one with each outgoing transaction. Their transactions will only continue to grow in size over time. Therefore, scalability remains a key challenge in the blockchain when dealing with large amounts of data.**

Our research focuses on designing a scalable permissionless blockchain in Proof_of_Work systems. We present SparseChain, a storage sharding protocol that allows scalable storage and provides permanent availability of data by incentive. The protocol decreases block replication while maintaining security. Peers can validate transactions even if their local chain is a shard and store only a part of the whole blockchain, making the validation process suitable for lightweight peers. The peers are incentivized to keep only a suitable amount of data and are rewarded for storing historical data. We present the protocol, and then we analyze its performance compared to the traditional blockchain.

***Index Terms*—Permissionless Blockchain, Proof of Work, Decentralization, Security, Availability, Proof of Random Access.**

I. Introduction

The blockchain is a series of blocks of data created by peers and validated in a distributed way without depending on central entities, allowing parties that do not trust each other to exchange funds and agree on a common view of their states. The blockchain system is built by respecting decided rules that make the consensus. Bitcoin [1] is the first digital currency implementation that introduces a blockchain system. This proposition is attractive since the technology complies with cryptographic algorithms (Elliptic Curve [23]) and hashing functions to protect the integrity and immutability of data stored on the distributed ledger using the Peer_to_Peer (P2P) network to interconnect nodes. Nodes are divided into light nodes, full nodes, and miners. The light node keeps only a copy of the block header. The full node validates transactions and blocks, and relays them to all miner nodes. Although technically not required, a miner is a full node (i.e., has full knowledge of the blockchain) that creates new blocks and is different from nodes that only validate data. All miner nodes rely on the full nodes to create blocks.

Bitcoin maintains a continuously-growing history of organized information to establish consensus and provide security. Full nodes replicate all previous transactions to avoid double-spending attacks, which are utilized as proof of correct status to keep the system running, i.e., 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.

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 [11], [13], [14], and Off_chain approaches, i.e., Lightning Network [12]. 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. 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. These techniques provide scalability but affect decentralization and result in security vulnerabilities, i.e., the Off_chain scalability solutions can introduce security threats to the blockchain because the data is stored not directly on the blockchain but through third-party protocols.

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.
- 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 II presents an overview of SparseChain, Section III discusses the related work, Section IV describe the storage sharding protocol which enhances the scalability of the blockchain, section V discusses the protocol economics, and section VI discusses the parameters for the protocol.

II. Overview

Previously, we proposed a sharding approach that reduces block replication in the chain [21]. The idea is that the process acts as in the Bitcoin protocol, but transactions are not recorded in full replication. Only the blocks header are stored by nodes to achieve consensus. Given a network with C nodes, the block replication is αC, with $(0 < \alpha < 1)$. Once a block is verified and confirmed by the network (i.e., a transaction is considered a success after six block confirmations), its replication decreases. We proposed an approach for a finite number of nodes C, and we underlined the major problem of data availability. Malicious behavior of miners can cause deleting data from nodes. This paper proposes a distributed protocol that scales the blockchain by decreasing the replication of blocks and ensuring permanent availability. We introduce an incentive mechanism to incentivize nodes to store a suitable amount of data and keep the historical data. We prove that every block will not be lost in time. Like the way Arweave [4] incentivizes peers to store data permanently, see II-C3, SparseChain rewards miners for two things were mining and storing data. Miners are incentivized to maintain data and can mine with a historical block or without.

A. Motivation

In traditional blokchain, each transaction is replicated redundantly across multiple nodes. 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. 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 be shared on a subset of the nodes, and nodes store only a small amount of data. More precisely, the block in the blockchain is constituted on the *block header* and the *body*. The block header contains the consensus data (Merkle root, previous block hash, nonce, and other metadata), and the body includes transactions. SparseChain deletes the body from blocks to decrease replication and keep each block header to achieve consensus. In addition, the protocol allows for an increase in 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.

B. Objectives

The SparseChain protocol achieves the following main goals:

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 protocol lowers the cost of holding the blockchain; in fact, the block size can be increased to allow more transactions without overburdening the network.

C. Background

1) Blockchain infrastructure: As adopted in most public cryptocurrencies, the blockchain infrastructure is based on distributed ledger (DLT). Blockchain requires consensus to ensure node replication. Each node (or peer) in the network follows the same set of rules, including transaction processing, block creation. Nodes communicate by sending messages through Gossip protocol and form a P2P network topology on top of the internet. Transactions are entered into the blockchain in chronological order and stored as a series of blocks scattered over multiple nodes. Each node replicates and saves an identical copy of the ledger on its computer, which it updates itself according to the protocol's rules. Every block encloses a set of transactions that should be valid and clear of double spending, and the full nodes store the chain since the genesis block. The blockchain is distributed, tamper resistant and does not rely on a centralized authority.

2) 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 on 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. Once verified, the transaction is added to a valid transaction block by miners competing and executing a consensus to extend the longest chain with blocks they generate.

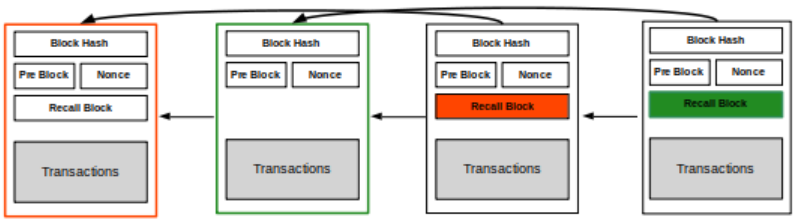

Fig. 1. An overview of the block data structure illustrates the link to both the previous block and recall block.

Traditional blockchain has robust incentive mechanisms to encourage nodes to maintain the network running economically. For example, 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' [6]. 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* [25] where all nodes are incentivized to share resources (data) freely at the network's maximum capacity and perform prosocial behaviors.

3) Arweave: Arweave set out to solve the problem of long-term data storage on the internet. It used blockchain technology to store data and implemented an incentive system for peers to store data permanently. Arweave constructed a blockweave instead of a blockchain. The blockweave is held together by many links inside the whole data storage. Therefore, the only way to add new data to the blockweave is for a peer to recall a randomly selected block already on the blockweave. Only the peers that can recall the previous random block are allowed to store new data [5]. 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. This improvement in data storage and economic rewards for miners who keep the data create conditions for data to be stored for long periods. In addition, 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 blocks from the genesis block. 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 incentivized through the Adaptive Interacting Incentive Agent (AIIA) meta-game [4], 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.

D. Architecture of SparseChain

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 V.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.

SparseChain rewards the storage of the history of the blockchain through the consensus algorithm. 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) as for Arweave. Therefore, there cannot be pure storage nodes that get rewarded for this; the miners need to be in charge of the storage by the protocol.

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. Unlike 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. 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 II-C3). 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 IV to make the scalability possible through sparsity.

III. Related Work

Sharding blockchain is the closest work to ours. Elastico is the first public sharding blockchain that tolerates byzantine adversaries [14]. 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 [24]) for intra-committee consensus. Omniledger, 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 [11]. The intra-shard consensus protocol uses a variation of ByzCoin [15]. 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. In sharding-based systems, database performance scales linearly with the number of nodes, necessitating the creation of complicated protocols to enable shard connectivity.

Mbinkeu et al. [16] looked into the memory management and access time of the Bitcoin protocol, which uses SQLite databases. A memory optimization approach based on a redundancy system is developed by Guo et al. [20] to reduce the storage capacity of each node. It suggested a redundancy-based optimization strategy that significantly reduces the storage capacity of blockchain system nodes and creates a fault-tolerant mechanism. Wang et al. [17] looked at distributing data across a blockchain network. This work proposed a balanced user input solution for search time and space occupation.Gennaro et al. [18] developed a centralized threshold signature mechanism for more efficient Bitcoin systems in light of the challenges with Bitcoin key management. To increase the performance and scalability of data [39] sharing, El-Hindi et al. [19] added a database layer to the blockchain system.

Mina [7] 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 [9]. 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. However, 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.

IV. Storage Sharding Protocol

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

A. System Model

There are two primary ways to represent states in blockchains. UTXO model is used in Bitcoin, in which the state is comprised of the unspent transaction outputs, and the Account model is used in Ethereum, which includes accounts and their balances. For the rest of this paper, we assume a Proof_of_Work chains. The exact state model, UTXO, Account or something else are irrelevant for our purposes. The blockchain is distributed over a higher number of nodes. N is the number of blocks in the blockchain.

B. Block structure

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 in case the miner processes a new block with a recall block (i.e., the entire contents of the recall block hashed with the previous block hash and current transactions), the block is linked to the previously added block and the recall block in the chain (see Fig.1). A miner can mine without a recall block; see IV-C. 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.

C. Block validity rules

Every block must follow two rules to be considered valid:

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 shown in Fig.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.

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 ensures security while reducing block replication.

In order to make sure that storage stays rewarded enough, Θ_{recall} is adjusted to have 50% of validating hashes $\leq \Theta_0$. It 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.

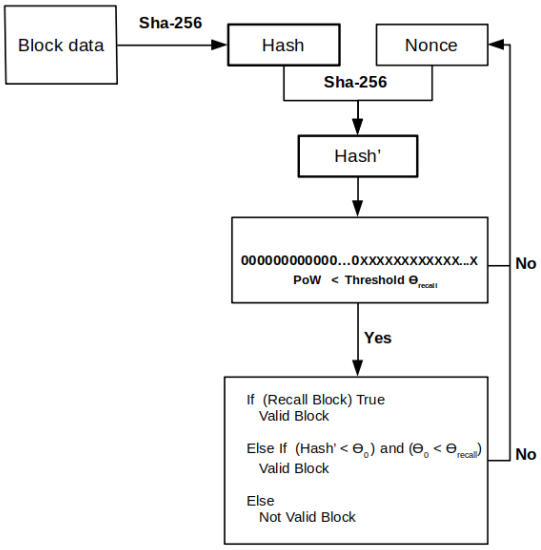

Fig. 2. An illustration of mining block in SparseChain

2) Recall block validity: The goal of SparseChain is to give an incentive to miners to store the history of the blockchain same as 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 IV.1.

Another difference is that each miner gets a set of valid recall blocks to store. Unlike Arweave, huge miners are not twice advantaged because they have a large computation and 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 IV.1.

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 VI.

Definition IV.1. 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.

Fig.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 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 use only one block, which is the same. Finally,

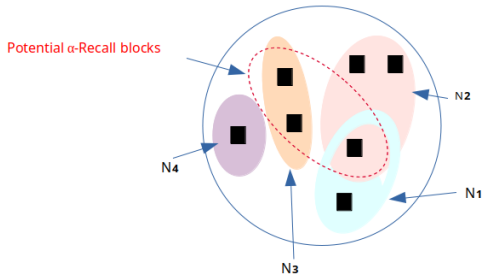

Fig. 3. A potential recall blocks.

node N_3 has two potential recall blocks that are not 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 a height higher than k_{min}. This is the number of blocks which mining gives the possibility to its successful miner to use recall blocks and get higher mining rewards through a higher probability of mining.

V. Protocol Equilibrium

A. 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 V.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.

B. Average reward of a miner

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

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

Lemma V.2. Let $n \geq 0$ be the number of blocks the node has validated. Then the probability for the node to have $k \geq 0$ valid recall blocks is given by

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

Now we quantify the reward obtain by each miner.

Theorem 1. If a miner mines under the sharding protocol of Section IV, 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. For all $k \geq 0$, ν_k is the fraction of the hash rate delivered by the miners that have mined k blocks.

C. Optimal behaviour of miners

1) Storage of their allocated blocks: We see from the reward equation (4) that the miners are rewarded for storing 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 a recall block. Therefore if the miner wants to maximize its reward, he will choose to store its attributed blocks. Also, knowing the number of blocks to store should not be too important by design.

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.

3) Downloading of all the blocks allocated for storage: A miner should 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 block by (1), the miner needs to store as many of these blocks to maximize its likelihood of obtaining 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

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

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

We study in Subsection VI-D the best parameter λ to choose.

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 an incentive to focus on the rare blocks.

5) Staying an active miner: Recall that we denote N as the number of blocks that allows the use of a recall block for mining. As the privilege of being a 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.

D. Block storage handling by miners

We have proved that the optimal behavior 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 deletes all these that become higher than the decreasing α. He may then delete all the associated blocks 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, preventing 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 new successful miners store them, as shown by Theorem 3.

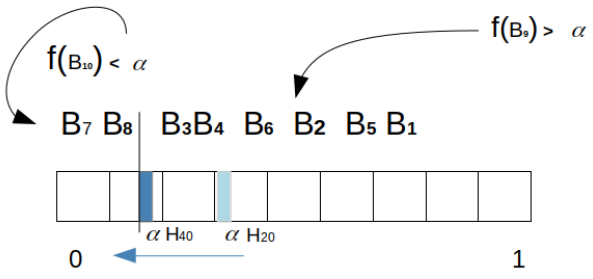

Fig. 4. The chain in a node that stores block according to IV.1

Fig.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 IV.1, 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 $H = 40$. In 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.

E. Guarantee that no block will be lost

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

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

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

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

VI. 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 a height higher than k_{min} allowing their miner to get storage reward. Notice that $N = H - k_{min} + 1$.

A. 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

$$\alpha := \frac{10\ln(H)}{N}. \tag{7}$$

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

To satisfy the incentive from subsection V-C5, we need to have $N \ll H$. As this can stay very marginal, we can pick

$$N := \frac{H}{\ln(\ln(H))}. \tag{8}$$

Then, by (7), we have

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

C. Trade-off with memory and transactions per seconds

By Proposition V.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:

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

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

D. Reward per stored block

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

$$rwd_{tot}\frac{hashrate_{miner}}{hashrate_{total}}\frac{\lambda}{2\alpha H}\frac{e^{-n\lambda}}{\sum_{k=0}^{\infty}(1-e^{-k\lambda})\nu_k}. \tag{10}$$

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

$$\beta := \frac{hashrate_{miner}}{hashrate_{total}} \approx \frac{n}{N}, \tag{11}$$

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 (9), (8) and (11) 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 $\sum_{k=0}^{\infty}(1-e^{-k\lambda})\nu_k \leq 1$ and 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 powerful miner should have $\beta := 1\%$ of the hashing power. To reward enough the miners to incentive them to store, we need to have:

$$rwd_{tot}\frac{\beta\lambda N}{20H\ln(H)}e^{-\beta\lambda N} \geq cost_{block}. \tag{12}$$

Notice that the cost to store $1Gb$ during one year is $\approx \$0.06$. From Subsection VI-C, 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 the order of magnitude. We get that we need to have it for one year long (assuming an annual mining reward of $\$100m$):

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

Then a good option would be taking

$$\lambda := \frac{500}{N}, \tag{13}$$

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

VII. Scale comparison

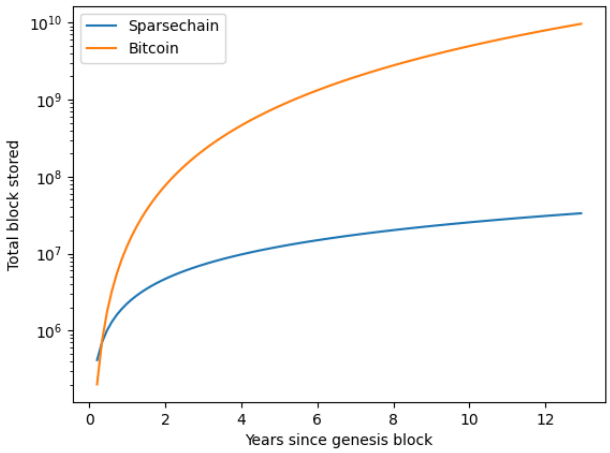

Fig. 5. Number of blocks stored after 13 years of blockchain for SparseChain and Bitcoin

Figure 5 shows the evolution of the total number of blocks stored for the Bitcoin blockchain and SparseChain. After 13 years, we see that SparseChain spares a factor of 1000 while guaranteeing the network's security.

We model the number of nodes of Bitcoin by $n_{nodes} = 15000(t_{years}/13)^{1.5}$, then its total number of blocks stored is given by Hn_{nodes}.

When it comes to SparseChain, supposing that no node has an excessive computation power, the total number of blocks stored is given by $N\alpha H$ as αH blocks are stored for each block B_k with $k \geq k_{min}$, and these blocks are all different with high probability if the node does not have excessive computing power. From the formulas given by Section VI, we get that the total number of blocks stored is given by $H\frac{10\ln(H)}{\ln(\ln(H))}$.

Finally, for the evolution of the number of blocks with respect to time for both Bitcoin and SparseChain, we take $H = 50000 \cdot t_{years}$.

VIII. 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.

References

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

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

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

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

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

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

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

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

[9] 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.

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

[11] 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.

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

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

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

[15] 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.

[16] Rodrigue Carlos Nana Mbinkeu and Bernabé Batchakui. Reducing disk storage with sqlite into bitcoin architecture. Int. J. Recent Contributions Eng. Sci. IT, 3:10–14, 2015

[17] Peng Zhao, Hongbing Cheng, Yicheng Fang, and Xiaoqing Wang. A secure storage strategy for blockchain based on mcmc algorithm. IEEE Access, 8:160815–160824, 2020.

[18] Dan Boneh, Rosario Gennaro, and Steven Goldfeder. Using level-1 homomorphic encryp- tion to improve threshold dsa signatures for bitcoin wallet security. In LATINCRYPT, 2017.

[19] Muhammad El-Hindi, Carsten Binnig, Arvind Arasu, Donald Kossmann, and Ravishankar Ramamurthy. Blockchaindb - a shared database on blockchains. Proc. VLDB Endow., 12:1597–1609, 2019.

[20] Zhaohui Guo, Zhen Gao, Haojuan Mei, Ming Zhao, and Jinsheng Yang. Design and optimization for storage mechanism of the public blockchain based on redundant residual number system. IEEE Access, 7:98546–98554, 2019.

[21] Kahina Khacef, Salima Benbernou, Mourad Ouziri, Muhammad Younas. Trade-off Between Security and Scalability in Blockchain Design: A Dynamic Sharding Approach. Conference on Deep Learning, Big Data and Blockchain, Aug 2021, Rome (virtual), Italy. pp.77-90.

[22] https://hal.archives-ouvertes.fr/hal-03676819/

[23] Neal Koblitz. Elliptic curve cryptosystems. Mathematics of Computation, 48:203–209, 1987.

[24] Miguel Castro and Barbara Liskov. Practical byzantine fault tolerance. In Proceedings of the Third Symposium on Operating Systems Design and Implementation, OSDI '99, page 173–186, USA, 1999. USENIX Association

[25] John Nash. Non-cooperative games. Annals of Mathematics, 54(2):286–295, 1951.