

Achieving Almost All Blockchain Functionalities with Polylogarithmic Storage

Parikshit Hegde¹, Robert Streit¹, Yanni Georgiades¹, Chaya Ganesh², and
Sriram Vishwanath¹

¹ The University of Texas at Austin, TX, USA {`hegde`, `rpstreit`,
`yanni.georgiades`, `sriram`}@utexas.edu

² Indian Institute of Science, KA, India
`chaya@iisc.ac.in`

Abstract. In current blockchain systems, full nodes that perform all of the available functionalities need to store the entire blockchain. In addition to the blockchain, full nodes also store a blockchain-summary, called the *state*, which is used to efficiently verify transactions. With the size of popular blockchains and their states growing rapidly, full nodes require massive storage resources in order to keep up with the scaling. This leads to a tug-of-war between scaling and decentralization since fewer entities can afford expensive resources. We present *hybrid nodes* for proof-of-work (PoW) cryptocurrencies which can validate transactions, validate blocks, validate states, mine, select the main chain, bootstrap new hybrid nodes, and verify payment proofs. With the use of a protocol called *trimming*, hybrid nodes only retain polylogarithmic number of blocks in the chain length in order to represent the proof-of-work of the blockchain. Hybrid nodes are also optimized for the storage of the state with the use of *stateless blockchain* protocols. The lowered storage requirements should enable more entities to join as hybrid nodes and improve the decentralization of the system. We define novel theoretical security models for hybrid nodes and show that they are provably secure. We also show that the storage requirement of hybrid nodes is near-optimal with respect to our security definitions.

Keywords: Blockchains, Cryptocurrency, Storage, NIPoPoW, Hybrid Nodes, Trimming

1 Introduction

Blockchains enable a group of untrusting parties to securely maintain a distributed ledger without relying on a trusted third party. Instead, the power to decide what is recorded in the blockchain is distributed amongst a set of decentralized nodes. This property is desirable for applications used by a set of mutually distrustful parties, such as a digital currency. For this reason, *cryptocurrencies* are a fundamental application of blockchains and are increasingly growing in popularity. In this paper, we focus on cryptocurrencies built on top of a proof-of-work (PoW) blockchain employing the longest chain rule. A blockchain node for a cryptocurrency typically has the following functionalities:

1. *Transaction validation*: When the node receives a new transaction, it checks if the transaction is valid with respect to the transactions already confirmed in the blockchain.
2. *Block validation*: When the node receives a new block, it verifies that the block hash is valid, all the transactions in the block are valid, and the block otherwise follows all of the conventions imposed by the protocol.
3. *State Validation*: Given a summary of currency ownership in the system, called the *state*, the node verifies that the state is consistent with the blockchain.
4. *Mining*: The node can append a block to the blockchain by verifying its contents and producing a PoW for the block. A node which does not mine blocks is assumed to have mining power 0.
5. *Chain Selection*: Given a set of conflicting chains, the node can choose the main chain which has the most PoW. Any two honest nodes which receive the same set of conflicting chains in the same order must select the same main chain.
6. *Bootstrapping new nodes to the blockchain*: A node can provide new nodes entering the system with the blockchain.
7. *Serving payment proofs*: Given a transaction tx , the node can provide a *proof* of tx 's inclusion in the blockchain.
8. *Verification of payment proofs*: Given a proof of transaction tx 's inclusion in the blockchain, the node can verify the correctness of the proof.

In order to have all of the functionalities above, a node must verify and store the entire blockchain. We will refer to such nodes as *full nodes*. Popular systems like Bitcoin and Ethereum also allow for other types of nodes with more limited functionalities [35, 29]. For instance, *pruned nodes* initially download the entire blockchain and verify it. However, they later *prune* the blockchain, meaning that they discard block data and only retain block headers for blocks older than the most recent k blocks in the blockchain. By retaining a summary of the blockchain called the *state*, they can still perform all desired functionalities except for serving proofs of payment and bootstrapping new nodes. *Lightweight nodes* only download the block-headers, and their only functionality is to verify payment proofs provided by full nodes. Importantly, full nodes are necessary to bootstrap both pruned and lightweight nodes into the blockchain.

Since full nodes need to store the entire blockchain, their resource requirements can be high. This is an entry barrier that makes fewer nodes participate, which leads to a centralization of trust. In this paper, we optimize storage requirements in order to lower this entry barrier. In deference to the storage capabilities of modern computational hardware, we divide storage into two categories. The first is *cold storage*, which is accessed infrequently and is stored on disk. This includes older blocks that are deep inside the blockchain. The second is *hot storage*, which is accessed frequently and is stored in memory. Naturally, the blockchain state used to validate blocks and transactions is kept in hot storage.

In this paper, we propose a new class called *hybrid nodes*. Hybrid nodes have all of the above functionalities except for the ability to provide payment

proofs. Importantly, hybrid nodes can bootstrap new hybrid nodes into the system, meaning they do not depend on any other type of nodes, including full nodes. Moreover, if B is the length of the blockchain, hybrid nodes only require $\text{polylog}(B)$ cold storage to represent the PoW of the chain. This is achieved by a process we call *trimming*, an extension of non-interactive-proofs-of-proof-of-work [25] (henceforth, NIPoPoW). NIPoPoW is a protocol that enables a prover (which is most often a full node) to provide payment proofs of $\text{polylog}(B)$ size rather than the traditional B size, but NIPoPoW still requires the prover to store the entire blockchain. We extend these techniques further in our trimming protocol to securely remove blocks and reduce storage.

We now comment on the *practical implications* of our proposed protocols for hybrid nodes. There are two main components of a blockchain with significant storage requirements for hybrid nodes. First is the storage required to represent the PoW of the chain, which is used by the consensus protocol. In traditional systems, since the entire chain of block-headers must be stored, the storage requirements for this component at the time of writing could be in the order of 100s of megabytes for systems such as Bitcoin and Ethereum. Our trimming protocols for hybrid nodes can decrease this requirement to the order of 100s of kilobytes. While 100MB might not seem large, if one wishes to run a number of blockchains on a single device then the storage requirement can quickly multiply into the gigabytes range if methods such as *trimming* are not employed. Moreover, since hybrid nodes only store $\text{polylog}(B)$ number of block headers, their storage requirement grows slower with time too. The second component that requires storage is the blockchain state (UTXO or account-based for cryptocurrencies). For instance, the size of Bitcoin’s UTXO set is roughly 4GB [3]. However, some novel stateless blockchain protocols reduce this storage requirement to the order of kilobytes by requiring clients to provide payment proofs [1]. In Section 7 and Appendix J, we show that hybrid nodes can employ stateless blockchain protocols, thus optimizing both their PoW and state storage.

Previous works, specifically CoinPrune and SecurePrune [28, 33], achieve the same functionalities as hybrid nodes with lower storage requirements than full nodes. They achieve this by storing a commitment to the blockchain-state in the blocks and pruning blocks that are deep in the blockchain. However, their storage requirement still scales linearly with blockchain length. Moreover, these works provide a largely qualitative analysis of their respective protocols. In contrast, we perform a rigorous security analysis and provide proofs that hybrid nodes are secure.

Concurrent with the initial submission of our work, we were made aware of an independent work that uses a modification of NIPoPoWs to obtain polylogarithmic storage [24]. Although both protocols are similarly motivated, we believe that our security definitions and the corresponding analysis are novel and crucial to this area. Of particular note, we believe that security against a trim-attack (see Section 5), is crucial for the operation of hybrid nodes. Unlike our protocol, [24] claim to not require *optimism* for *succinctness* (see further in Theorem 3

and Remark 1). However, we note that it doesn't seem economically viable for an adversary to expend resources to simply hurt the succinctness of hybrid nodes.

We defer the discussion of other related works which aim to optimize blockchain storage to Appendix A.

We now summarize our results and outline the organization of the paper. In section 2, we introduce the basic model and notation. In section 3, we summarize CoinPrune and NIPoPoW, which are building blocks for our protocol. In section 4, we explain the properties of the hybrid node's chain and describe the trimming protocol, though we defer the description of chain selection and state verification protocols to Appendix C due to space constraints. In section 5, we introduce novel security definitions for hybrid nodes, including *trim-attacked*, *congruence*, *state-attacked*, and *bootstrap-attacked*. In section 6, we show that hybrid nodes satisfy all the security properties with high probability, and we also discuss the polylogarithmic storage requirement and the lower bound on the storage requirement. For brevity, formal proofs for these results are deferred to Appendices E-H. In section 7, we illustrate that when combined with stateless blockchain protocols, hybrid nodes are optimized both in terms of cold and hot storage. And, we examine directions for future work in Section 8.

Our primary contributions are the protocols associated with the hybrid nodes and the novel security definitions and their associated theorems. These are in sections 4, 5 and 6.

2 Model and Notation

In this work, we consider a set of nodes running a PoW blockchain. We model the system using continuous time, which accurately models systems with high hash-rates such as Bitcoin and Ethereum[2, 15]. In this section, we restrict the model description to the essentials required to describe our protocol. Supplemental model details used for security analysis are deferred to Appendix B.

Several communication models are considered in the literature. The simplest is the *synchronous* model where a block broadcast by a node at a certain time is received by all other nodes immediately [17, 16]. Since time is continuous, no more than one block is mined at any given time, implying only one block could be in communication at any given time. More complicated communication models with communication delays are also considered in the literature [30, 36]. For the sake of simplicity, we consider the synchronous model in this paper and leave it to future work to transfer our results to more complicated communication models. Note that because of synchronous communication, all honest nodes have knowledge of the same set of blocks at any given time.

Basic Blockchain Notation The honest (longest-)chain at time t is represented by \mathcal{C}_t . The number of blocks in \mathcal{C}_t is called the chain-length and is denoted as B_t . When the time t is clear from context, we may drop the subscript and refer to it as just \mathcal{C} . Blocks in \mathcal{C} are indexed as an array in a similar convention to the Python programming language, meaning that $\mathcal{C}[i]$ is denoted as block i . Since it

is convenient, we refer to a block by its index i and not its contents. $\mathcal{C}[0]$ is called the *genesis block*. $\mathcal{C}[i_1 : i_2]$ represents the segment of the chain from block i_1 to block $(i_2 - 1)$. If at any time an honest node hears of another chain \mathcal{D} which is longer than \mathcal{C}_t , then it adopts \mathcal{D} as the honest chain (i.e., $\mathcal{C}_{t^+} = \mathcal{D}$, where t^+ indicates the time incrementally after t). The last common block between two chains \mathcal{C} and \mathcal{D} is called the *latest common ancestor* (LCA), and is denoted as $b = \text{LCA}(\mathcal{C}, \mathcal{D})$. Specifically, $\mathcal{C}_t[: b+1] = \mathcal{D}[: b+1]$, and $\mathcal{C}_t[b+1 :] \cap \mathcal{D}[b+1 :] = \emptyset$. When a new block \bar{b} is appended to \mathcal{C} , we denote the extended chain as $\mathcal{C}\bar{b}$.

In our model, hybrid nodes do not store the entire chain \mathcal{C}_t , but instead store a trimmed version which contains fewer blocks than \mathcal{C}_t . The trimming protocol and its associated notation is described in Section 4.

Blockchain State When a new transaction is submitted, a node must check if it is “valid” with respect to the chain \mathcal{C}_t . This could be accomplished by parsing through the complete log of transactions in \mathcal{C}_t . However, due to the rapidly increasing size of \mathcal{C}_t , it is far more efficient for a node to validate transactions against a summary of the chain called the state, and denoted as $\text{state}(\mathcal{C}_t)$. Equivalently, we may refer to the state as $\text{state}(B_t)$, where B_t is the length of the chain \mathcal{C}_t . Validating a transaction against \mathcal{C}_t is equivalent to validating it with respect to $\text{state}(\mathcal{C}_t)$, so using $\text{state}(\mathcal{C}_t)$ is preferred due to its smaller hot storage requirement. After a new block \bar{b} is added to the chain, the new state is computed as $\text{state}(\mathcal{C}_t \bar{b}) = F(\text{state}(\mathcal{C}_t), \bar{b})$, where $F()$ is a function that applies the transactions in \bar{b} to $\text{state}(\mathcal{C}_t)$. When the new block is clear from context, we denote the function simply as $F(\text{state}(\mathcal{C}_t))$, and when the function is applied on n sequential new blocks, we denote the operation as $\text{state}(\mathcal{C}_t b_1 b_2 \dots b_n) = F^n(\mathcal{C}_t)$. Two types of states are popular: 1) *UTXO-based State*: this stands for unspent transaction output, and is used by Bitcoin. The UTXO state consists of a list of unspent coins. A new transaction is valid with respect to the state, if it consumes one or more of these coins, and creates new coins whose total value is no larger than the consumed coins; 2) *Account Based State*: This is used in Ethereum. An account-based state consists of a vector of key-value mappings, with one mapping corresponding to each user. The key establishes the user’s identity, and the value establishes the balance in the user’s account. A user can issue a transaction that transfers a part of his account’s balance to another user.

Interlinks In traditional blockchains like Bitcoin, each block contains a link (using a hash) to the previous block in the chain. To enable hybrid nodes to store the blockchain in a succinct way, we employ a clever link structure called the *interlink*. Interlinks were introduced in [23] and further developed in [25].

In the interlink model, a block contains the following information: 1) transactions in the block; 2) the Merkle root x of all the transactions in the block; 3) the Merkle root $y(\text{state}(B_t))$ of the corresponding blockchain state; 4) the block index i ; 5) the *interlink*, which contains hash links to several previous blocks and is described in detail in the following paragraphs; 6) the random nonce η ; and 7) the block hash $\text{id} = H(\eta, x, y, i, \text{interlink})$, where $H()$ is a hash function. For

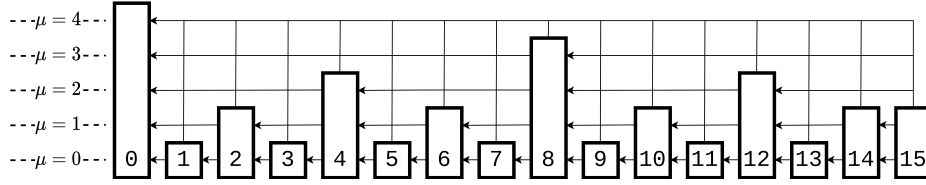


Fig. 1. An example of the interlink structure (inspired by Figure 1 of [11]). At the bottom of each block is its index, and the block’s height signifies its superblock level. Each block has a link to the closest ancestor at every level, which is shown by the arrows linking blocks to ancestors. Notice all the blocks are contained in level 0, and only the genesis block is in level 4.

a block to be valid, id must contain at least T leading 0’s. Equivalently, we say that $\text{id} \leq 2^{-T}$. All the information in the block *except* the list of transactions is referred to as the *block-header*. Observe that the id of the block can be verified given just the block-header.

To describe the interlink, we first need to define *superblocks*. A level- μ superblock is a block with $\text{id} \leq 2^{-(T+\mu)}$. Since a valid block satisfies $\text{id} \leq 2^{-T}$, all valid blocks are level-0 superblocks. The genesis block is defined to be a superblock of every level from 0 to ∞ . And, a level- μ superblock is also a level- μ' superblock for all $0 \leq \mu' \leq \mu$, since $2^{-(T+\mu)} \leq 2^{-(T+\mu')}$.

The **interlink** data-structure in a block contains a link to the previous superblock of level μ for every level μ that is in the chain \mathcal{C}_t up to that block. Since the previous block will always be a superblock of level at least 0, the interlink always contains a link to the previous block (thus, without any further modification, the security properties of the blockchain are unaffected). Also, since the genesis block is of all possible levels, a link to the genesis block is always included. A pictorial example of this is shown in Figure 1.

Using interlinks, it is possible to “skip” over blocks when traversing the blockchain. To be more specific, it is useful to define notation for “traversing the blockchain at level- μ ”. For any given chain \mathcal{C} , the level- μ *upchain*, denoted $\mathcal{C} \uparrow^\mu$, is the sequence of all level- μ superblocks in \mathcal{C} . That is,

$$\mathcal{C} \uparrow^\mu \triangleq \left\{ b : b \in \mathcal{C}, \text{ and } \text{id}(b) \leq 2^{-(T+\mu)} \right\}. \quad (1)$$

Note that although it is convenient to use set-notation to define it, $\mathcal{C} \uparrow^\mu$ is a sequence with the order of its blocks being the same as they are in \mathcal{C} . From the definition of the interlink, each block in the upchain $\mathcal{C} \uparrow^\mu$ contains a reference to the previous block in the upchain. Therefore, it is possible to traverse through $\mathcal{C} \uparrow^\mu$. Additionally, a chain \mathcal{C}' is called a level- μ *superchain* if all its blocks are level- μ superblocks. That is, if the underlying chain of \mathcal{C}' is \mathcal{C} , then $\mathcal{C}' \subseteq \mathcal{C} \uparrow^\mu$.

We use square-brackets to index $\mathcal{C} \uparrow^\mu$, similar to a python array. However, at times it is useful to refer to blocks in $\mathcal{C} \uparrow^\mu$ according to the block’s index in \mathcal{C} . In this case, we use curly-braces to index $\mathcal{C} \uparrow^\mu$. This is best illustrated using an example. Consider, $\mathcal{C} = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}$, and let $\mathcal{C} \uparrow^\mu = \{0, 3, 4, 7, 10\}$.

the PoW of the chain is established by sending the entire chain of block headers to the lightweight client. NIPoPoWs optimize this step by making the following observation. Informally, by the property of concentration around the means, the μ -upchain $\mathcal{C} \uparrow^\mu$ of an underlying chain \mathcal{C} is such that $2^\mu |\mathcal{C} \uparrow^\mu| \approx |\mathcal{C}|$ (as an illustration, in Figure 1, superblocks of level-2 appear roughly every $2^2 = 4$ blocks). Recall that a level- μ superbloc is 2^μ times harder to find than a regular block (i.e., a level-0 superbloc). Therefore, if $2^\mu |\mathcal{C} \uparrow^\mu| \approx |\mathcal{C}|$, then it is as hard for an adversary to create a fork around $\mathcal{C} \uparrow^\mu$ with level- μ superblocs as it is to create a fork around \mathcal{C} with level-0 superblocs. Therefore, it is sufficient to just provide $\mathcal{C} \uparrow^\mu$ as a proof of the PoW of the chain. For a large enough level μ , $\mathcal{C} \uparrow^\mu$ is much smaller in size than the underlying chain \mathcal{C} , thus making NIPoPoWs much faster than traditional protocols.

Our protocol differs from NIPoPoWs in several ways. First, the goal of our protocol is to optimize a hybrid node’s storage while retaining almost all of a full node’s functionalities, while the goal of NIPoPoWs is to provide succinct payment proofs. Second, NIPoPoWs do not optimize the prover’s storage since the prover must still store the entire blockchain. Third, NIPoPoWs are one-time proofs of payment, meaning they need to be generated afresh for every new proof request, whereas our protocol proceeds in an iterative manner throughout the blockchain’s execution. In particular, we employ different *level ranges* (elaborated in the next section) in order to optimize storage throughout time, whereas NIPoPoW only uses a single level range. Fourth, since our end goal is different, our security requirements are different from NIPoPoWs. Lastly, we note that since our security models are different, we use vastly different parameters in our protocol compared to NIPoPoW, and also do novel analysis.

4 Trimming Protocol

In this section we describe our protocol to trim the blockchain. Other associated protocols that compare trimmed-chains to select the main chain, and that verify the blockchain state are described in Appendix C due to lack of space.

First, we start with an intuitive description of the trimmed chain which is best understood by referring to the example in Figure 3. Similar to NIPoPoWs, the high-level idea in our approach is to retain only a subset of (super)blocks in order to represent the proof of work of the chain. Let the trimmed chain be denoted by \mathcal{P}_t (subscript t may be omitted when time is clear from context). \mathcal{P} is a subset of the complete blockchain \mathcal{C} , i.e., $\mathcal{P} \subseteq \mathcal{C}$. And, it has an associated number B'_t called the *trimming point*. All blocks to the right of B'_t are retained, including their data and block headers. That is, $\mathcal{P}\{B' : \} = \mathcal{C}[B' :]$. We refer to $\mathcal{P}\{B' : \}$ as the *untrimmed tail*. Blocks to the left of B' may be trimmed, meaning $\mathcal{P}\{ : B'\} \subseteq \mathcal{C}\{ : B'\}$. Only the block-headers of the blocks in $\mathcal{P}\{ : B'\}$ are retained. The blocks that are not in $\mathcal{P}\{ : B'\}$ are permanently deleted by the hybrid node. Here is where we differ from pruning. In pruning, all the block headers are retained. In trimming, blocks are completely deleted, including their headers.

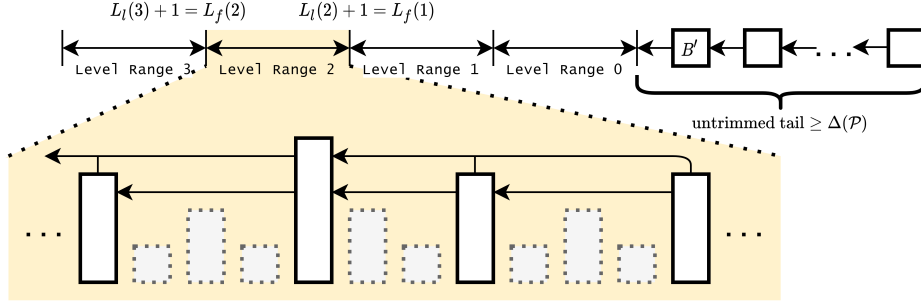


Fig. 3. An example of a trimmed chain at an honest node. Notice how it is partitioned into distinct level ranges, each beginning once its predecessor ends. We show a portion of the second level range, $\mathcal{P}\{L_f(2) : L_l(2)\}$. The greyed blocks with dashed outlines represents blocks that have been trimmed. Like Figure 1, the height of the (super)block is its highest level. Furthermore, after the trimming point, B' , is the untrimmed tail where all the blocks are retained.

The trimmed section of \mathcal{P} is further partitioned into *level-ranges*, each level-range corresponding to a unique level μ . A level-range is spread contiguously over a region of the blockchain, and each level range begins at the point its predecessor ends. We denote the starts and ends of level ranges by *level-range functions* $L_f : \mathbb{Z}^+ \rightarrow \mathbb{Z}^+$ and $L_l : \mathbb{Z}^+ \rightarrow \mathbb{Z}^+$. We define $L_f(\mu)$ as the index of the first block in level-range μ . Similarly, $L_l(\mu)$ is the last block in the level-range μ . Beyond a certain level μ_h , called the highest level, the level-range functions are 0: $L_f(\mu) = L_l(\mu) = 0$ for $\mu > \mu_h$. Also, below a certain level $\mu < \mu_l$, called the lowest level, we have $L_f(\mu) = L_l(\mu) = B' - 1$. Notice then that for $\mu_l \leq \mu < \mu_h$, we have $L_l(\mu + 1) + 1 = L_f(\mu)$. The level ranges are also pictorially shown in the example in Figure 3.

At level-range μ , we are primarily interested in level- μ superblocks. As explained with the intuition of NIPoPoW, we need to weigh level- μ superblocks by 2^μ . In order to avoid confusion with PoW, which traditionally does not look at super-levels, we call this notion of weighted PoW simply as the *weight* at level- μ . For the level-range μ , we denote the weight-function $W(\mathcal{P}, \mu)$ as,

$$W(\mathcal{P}, \mu) = 2^\mu |\mathcal{P}\{L_f(\mu) : L_l(\mu) + 1\} \uparrow^\mu|.$$

Our trimming protocol in Algorithm 1 ensures that a higher level-range precedes a lower-level range (as illustrated in Figure 3). Therefore, we can compute the sum of work from the genesis block up to and including level range μ by, $S(\mathcal{P}, \mu) = \sum_{\mu \leq \mu' \leq \mu_h} W(\mathcal{P}, \mu')$.

Since hybrid nodes do not have access to the underlying chain, $S(\mathcal{P}, \mu)$ can be interpreted as their estimate of the PoW up to block $L_l(\mu)$. Note that $W(\mathcal{P}, \mu)$ and $S(\mathcal{P}, \mu)$ are functions of the level-range functions as well. However, we assume that the level-range functions are implicitly defined by \mathcal{P} in order to keep the notation minimal.

Algorithm 1: Trimming Protocol

```

input           :  $\mathcal{P} ::$  My (trimmed) chain
                   $Q ::$  The trimming interval
                   $\mu_h ::$  Highest level-range in  $\mathcal{P}$ 
                   $L_f(), L_l() ::$  Level-Range functions
                   $g(), f(), \Delta() ::$  Protocol parameter-functions
                   $\text{good}_{\delta, g}() ::$  Good-Superchain function (see Appendix C)

1 on event  $\mathcal{P}$  has grown by  $Q$  blocks since the last trimming attempt:
2    $B' = \mathcal{P}[-1] - \Delta(\mathcal{P})$ 
3   for  $\mu$  from  $\mu_h + 1$  down to 1 do
4      $g \leftarrow g(\mathcal{P}, \mu)$ 
5      $f \leftarrow f(\mathcal{P}, \mu)$ 
6     if  $|\mathcal{P}\{L_f(\mu) : B'\}^{\uparrow\mu}| \geq f$  and  $\text{good}_{\delta, g}(\mathcal{P}\{L_f(\mu) : B'\}^{\uparrow\mu}, \mu)$  then
7        $\mathcal{P}, L_f, L_l, \text{success} \leftarrow \text{trim}(\mathcal{P}, L_f, L_l, B', \mu, g, f)$ 
8       if  $\text{success} = 1$  then
9         break
10  return
11
12 func  $\text{trim}(\mathcal{D}, L_f, L_l, B', \mu, g, f)$ :
13    $\mathcal{E} \leftarrow \mathcal{D}\{L_f(\mu) : B'\}^{\uparrow\mu}$ 
14    $A \leftarrow \mathcal{E}[-f]$ 
15
16   for  $\mu'$  from  $\mu - 1$  to 0 do
17      $\alpha \leftarrow \mathcal{D}\{A : B'\}^{\uparrow\mu'}$ 
18      $\mathcal{E} \leftarrow \mathcal{E} \cup \alpha$ 
19
20     if  $|\alpha| \geq f$  and  $\text{good}_{\delta, g}(\alpha, \mu')$  then
21        $A \leftarrow \alpha[-f]$ 
22
23   if  $|\alpha| \geq f$  and  $\text{good}_{\delta, g}(\alpha, \mu')$  then
24      $L_f(\mu'), L_l(\mu') \leftarrow B' - 1$ , for all  $\mu < \mu'$ 
25      $L_l(\mu) \leftarrow B' - 1$ 
26      $\mathcal{D} \leftarrow \mathcal{D}\{L_f(\mu)\} \cup \mathcal{E} \cup \mathcal{D}\{B' : \}$ 
27     return  $\mathcal{D}, L_f, L_l, 1$ 
28   else
29     return  $\mathcal{D}, L_f, L_l, 0$ 

```

The trimming procedure is detailed in Algorithm 1. We briefly describe it here. Trimming is attempted every time the chain grows by Q blocks (line 1). We call Q the trimming interval. The trimming point is set by the required chain-tail length (line 2). Given that μ_h is the highest level-range in the current trimmed chain, we attempt to trim it further to level $\mu_h + 1$. If trimming to that level is not possible, we try to do it to one level lower and so on (line 3). Trimming to a level μ can be attempted if the specified range of blocks contains enough level- μ superblocks, and if the corresponding μ -upchain is good (line 6).

Roughly, a μ -upchain $\mathcal{C} \uparrow^\mu$ is good if its weight represents the weight of the other level upchains ($|\mathcal{C} \uparrow^\mu| \approx 2^{(\mu-\mu')} |\mathcal{C} \uparrow^{\mu'}|$, $\mu' < \mu$). For the full definition of *good-superchain*, see Appendix C. Given condition on line 6 is satisfied, the **trim** function is called for level μ (line 7).

The **trim** function is very similar to the goodness-aware **Prove** algorithm of the NIPoPoW protocol [25, Algorithm 8]. In the **trim** function, the μ -upchain is obtained first (line 13). Next, the $(\mu - 1)$ -level upchain under the last f blocks of the μ -level upchain is also added (lines 17 and 18). If the $(\mu - 1)$ -level upchain is good, then the $(\mu - 2)$ -level upchain under its last f blocks is added (lines 20 and 21). Otherwise, the $(\mu - 2)$ -level upchain under the last f blocks of the μ -upchain are added. This procedure continues until level 0.

At the end, the **trim** function checks if the trimming was a “success” by checking if level 0 of the trimmed chain is good. The trim being a success means that it is at least as hard for an adversary to create a longer fork around the trimmed chain, as it would be to do so around the complete chain. Intuitively, this is because the necessary levels of the upchains in this range are good, meaning that they represent the PoW of their corresponding downchains.

In case the trim is a success, the trim function along with the new level range functions are returned (lines 23 to 27) indicating that the trim can be used. Otherwise, the old trimmed chain and level range functions are returned (lines 28 and 29).

Chain Selection Hybrid nodes need to have a protocol, $\text{Compare}(\mathcal{C}^{(1)}, \mathcal{C}^{(2)})$, to chose the main chain given two conflicting chains, $\mathcal{C}^{(1)}$ and $\mathcal{C}^{(2)}$. Full nodes (that store the entire blockchain) simply choose the longer of the two chains as the main chain. The chain selection protocol for hybrid nodes is a little more complicated since they do not store the entire chain. At a high level, they use the sum of the cumulative weight, $S(\mathcal{P}, 0)$, of the trimmed portion of a chain and the length of untrimmed section of the chain as a proxy for the chain length. The complete algorithm involves some more details which are described in full in Appendix C.2.

State Verification Similar to CoinPrune [28], a short commitment to the state at the block is stored in every block. Therefore, a hybrid node can verify the correctness of the blockchain’s state by comparing it to the corresponding state commitment. The protocol is called **state – verify** and is presented with more details in Appendix C.3.

Hybrid Node’s Functionalities Here, we describe how a hybrid-node employing the trimming algorithm has the functionalities claimed in Section 1. Since a hybrid node stores the state of the blockchain at its tip, it can perform transaction validation, block validation and state validation. Using the $\text{Compare}(\cdot, \cdot)$ protocol from Appendix C.2, a hybrid node can select the main chain given competing chains. As a consequence, the node can perform mining as well.

A new node joining the system can choose the main chain using the $\text{Compare}(\cdot, \cdot)$ protocol, and verify the state using the state-verify protocol from Appendix C.3. Thus, hybrid nodes can bootstrap other hybrid nodes into the system.

The hybrid node verifies payment proofs as follows. In traditional systems like Bitcoin, the prover provides the chain of block-headers in order to establish the PoW, and then provides a short proof for the transaction’s inclusion (inclusion-proof) in the chain. In NIPoPoWs the prover provides a superchain (which is logarithmic in the size of the underlying chain) in order to establish the PoW. In either case, the hybrid node can use the $\text{Compare}(\cdot, \cdot)$ protocol to compare the given chain (superchain) to its own trimmed chain. If the two chains only differ near the tail and the inclusion-proof is consistent with the provided chain, then the hybrid node approves the prover’s payment proof. Otherwise, it returns false.

5 Security Definitions

Security from Trim Attack Consider the dangerous attack scenario depicted in Figure 4, where the adversary provides a trimmed-chain $\mathcal{P}^{(2)}$ which is “longer” than the honest chain $\mathcal{P}^{(1)}$ (i.e. $\text{Compare}(\mathcal{P}^{(1)}, \mathcal{P}^{(2)}) = \mathcal{P}^{(2)}$) and the LCA between the two chains precedes the honest chain’s trimming point: $b = \text{LCA}(\mathcal{P}^{(1)}, \mathcal{P}^{(2)}) < B^{(1)}$. Denote b_1 to be the (super)block after b in $\mathcal{P}^{(2)}$. The honest node cannot verify the state transition from b to b_1 . This is because the honest node only has access to block headers and not the full state, meaning they can only verify the validity of the block headers, but not of the state transition between the blocks. Moreover, even if the honest node had access to the state at those blocks, there may be a number of blocks between b and b_1 that were skipped during the trimming.

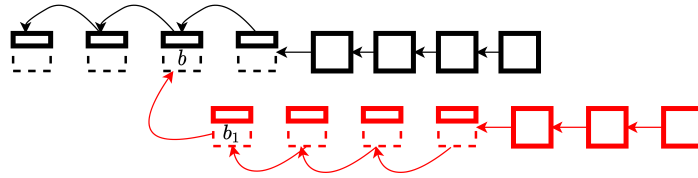


Fig. 4. A trim-attack. The honest chain is shown in black, and adversary’s chain in red. Complete square-boxes indicate the complete blocks beyond the trimming point, and incomplete boxes indicate block-headers of blocks before the trimming point. Curved arrows indicate that the corresponding blocks may not be subsequent blocks.

Therefore, if at any time the adversary is able to create a fork as in Figure 4, they could arbitrarily alter the state of the chain to their advantage. For instance, the adversary could transfer all of the chain’s cryptocurrency into their

own accounts. One approach to circumventing this problem is to have the hybrid node re-download the blockchain from a full node in case it encounters a fork preceding its trimming point. However, in that case the security of the system would again rely on the small number of full nodes. We require that hybrid nodes can work independently from full nodes to keep the system as decentralized as possible. To accomplish this, we ensure that there exists no time when an adversary is able to create a fork from beyond a hybrid node's trimming point.

Definition 1 (Attack on the trimmed Chain). *Let ω be the fixed randomness³. Let $\mathcal{P}_{t,\omega}^{(1)}$ be the honest trimmed chain at some honest node at time t . Let $\mathcal{P}_{t,\omega}^{(2)}$ be the adversary's trimmed chain. Let $B'(\mathcal{P}^{(1)})$ and $B'(\mathcal{P}^{(2)})$ be their respective trimming points. Let $b_{t,\omega}$ be the LCA block between them: $b_{t,\omega} = \text{LCA}(\mathcal{P}_{t,\omega}^{(1)}, \mathcal{P}_{t,\omega}^{(2)})$. Then, we say that the system is **trim-attacked** if there exists a time t such that the adversarial trimmed chain is declared to be longer than the honest trimmed chain and the LCA block is before the honest node's trimming point. That is,*

$$\text{trim-attacked} = \{\omega : \exists t \text{ s.t. } b_{t,\omega} < B'(\mathcal{P}_{t,\omega}^{(1)}), \text{ and } \text{Compare}(\mathcal{P}_{t,\omega}^{(1)}, \mathcal{P}_{t,\omega}^{(2)}) = \mathcal{P}_{t,\omega}^{(2)}\}.$$

Congruence Since hybrid nodes only have access to trimmed chains \mathcal{P} , we need that the selection of the main trimmed-chain made according to $\text{Compare}(\cdot, \cdot)$ is in agreement with the underlying complete chains. We formalize this by a property called *congruence*.

Definition 2 (Congruence). *Given any two (complete) chains $\mathcal{C}^{(1)}$ and $\mathcal{C}^{(2)}$ with corresponding trimmed chains $\mathcal{P}^{(1)}$ and $\mathcal{P}^{(2)}$, they are said to be congruent with each other if, $|\mathcal{C}^{(1)}| > |\mathcal{C}^{(2)}| \implies \text{Compare}(\mathcal{P}^{(1)}, \mathcal{P}^{(2)}) = \mathcal{P}^{(1)}$.*

State Security Hybrid nodes do not have access to transaction history preceding their trimming point. Instead, they rely on the state at the trimming point, $\text{state}(B'_t)$, to compute state at the tip of the chain. State is verified using `state-verify` (Algorithm 3). Firstly, if the adversary launches a trimming attack (Definition 1), then they could change the state of the blockchain arbitrarily. Additionally, the state of the hybrid node is also attacked if at any point the adversary can create a different state, state' , that also passes through `state-verify`.

Definition 3 (Attack on the State). *Let ω be the fixed randomness. At some time t , let the trimming point of an honest chain $\mathcal{P}_{t,\omega}^{(1)}$ be $B'_{t,\omega}^{(1)}$, and let its associated state be $\text{state}^{(1)}(B'_{t,\omega}^{(1)})$. Let the trimming point of the adversary's chain $\mathcal{P}_{t,\omega}^{(2)}$ be $B'_{t,\omega}^{(2)}$, and let its claimed state be $\text{state}_{t,\omega}^{(2)}$. The state of the honest node is said to be **attacked**, denoted by **state-attacked**, if there is either a trim-attack*

³ randomness is w.r.t., the stochastic model described in Appendix B

or there exists a time t such that, $\text{state}_{t,\omega}^{(2)} \neq \text{state}^{(1)}(B'_{t,\omega})$, such that it verifies against the chain. Denoting, $\mathcal{P}_{t,\omega} = \text{Compare}(\mathcal{P}_{t,\omega}^{(1)}, \mathcal{P}_{t,\omega}^{(2)})$, we define,

$$\text{state-attacked} = \{\omega : \omega \in \text{trim-attacked}\}$$

$$\bigcup \{\omega : \exists t \text{ such that } \text{state-verify}(\text{state}_{t,\omega}^{(2)}, \mathcal{P}_{t,\omega}) = 1\}.$$

Bootstrapping Security When a new (honest) node joins the system, it downloads (possibly trimmed) copies of the blockchain from a number of other hybrid or full nodes. It then chooses the main chain using the $\text{Compare}(\cdot, \cdot)$ protocol and then downloads and verifies the state using the state-verify protocol. In order to show that it adopts the honest trimmed chain and the honest state, we define bootstrapping security below.

Definition 4 (Bootstrapping Security). *A joining node is said to be securely bootstrapped into the system if, at the point of it joining, it adopts a trimmed chain that is not trim-attacked with respect to the system's honest chain, and it adopts a state which is not state-attacked with respect to the system's honest state. Otherwise, the node is said to be bootstrap-attacked.*

In our system model, we assume that the mining rate of the honest parties and the adversary remains constant. This might seem to be counter-factual to Definition 4 because it assumes that nodes can join the system. We remark that we make the constant mining rate assumption to make the rigorous security analysis tractable. Practically, we conjecture that if the rate of nodes joining and leaving the system is nearly equal and small enough, then the constant mining rate assumption is a good model for the system.

6 Security Results for Hybrid Nodes

The protocol has security parameters $k, k' \in \mathbb{N}$, $a, c \in \mathbb{R}^+$, and $\delta \in (0, 1)$. Referring to the protocol parameter-functions from Algorithm 1 are defined as: $\Delta(\mathcal{P}) = k' + a \log(S(\mathcal{P}, 0) + |\mathcal{P}\{B' : \cdot\}|)$, $g(\mathcal{P}, \mu) = k + a \log S(\mathcal{P}, \mu)$, and $f(\mathcal{P}, \mu) = c \cdot g(\mathcal{P}, \mu)$. We simply state the results here without proof. The proofs can be found in Appendix E. All the results are with respect to the stochastic model defined in Appendix B.

First, we show if the honest nodes have the majority of mining power, we can choose security parameters such that our protocol is secure against a trim-attack.

Theorem 1 (Security of the trimmed Chain). *Assume an honest majority, $\lambda_h > \lambda_a$, where λ_h and λ_a are the mining rates of the honest and adversarial nodes respectively. Let the trimming algorithm (Algorithm 1) parameters k, k', c, a, δ satisfy,*

$$k' = k - a \log(((1 + \delta^2) + \delta)/\delta), \quad 1 < (1 - \delta)^3 \frac{\lambda_h}{\lambda_a} \frac{c' - 1}{c'} \frac{c - 1 - c'}{c},$$

$$1 < (1 - \delta)^5 \frac{\lambda_h}{\lambda_a} \frac{c - 1}{c}, \quad a \geq \frac{8}{\delta^2},$$

where $2 < c' < c$ is some constant. Then⁴, $\mathbb{P}(\text{trim-attacked}) = \text{negl}(k)$.

As a corollary to the above theorem, we can conclude the chain-selection made by hybrid-nodes is consistent with underlying chain lengths.

Corollary 1. *The congruence property holds except with probability $\text{negl}(k)$.*

Theorem 2 (State Security). *Assuming honest majority, $\lambda_h > \lambda_a$, and that the security parameters are as in Theorem 1, then, $\mathbb{P}(\text{state-attacked}) = \text{negl}(k)$.*

Corollary 2 (Bootstrapping Security). *Assume the arrival process of new nodes is independent of the randomness of the blockchain system. If we further assume that a new node contacts at least one honest node and that there is honest mining majority, then **bootstrap-attacked** occurs with probability $\text{negl}(k)$.*

In the next theorem, we provide an upper bound on the cold-storage required to store the trimmed chain. This theorem relies on the assumption that a particular kind of adversary is absent. We further comment on this in Remark 1. Additionally, we also note that the following theorem doesn't account for the storage required to store the state at the trimming point. We discuss optimizing the storage for the state using stateless blockchains in Section 7.

Theorem 3 (Optimistic Succinctness). *Assuming that all the nodes are honest, the cold-storage requirements for a hybrid node to store the trimmed-chain \mathcal{P} is $O(\log^4 B)$ with high probability in k . Here, B is the length of the underlying chain \mathcal{C} .*

Remark 1 (Need of Optimism for Succinctness). Since our protocol is an extension of NIPoPoWs, we need to rely on optimism for succinctness as well. In particular, note that when trimming we need to ensure that the level ranges are “good”. We have also shown that under honest behaviour, the sizes of the different upchains concentrate around the mean, thus leading to good level-ranges with high probability. However, an adversary with a small mining power could launch an attack which hampers the concentrations around the means and thus makes good-superchains more unlikely. This attack is formally described in [25]. We note that, just like in NIPoPoWs, the adversary can only hurt the succinctness of our model, but not its security. Although the economics of launching an attack on the hybrid node's succinctness is yet to be studied formally, we venture to guess that such an attack wouldn't be economically viable to the adversary.

Clearly, the size of the untrimmed tail is a lower bound on the cold-storage required. In the next theorem we prove that the length of the chain tail has to be at least $\log B$ in order to protect a hybrid node against a **trim-attack**. This implies that our protocol is near-optimal in terms of cold storage, since it only requires $\text{polylog}(B)$ storage.

Theorem 4 (Vulnerability of Short Tails). *Let the blockchain length be B_t at time t , and the trimming point be B'_t . If at all times t , $\mathbb{P}(B_t - B'_t = o(\log B_t)) > \epsilon$, for some $\epsilon > 0$, then there exists an adversary with small mining power, $\lambda_a < \lambda_h$, such that $\mathbb{P}(\text{trim-attacked}) = 1$.*

⁴ In our work, $\text{negl}(k) = e^{-\Omega(k)} < \frac{1}{\text{poly}(k)}$.

7 Optimizing State Storage with Stateless Blockchains

Our trimming protocol optimizes the amount of cold storage required to represent the PoW in the blockchain. In this section, we outline how our work can interface with methods optimizing the storage of the *state*. Furthermore, a detailed description is found in App. J. In hybrid nodes, states need to be stored in cold and hot storage. Along with the length of the blockchain, the size of the blockchain-state is rapidly increasing with time. For instance, at the time of writing Bitcoin’s UTXO state is almost 4GB in size, which motivates lessening high storage and verification time by extending our work to optimize both the state’s cold-storage at the trimming point, and its hot-storage at the chain tip.

We described that storing state-commitments in the blocks enables one to securely establish the state of a trimmed chain, by use of the **state-verify** protocol described in Algorithm 3. In a different line of work, called *stateless blockchains*, state-commitments are used to reduce the amount of hot-storage required for transaction validation, thus speeding up validation. We claim that when hybrid nodes are used in stateless-blockchains, then state commitments can serve a dual purpose: 1) they can be used to establish the state of the trimmed chain; 2) and, they can be used to perform stateless transaction validation.

Stateless validation, first proposed by Todd [38], is a scheme where nodes validating transactions store a short cryptographic state-commitment rather than the entire state. A client then provides a membership proof that the node can verify against the commitment. Recently, several constructions have been developed to perform stateless validation in both the UTXO and account-based state models using various cryptographic primitives [10, 13, 4, 39, 34, 1, 19].

Stateless blockchains optimize hot-storage by obviating the need to store the state at the tip of the chain in the RAM. At this point, the state at the trimming point is still stored in the cold storage. We can use this to provide an interface from trimming to stateless blockchains: First, if the proof for some client becomes outdated past the trimming point because it was offline for a long time, then the client can recompute its latest proof by querying a hybrid node for $\text{state}(B'_t)$ and the untrimmed chain tail. Second, if a fork in the chain makes the proofs of several clients invalid, they can all recompute their proofs by contacting the hybrid node similarly. In App. J, we describe a way to optimize the cold-storage of the state at the trimming point as well.

8 Future Directions

In this paper, we presented hybrid nodes which use trimming to optimize the cold-storage required to represent the PoW in the chain. When used in conjunction with stateless blockchains, we illustrated that hybrid nodes are optimized both in terms of cold and hot storage. In this section, we lay out some directions for future work.

We assume that there is honest majority, $\lambda_h > \lambda_a$, throughout the execution of the blockchain protocol. Given that we use novel security models, it will be interesting to study if it is economically viable for an adversary to acquire massive

resources in order to attain a majority mining power temporarily, and launch an attack such as a *trim – attack*. Similarly, it will be interesting to study the economics of an adversary who simply attempts to hurt the succinctness of hybrid nodes. In hurting the succinctness, fewer nodes would join as hybrid nodes. Thus, an adversary could attempt to take advantage of the reduced decentralization.

We have proved in Theorem 4, that our protocol is near-optimal in terms of the storage it requires to represent a blockchain’s PoW if it is to be secure against a *trim – attack*. However, it is at the moment unclear if further storage optimization in terms of storing the state is possible for a node that needs to have all the functionalities of a hybrid node (without assuming properties on the clients as we do in Section 7).

In our analysis, we assume that the block *id*’s have a constant difficulty target throughout the blockchain’s execution. It may be useful to relax this assumption in order to make our model closer to practical systems.

Finally, our work in this paper has mainly focused on theoretical analysis. An interesting line of work would be to implement the protocol and study the practical gains of using hybrid nodes.

References

- [1] Agrawal, S. and Raghuraman, S. “KVAC: Key-Value Commitments for Blockchains and Beyond”. In: *International Conference on the Theory and Application of Cryptology and Information Security*. Springer. 2020, pp. 839–869.
- [2] *Bitcoin Hash Rate*. Available at <https://www.coinwarz.com/mining/bitcoin/hashrate-chart> (2021/04/29).
- [3] *Bitcoin UTXO size*. Available at <https://tinyurl.com/cr7w2ep5> (2021/09/13).
- [4] Boneh, D., Bünz, B., and Fisch, B. “Batching techniques for accumulators with applications to iops and stateless blockchains”. In: *Annual International Cryptology Conference*. Springer. 2019, pp. 561–586.
- [5] Bonneau, J. et al. “Coda: Decentralized Cryptocurrency at Scale.” In: *IACR Cryptol. ePrint Arch.* 2020 (2020), p. 352.
- [6] Bünz, B. et al. “Flyclient: Super-light clients for cryptocurrencies”. In: *2020 IEEE Symposium on Security and Privacy (SP)*. IEEE. 2020, pp. 928–946.
- [7] Canonne, C. *A Short Note on Poisson Tail Bounds*. Available at <http://www.cs.columbia.edu/~ccanonne/files/misc/2017-poissonconcentration.pdf> (2021/05/18).
- [8] Catalano, D. and Fiore, D. “Vector commitments and their applications”. In: *International Workshop on Public Key Cryptography*. Springer. 2013, pp. 55–72.
- [9] Chen, W. et al. *Reducing Participation Costs via Incremental Verification for Ledger Systems*. Tech. rep. Cryptology ePrint Archive, Report 2020/1522, 2020.

- [10] Chepurnoy, A., Papamanthou, C., and Zhang, Y. “Edrax: A Cryptocurrency with Stateless Transaction Validation.” In: *IACR Cryptol. ePrint Arch.* 2018 (2018), p. 968.
- [11] Daveas, S. et al. “A Gas-Efficient Superlight Bitcoin Client in Solidity”. In: *Proceedings of the 2nd ACM Conference on Advances in Financial Technologies*. AFT ’20. New York, NY, USA: Association for Computing Machinery, 2020, pp. 132–144. ISBN: 9781450381390. DOI: 10.1145/3419614.3423255. URL: <https://doi.org/10.1145/3419614.3423255>.
- [12] Dembo, A. et al. “Everything is a race and nakamoto always wins”. In: *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*. 2020, pp. 859–878.
- [13] Dryja, T. “Utreexo: A dynamic hash-based accumulator optimized for the Bitcoin UTXO set.” In: *IACR Cryptol. ePrint Arch.* 2019 (2019), p. 611.
- [14] Dubhashi, D. and Panconesi, A. “Concentration of measure for the analysis of randomised algorithms”. In: *Draft Manuscript*, <http://www.brics.dk/ale/papers.html> (1998).
- [15] *Ethereum Hash Rate*. Available at <https://www.coinwarz.com/mining/ethereum/hashrate-chart> (2021/04/29).
- [16] Eyal, I. and Sirer, E. G. “Majority is not enough: Bitcoin mining is vulnerable”. In: *International conference on financial cryptography and data security*. Springer. 2014, pp. 436–454.
- [17] Garay, J., Kiayias, A., and Leonardos, N. “The bitcoin backbone protocol: Analysis and applications”. In: *Annual international conference on the theory and applications of cryptographic techniques*. Springer. 2015, pp. 281–310.
- [18] Gazi, P., Kiayias, A., and Russell, A. “Tight consistency bounds for bitcoin”. In: *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*. 2020, pp. 819–838.
- [19] Gorbunov, S. et al. “Pointproofs: aggregating proofs for multiple vector commitments”. In: *Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security*. 2020, pp. 2007–2023.
- [20] Kadhe, S., Chung, J., and Ramchandran, K. “SeF: A Secure Fountain Architecture for Slashing Storage Costs in Blockchains”. In: *CoRR* abs/1906.12140 (2019). arXiv: 1906.12140. URL: <http://arxiv.org/abs/1906.12140>.
- [21] Karantias, K., Kiayias, A., and Zindros, D. “Compact storage of superblocks for nipopow applications”. In: *Mathematical Research for Blockchain Economy*. Springer, 2020, pp. 77–91.
- [22] Kattis, A. and Bonneau, J. “Proof of Necessary Work: Succinct State Verification with Fairness Guarantees.” In: *IACR Cryptol. ePrint Arch.* 2020 (2020), p. 190.
- [23] Kiayias, A., Lamprou, N., and Stouka, A.-P. “Proofs of proofs of work with sublinear complexity”. In: *International Conference on Financial Cryptography and Data Security*. Springer. 2016, pp. 61–78.

- [24] Kiayias, A., Leonardos, N., and Zindros, D. *Mining in Logarithmic Space*. Cryptology ePrint Archive, Report 2021/623. <https://ia.cr/2021/623>. 2021.
- [25] Kiayias, A., Miller, A., and Zindros, D. “Non-interactive proofs of proof-of-work”. In: *International Conference on Financial Cryptography and Data Security*. Springer. 2020, pp. 505–522.
- [26] Kiayias, A., Polydouri, A., and Zindros, D. “The Velvet Path to Superlight Blockchain Clients”. In: (2020).
- [27] Li, S. et al. “PolyShard: Coded Sharding Achieves Linearly Scaling Efficiency and Security Simultaneously”. In: *Trans. Info. For. Sec.* 16 (Jan. 2021), pp. 249–261. ISSN: 1556-6013. DOI: 10.1109/TIFS.2020.3009610. URL: <https://doi.org/10.1109/TIFS.2020.3009610>.
- [28] Matzutt, R. et al. “How to Securely Prune Bitcoin’s Blockchain”. In: *2020 IFIP Networking Conference (Networking)*. IEEE. 2020, pp. 298–306.
- [29] *Nodes and Clients*. Available at <https://ethereum.org/en/developers/docs/nodes-and-clients/> (2021/05/20).
- [30] Pass, R., Seeman, L., and Shelat, A. “Analysis of the blockchain protocol in asynchronous networks”. In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2017, pp. 643–673.
- [31] Perard, D. et al. “Erasure Code-Based Low Storage Blockchain Node”. In: *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCoM) and IEEE Smart Data (SmartData)*. 2018, pp. 1622–1627. DOI: 10.1109/Cybermatics_2018.2018.00271.
- [32] Raman, R. K. and Varshney, L. R. “Dynamic Distributed Storage for Scaling Blockchains”. In: *CoRR* abs/1711.07617 (2017). arXiv: 1711.07617. URL: <http://arxiv.org/abs/1711.07617>.
- [33] Reddy, B. S. “securePrune: Secure block pruning in UTXO based blockchains using Accumulators”. In: *2021 International Conference on COMMunication Systems & NETWORKS (COMSNETS)*. IEEE. 2021, pp. 174–178.
- [34] Reyzin, L. et al. “Improving authenticated dynamic dictionaries, with applications to cryptocurrencies”. In: *International Conference on Financial Cryptography and Data Security*. Springer. 2017, pp. 376–392.
- [35] *Running a Full Node*. Available at <https://bitcoin.org/en/full-node> (2021/05/20).
- [36] Sankagiri, S., Gandlur, S., and Hajek, B. “The Longest-Chain Protocol Under Random Delays”. In: *arXiv preprint arXiv:2102.00973* (2021).
- [37] Sompolinsky, Y. and Zohar, A. “Bitcoin’s security model revisited”. In: *arXiv preprint arXiv:1605.09193* (2016).
- [38] Todd, P. *Making UTXO Set Growth Irrelevant With Low-Latency Delayed TXO Commitments*. Available at <https://petertodd.org/2016/delayed-txo-commitments> (2021/05/24).

- [39] Tomescu, A. et al. “Aggregatable subvector commitments for stateless cryptocurrencies”. In: *International Conference on Security and Cryptography for Networks*. Springer. 2020, pp. 45–64.
- [40] Valiant, P. “Incrementally verifiable computation or proofs of knowledge imply time/space efficiency”. In: *Theory of Cryptography Conference*. Springer. 2008, pp. 1–18.

A Other Related Works

Coding Based Storage Reduction Several recent works have employed erasure-codes to reduce cold storage costs in blockchains [31, 27, 20, 32]. The core idea in these works is that each node stores a coded version of a random subset of the blockchain. A new joining node can then download the coded fragments from several nodes and reconstruct the entire blockchain. Although this does lead to a significant reduction in storage requirements, the storage in these methods has an information theoretic lower bound of $\Omega(B/N)$, where B is the length of the blockchain and N is the number of nodes [27]. In our proposed method, the cold storage cost scales *poly-logarithmically* in the length of the blockchain.

Incrementally Verifiable Computation Incrementally verifiable computation (IVC) is an orthogonal approach, initially proposed for general settings by Valiant [40]. IVC's provide a succinct argument of knowledge that some state is correctly computed from the initial state of computation (and all preceding states of computation) by performing a recursive composition of state transitions. In the context of blockchains, [22, 5, 9] employ IVC to reduce state verification time. For the sake of comparison, we note that these methods are based on generic techniques, like applying a Succinct Non-interactive ARgument of Knowledge (SNARK) on a suitable compliance predicate and recursively composing such SNARKs, which result in concretely high overhead costs for the prover. Furthermore, in [22], which is most directly related to our work amongst the three, the authors acknowledge that their approach does not lessen storage requirements of mining nodes. In contrast, the hybrid nodes which have mining capability and more.

Lightweight Clients with Sublinear Complexity Recall lightweight clients only seek to verify payment proofs. Apart from NIPoPoW, another popular method that has sublinear complexity for lightweight clients is *FlyClient* [6]. It uses probabilistic block sampling and a novel commitment scheme called merkle-mountain-ranges. FlyClient is more versatile than NIPoPoW's in that it doesn't require optimism for succinctness, and it allows for varying block difficulties. Unfortunately, at the moment, we don't believe the FlyClient protocol can be used for hybrid nodes. In addition to theoretical works, there have been recent works that have also studied practical aspects of NIPoPoW and FlyClient [21, 11, 26].

B Additional Model Details

B.1 Communication Model

Several communication models are considered in the literature. The simplest is the *synchronous* model where a block broadcast by a node at a certain time is received by all other nodes immediately [17, 16]. Since time is continuous, no more than one block is mined at any given time, implying only one block could be

in communication at any given time. More complicated communication models with communication delays are also considered in the literature [30, 36]. For the sake of simplicity, we consider the synchronous model in this paper and leave it to future work to transfer our results to more complicated communication models.

B.2 Additional Notation

We define the notion of a *downchain* slightly different from [23]. Let $\mathcal{C}' \subseteq \mathcal{C} \uparrow^\mu$ be a level- μ superchain. Let $\mathcal{C}'[0] = \mathcal{C} \uparrow^\mu [j]$. Then, the corresponding downchain is defined as,

$$\mathcal{C}' \downarrow \triangleq \begin{cases} \mathcal{C}[\mathcal{C} \uparrow^\mu [j-1] + 1 : \mathcal{C}'[-1]], & \text{if } j > 0, \\ \mathcal{C}[: \mathcal{C}'[-1]], & \text{if } j = 0. \end{cases} \quad (2)$$

B.3 Stochastic Model for the Blockchain.

In order to study the behavior of the blockchain, we consider a stochastic model for the blockchain which has been used previously in the literature to study blockchain security [12, 18, 37]. The stochastic model allows us to abstract away the cryptographic details of PoW mining and focus specifically on the dynamics of the blockchain growth. In particular, we model PoW mining as calls to an ideal random functionality. At each query, independent of everything else, the ideal functionality returns “success” with a small probability p and “failure” otherwise. After describing our protocol and its analysis, we interpret our results under popular cryptographic models such as the model used by Pass., et al [30] and the Bitcoin Backbone Model [23] in Section I.

Since we use a continuous time model, we represent mining as a Poisson process as has been used in previous works [18, 12]. Let Ω be the sample-space, and let $\omega \in \Omega$ denote an outcome (also called fixed randomness). Let the overall mining process of honest nodes be a Poisson process of rate λ_h and let the adversary’s mining be a Poisson process of rate λ_a . The honest and adversarial processes are independent of each other. Honest blocks add their mined blocks to the tip of their longest chains and immediately broadcast the block. However, the adversary could add their mined block at any point on the chain. Moreover, the adversary may choose to keep their mined blocks secret and broadcast them at any future time of their choice. Once a block is broadcast, all the honest nodes receive it immediately. Honest nodes are said to have a *majority* if $\lambda_h > \lambda_a$.

Observe that a level- μ superblock is 2^μ times harder to find than a regular block. In other words, a regular block is a level- μ superblock with probability $2^{-\mu}$. Therefore, the arrival process for honest (adversarial) level- μ superblocks is a Poisson process of rate $\lambda_h/2^\mu$ ($\lambda_a/2^\mu$). Here as well, the honest and adversarial arrival processes are independent of each other. However, we note that the arrival processes for honest (adversarial) level- μ_1 and level- μ_2 superblocks are not independent even if $\mu_1 \neq \mu_2$. This is because a level- μ_1 superblock is also a level- μ_2 superblock if $\mu_1 \geq \mu_2$.

Next, we also capture the state-commitment process using the ideal random functionality. Recall that we include the Merkle-root $y(\text{state})$ of the **state** at that point in the block header. An adversary could attempt to produce a malicious state, $\text{state}' \neq \text{state}$, which also produces the same Merkle-root $y(\text{state})$. During this attempt, the adversary would query the tuple $(\text{state}', y(\text{state}))$ to the random functionality and receive “success” with a very small probability q and “failure” otherwise, independently of everything else. If the adversary receives a success, we say $\text{state}' \equiv \text{state}$. Here, q can be assumed to be very small compared to the block success probability p . This is because a block requires only the first T bits of the hash-string to be 0, whereas in the case of the malicious state, the entire hash-string has to match the original commitment y . In other words, the adversary must find a *collision* on the hash function used in the Merkle tree, which is assumed to occur only with negligible probability with respect to p . Assume that the adversary tries to find a malicious state at the tip of the chain at some time t . By time $s \geq t$, let $N_t(s)$ denote the number of subsequent blocks’ states for which the adversary’s malicious state sequence is equivalent to, i.e.,

$$F^n(\text{state}'(\mathcal{C}_t)) \equiv F^n(\text{state}(\mathcal{C}_t)) \quad , \text{ for all } 0 \leq n < N_t(s).$$

Then, $N_t(s)$ is a Poisson process of rate λ_s , where $\lambda_s \ll \lambda_a$ because $q \ll p$.

C Our Protocols: Further Details

The notion of a superchain’s weight being approximately equal to the PoW of the corresponding downchain was formalized as *superquality* in [25]. Adding to this idea, we define a notion of a dominant superchain below.

Good Superchain First, we require that every sufficiently large suffix of the upchain weighs at least $(1 - \delta)$ times the amount of PoW of its corresponding downchain.

Definition 5 (Superquality). *Let \mathcal{C} be a blockchain, and $\mathcal{C} \uparrow^\mu$ be its level- μ upchain. $\mathcal{C} \uparrow^\mu$ is said to have (δ, g) superquality, for $\delta > 0$ and $g \in \mathbb{N}$, if for all $g' \geq g$, the following is true:*

$$|\mathcal{C} \uparrow^\mu [-g' :]| \geq (1 - \delta) 2^{-\mu} |\mathcal{C} \uparrow^\mu [-g' :]| \downarrow.$$

Next, we define a dominant superchain.

Definition 6 (Dominant Superchain). *Consider a μ -superchain \mathcal{C}' of an underlying chain, $\mathcal{C} = \mathcal{C}' \downarrow$, and parameter $g \in \mathbb{N}$. \mathcal{C}' is said to be a dominant superchain with parameter g if the following is true. Let*

$\mathcal{P}' \subseteq \mathcal{C}$ be any trimmed chain with associated level range functions L'_f and L'_l . Let, $\mu_1 \leq \mu' \leq \mu$, and $L'_f(\mu') \leq l \leq L'_l(\mu')$. Then,

$$2^{\mu_1} |\mathcal{P}' \{l : L'_l(\mu') + 1\} \uparrow^{\mu_1}| + \sum_{\mu'' < \mu'} 2^{\mu''} |\mathcal{P}' \{L'_f(\mu'') : L'_l(\mu'') + 1\} \uparrow^{\mu''}| \geq 2^\mu g, \quad (3)$$

implies that,

$$|\mathcal{P}' \uparrow^\mu| \geq 1.$$

As intuition, first note that \mathcal{C}' and \mathcal{P}' have the same underlying downchain. We are interested in the weight of \mathcal{P}' to the right of a block l in level range μ' . For reasons that will be clear from our trimming and chain-compare protocols (Algorithms 1 and 2), we allow weighing the first level range μ' using a different level $\mu_1 \leq \mu'$. Then, the dominant-superchain property says that if \mathcal{P}' has a weight that is large enough, then it contains at least 1 level- μ superblock.

A superchain is a *good-superchain* if it has superchain quality and is a dominant superchain.

Definition 7 (Good Superchain). A μ -superchain \mathcal{C}' of an underlying chain $\mathcal{C} = \mathcal{C}' \downarrow$, is a good-superchain, denoted $\text{good}_{\delta,g}(\mathcal{C}', \mu)$, if it satisfies superquality with parameters (δ, g) and is a dominant superchain with parameter g .

Intuitively, if all the respective quantities above are equal to their mean values, then $\mathcal{C} \uparrow^\mu$ is a good superchain. As long as an adversary does not try to violate the concentration around the means, a superchain is a good superchain with high probability. The proof for superquality can be found in [25]. The proof for dominant superchain property can be found in Appendix G.

C.1 Trimming

With the notion of a good superchain defined, our trimming protocol description from Section 4 can be completed by specifying the parameter functions used in Algorithm 1. We next describe our trimming protocol. The protocol has security parameters $k, k' \in \mathbb{N}$, $a, c \in \mathbb{R}^+$, and $\delta \in (0, 1)$. The length of the untrimmed tail that is left untouched is dictated by the function $\Delta(\mathcal{P})$. If we denote B' to be the old trimming point, then let⁵,

$$\Delta(\mathcal{P}) = k' + a \log(S(\mathcal{P}, 0) + |\mathcal{P}\{B' : \cdot\}|). \quad (4)$$

When $\mathcal{P}\{B' : \cdot\}$ is trimmed to a certain level μ , we first ensure that the relevant upchain is a good superchain. The parameter g used here for level μ is given by,

$$g(\mathcal{P}, \mu) = k + a \log S(\mathcal{P}, \mu). \quad (5)$$

⁵ Unless otherwise mentioned, logarithms are to the base e .

Apart from requiring that the μ -upchain is a good-superchain, we require that there are “enough” number of superblocks of that level μ . This is given by

$$f(\mathcal{P}, \mu) = c \cdot g(\mathcal{P}, \mu) \quad (6)$$

We require the functions Δ, g and f to depend on the logarithm of the estimated PoW to obtain strong concentration bounds, which achieves security properties. As far as we know, this notion was first used to obtain strong security for blockchains in [37] where the authors call it “logarithmic waiting time”.

C.2 Comparing Work in trimmed Chains

Hybrid nodes need to have a protocol, $\text{Compare}(\mathcal{C}^{(1)}, \mathcal{C}^{(2)})$, to chose the main chain given two conflicting chains, $\mathcal{C}^{(1)}$ and $\mathcal{C}^{(2)}$. Full nodes (that store the entire blockchain) simply choose the longer of the two chains as the main chain. The chain selection protocol for hybrid nodes is a little more complicated since they do not store the entire chain. At a high level, they use the sum of the cumulative weight, $S(\mathcal{P}, 0)$, of the trimmed portion of a chain and the length of untrimmed section of the chain as a proxy for the chain length. Next, we explain this algorithm more formally.

Let $\mathcal{P}^{(1)}$ and $\mathcal{P}^{(2)}$ be two trimmed chains with underlying (complete) chains $\mathcal{C}^{(1)}$ and $\mathcal{C}^{(2)}$. We need a protocol, $\text{Compare}(\mathcal{P}^{(1)}, \mathcal{P}^{(2)})$, that outputs $\mathcal{P}^{(1)}$ if $|\mathcal{C}^{(1)}| > |\mathcal{C}^{(2)}|$, otherwise it outputs $\mathcal{P}^{(2)}$. We allow error with small probability given by the security definitions in Section 5.

The protocol is shown in Algorithm 2 and we summarize it here. First, only the parts of the chains after their LCA block need to be compared (line 1). The algorithm assigns weights to each chain, denoted $\text{Weight}(\mathcal{P})$, and declares the chain with the higher weight the winner (line 5). The weight is assigned to (the non-common part of) a trimmed chain as follows. If the chain doesn’t have a long enough untrimmed tail, then its weight is set to 0 (line 8). If the LCA block b is after the trimming point, then the weight is set to the number of blocks after b (lines 11 and 12). Otherwise, the level range in which b lies is computed (line 14). For the first level range, a set M of levels is formed with enough number of superblocks of that level (line 16). Then, the highest PoW among the upchains in set M is considered (line 17). For all subsequent level-ranges, the PoW corresponding their level is added (lines 18 to 20). At the end, the PoW of the untrimmed-tail is added and returned (line 22).

The $\text{Compare}(\cdot, \cdot)$ algorithm is used in two ways. First, by hybrid-nodes to compute the main chain when they receive another (trimmed) chain which is not the same as their trimmed chain. Second, by new hybrid-nodes to bootstrap into the system. The new node receives (trimmed) chains from other nodes, and then calls $\text{Compare}(\cdot, \cdot)$ on pairs of chains, until it is left with 1 winner.

C.3 State Verification

Similar to CoinPrune [28], a short commitment to the state at the block is stored in every block. Therefore, a hybrid node can verify the correctness of the

Algorithm 2: Compare(\cdot, \cdot) Protocol to compare work in trimmed chains

input : $\mathcal{P}^{(1)}, L_f^{(1)}, L_l^{(1)}, B'^{(1)} :: \text{Chain 1}$
 $\mathcal{P}^{(2)}, L_f^{(2)}, L_l^{(2)}, B'^{(2)} :: \text{Chain 2}$

- 1 $b = \text{LCA}(\mathcal{P}^{(1)}, \mathcal{P}^{(2)})$
- 2 $W^{(1)} = \text{Weight}(\mathcal{P}^{(1)}, L_f^{(1)}, L_l^{(1)}, B'^{(1)}, b)$
- 3 $W^{(2)} = \text{Weight}(\mathcal{P}^{(2)}, L_f^{(2)}, L_l^{(2)}, B'^{(2)}, b)$
- 4
- 5 **return** $\text{argmax}(W^{(1)}, W^{(2)})$
- 6
- 7 **func** $\text{Weight}(\mathcal{P}, L_f, L_l, B', b)$:
- 8 **if** $|\mathcal{P}\{B' : \}| < \Delta(g(\mathcal{P}, 0) + |\mathcal{P}\{B' : \}|)$ **then**
- 9 **return** 0
- 10
- 11 **if** $b \geq B'$ **then**
- 12 **return** $|\mathcal{P}\{b : \}|$
- 13
- 14 $\hat{\mu} \leftarrow \text{such that } L_f(\hat{\mu}) \leq b \leq L_l(\hat{\mu})$
- 15
- 16 $M = \left\{ \mu' : |\mathcal{P} \uparrow^{\mu'} \{b : L_l(\hat{\mu}) + 1\}| \geq f(\mathcal{P}, \hat{\mu}) \right\}$
- 17 $W = \max_{\mu' \in M \cup \{0\}} \left\{ 2^{\mu'} |\mathcal{P} \uparrow^{\mu'} \{b : L_l(\hat{\mu}) + 1\}| \right\}.$
- 18 **for** μ *from* $\hat{\mu} - 1$ *to* 0 **do**
- 19 **if** $|\mathcal{P} \uparrow^{\mu} \{L_f(\mu) : L_l(\mu) + 1\}| \geq f(\mathcal{P}, \mu)$ **then**
- 20 $W = W + 2^{\mu} |\mathcal{P} \uparrow^{\mu} \{L_f(\mu) : L_l(\mu) + 1\}|.$
- 21
- 22 **return** $W + |\mathcal{P}\{B' : \}|$

blockchain's state by comparing it to the corresponding state commitment. The protocol is called **State – Verify** and is presented with more details in Algorithm 3, and it is summarized in the following paragraph.

After a new node adopts a trimmed chain \mathcal{P} with trimming point B' , it needs to download the state, $\text{state}(B')$, at its trimming point. Since all blocks to the right of the trimming point are retained, the node can compute all the states corresponding to blocks in the untrimmed tail. In order to verify the state, the node checks whether the Merkle-root of the computed state for each block in the untrimmed tail matches its corresponding Merkle-root y stored within the block-header. If the Merkle-roots do not match at any block in the untrimmed tail, then the algorithm returns False. This is shown in Algorithm 3.

Algorithm 3: State – Verify Protocol to verify the state of a trimmed chain

input $\mathcal{P} ::$ trimmed chain
 $B' ::$ trimming point
 $\text{state}(B') ::$ State at trimming point

1 **for** i from 0 to $B - B'$ **do**
2 Compute $\text{state}(B' + i)$
3 **if** $\text{Merkle} - \text{root}(\text{state}(B' + i)) \neq y(B' + i)$ **then**
4 **return** False
5 **return** True

D Poisson Process Preliminaries

X is a Poisson random variable with rate λ , denoted $X \sim \text{Poisson}(\lambda)$, if its probability distribution satisfies,

$$\mathcal{P}(X = x) = e^{-\lambda} \frac{\lambda^x}{x!}, \quad \text{for } x \in \mathbb{Z}^+.$$

Let $A(t)$ be a random process indexed by time $t \geq 0$. Denote $A(t_1, t_2) = A(t_2) - A(t_1)$, where $t_2 \geq t_1$. A is a Poisson point process (or simply, Poisson process) of rate λ , denoted as $A \sim PPP(\lambda)$, if,

$$A(t) \sim \text{Poisson}(\lambda t), \text{ for all } t \geq 0,$$

and for disjoint intervals (t_1, t_2) and (t_3, t_4) ,

$$A(t_1, t_2) \text{ is independent of } A(t_3, t_4).$$

Since the range of A is the positive integers, and can be shown to be non-decreasing with time, it can be thought of as a counting process. We also note the fact the time between two successive events in A is called the interarrival time, and it is distributed as an exponential random variable of rate λ .

First, we recall concentration bounds for a Poisson random variable, without proof. The proof is obtained by applying Chernoff Bound [7].

Lemma 1. *Let $X \sim \text{Poisson}(\lambda)$, for some $\lambda > 0$. Then, for any $x > 0$,*

$$\begin{aligned} \mathbb{P}(X \geq \lambda + x) &\leq e^{-\frac{x^2}{2(\lambda+x)}}, \\ \mathbb{P}(X \leq \lambda - x) &\leq e^{-\frac{x^2}{2(\lambda+x)}}. \end{aligned}$$

In the following lemma, we upper bound and lower the time required for n events to occur in a Poisson process.

Lemma 2. *Let $A \sim PPP(\lambda)$. Let T be a random variable indicating the time taken for n events to occur in the Poisson process. That is, $A(T) = n$. Then, except with probability $e^{-n\frac{\delta'^2}{2}}$, $T \leq \frac{n(1+\delta')}{\lambda}$. Similarly, except with probability $e^{-n\frac{\delta'^2}{2}}$, $T \geq \frac{n(1-\delta')}{\lambda}$.*

Proof. Recall that in Poisson point process, interarrival times are i.i.d., exponential. Therefore, for $X_1, \dots, X_n \stackrel{iid}{\sim} \exp(\lambda_1)$, $T \stackrel{d}{=} \sum_{i=1}^n X_i$. We can use Chernoff bound on T as follows:

$$\begin{aligned} \mathbb{P}\left(T \geq \frac{n(1+\delta')}{\lambda_1}\right) &\leq e^{-n \sup_{s>0} \left(\frac{s(1+\delta')}{\lambda_1} + \log(1-s/\lambda_1)\right)}, \\ &= e^{-n(\delta' - \log(1+\delta'))}, \\ &\leq e^{-n\frac{\delta'^2}{2}}. \end{aligned}$$

The upper bound is obtained similarly.

Lemma 3. *Let $A_1 \sim PPP(\lambda_1)$ and $A_2 \sim PPP(\lambda_2)$ be independent Poisson point processes, where $\lambda_2 < \lambda_1$.*

Assume that A_1 observes n events in a certain time range. Then, A_2 observes at most $(1+\delta')^2 \lambda_2 / \lambda_1 n$ events except with probability $2e^{-\frac{n\lambda_2}{2\lambda_1} \delta'^2}$.

Similarly, assuming that A_2 observes n events, A_1 observes at most $(1+\delta')^2 \lambda_1 / \lambda_2 n$ events except with probability $2e^{-\frac{n}{2} \delta'^2}$.

Proof. From Lemma 2, the time required from process A_1 to observe n events is upper bounded by $n(1+\delta')/\lambda_1$ except with probability $e^{-n\delta'^2/2}$. So, over this range of time, except with probability $e^{-n\frac{\delta'^2}{2}}$, A_2 is stochastically upper bounded by $Y \sim \text{Poisson}\left(n\frac{\lambda_2(1+\delta')}{\lambda_1}\right)$. Now, applying the bound for a Poisson random variable from Lemma 1,

$$\begin{aligned} \mathbb{P}\left(Y \geq \frac{n\lambda_2}{\lambda_1}(1+\delta')^2\right) &\leq \exp\left(-\frac{n\left(\frac{\lambda_2(1+\delta')}{\lambda_1}\right)^2 \delta'^2}{2n\frac{\lambda_2}{\lambda_1}(1+\delta')^2}\right), \\ &= \exp\left(-\frac{n\lambda_2}{2\lambda_1} \delta'^2\right). \end{aligned}$$

Therefore, over the time when A_1 observes n events, $A_2 \leq (1+\delta')^2 \lambda_2 / \lambda_1 n$ except with probability $2e^{-\frac{n\lambda_2}{2\lambda_1} \delta'^2}$.

The other bound is obtained similarly.

Lemma 4. *Let $A_1 \sim PPP(\lambda_1)$ and $A_2 \sim PPP(\lambda_2)$ be independent Poisson point processes, and $A = A_1 + A_2$. Given that over some range of time $A = n$, then $A_1 \leq \frac{n\lambda_1}{\lambda_1+\lambda_2}(1+\delta')$ except with probability $e^{-\frac{n\lambda_1}{3(\lambda_1+\lambda_2)} \frac{\delta'^2(2)}{3}}$.*

Proof. Given A , A_1 behaves like a binomial random variable: $(A_1 | A = n) \sim \text{Bin}\left(n, \frac{\lambda_1}{\lambda_1 + \lambda_2}\right)$. Therefore, the upper bound on A_1 is obtained by using the standard Chernoff bound for binomial random variables.

E Security Proofs

Let's recall the notation. $\mathcal{P}^{(1)}$ is the honest trimmed chain and $\mathcal{P}^{(2)}$ is the adversarial chain. The underlying chains be $\mathcal{C}^{(1)}$ and $\mathcal{C}^{(2)}$ respectively. Their respective trimming points are $B'^{(1)}(t)$ and $B'^{(2)}(t)$, and their respective chain lengths are $B^{(1)}$ and $B^{(2)}$. Let $b = \text{LCA}(\mathcal{P}^{(1)}, \mathcal{P}^{(2)})$. And, $b' = \text{LCA}(\mathcal{C}^{(1)}, \mathcal{C}^{(2)})$. Trivially, $b' \geq b$. Observe that b' and b may not be equal because the trimmed chains may skip blocks and thus have an earlier LCA block.

$L^{(1)}(\cdot)$ is the level range function for $\mathcal{P}^{(1)}$, and $L^{(2)}$ for $\mathcal{P}^{(2)}$. Denote,

$$\begin{aligned} \mu^{(1)} \text{ such that } L^{(1)}(\mu^{(1)} + 1) < b \leq L^{(1)}(\mu^{(1)}), \\ \mu^{(2)} \text{ such that } L^{(2)}(\mu^{(2)} + 1) < b \leq L^{(2)}(\mu^{(2)}). \end{aligned}$$

From the method by which $\text{Weight}()$ function in Algorithm 2 chooses levels, for the level ranges of $\mu^{(1)}$ and $\mu^{(2)}$, the $\text{Weight}()$ function might use levels $\hat{\mu}^{(1)} \leq \mu^{(1)}$ and $\hat{\mu}^{(2)} \leq \mu^{(2)}$ respectively (see line 16 and 17 of Algorithm 2). For subsequent level ranges, the corresponding levels are used by $\text{Weight}()$ (see line 20).

First, we bound the gap between the LCA of the pruned chains, b , and the LCA of the underlying chains, b' .

Lemma 5. *Denote,*

$$M = \max \left\{ 2^{\hat{\mu}^{(1)}} g\left(\mathcal{P}^{(1)}, \mu^{(1)}\right), \quad 2^{\mu^{(1)}-1} g\left(\mathcal{P}^{(1)}, \mu^{(1)} - 1\right) \right\}.$$

M essentially indicates the maximum of PoW of the first and second level ranges of the honest trimmed chain.

Let b' be in the $\mu'^{(2)}$ -level range in $\mathcal{P}^{(2)}$. That is,

$$L^{(2)}(\mu'^{(2)} + 1) < b' \leq L^{(2)}(\mu'^{(2)}).$$

If $\mu'^{(2)} = \mu^{(2)}$, then,

$$2^{\hat{\mu}^{(2)}} |\mathcal{P}^{(2)}\{b + 1 : b' + 1\} \uparrow^{\hat{\mu}^{(2)}}| \leq M.$$

Otherwise,

$$\begin{aligned} & 2^{\hat{\mu}^{(2)}} |\mathcal{P}^{(2)}\{b + 1 : L(\mu^{(2)}) + 1\} \uparrow^{\hat{\mu}^{(2)}}| \\ & + \sum_{\mu^{(2)} > \mu > \mu'^{(2)}} 2^\mu W^{(2)}(\mathcal{P}^{(2)}, \mu) \\ & + 2^{\mu'^{(2)}} |\mathcal{P}\{L(\mu'^{(2)}) + 1 : b' + 1\} \uparrow^{\mu'^{(2)}}| \leq M. \end{aligned}$$

The above two equations essentially convey that the weight of the adversary's trimmed chain $\mathcal{P}^{(2)}$ between b and b' is upper bounded by M .

Proof (Proof of Lemma 5). This is a direct consequence of the dominant-superchain property.

Next, we upper bound the weight of a trimmed chain.

Lemma 6. *Let \mathcal{C} be a blockchain and \mathcal{P} be its corresponding trimmed chain produced by Algorithm 1. Let $\delta \leq 1/\sqrt{2}$. If $|\mathcal{C}| = B$, then except with probability $\frac{\text{negl}(k)}{B^{a\delta^2/2-1}}$, $S(\mathcal{P}, 0) + |\mathcal{P}\{B' : \}| \leq ((1 + \delta)^2 + \delta)B$.*

Proof. Let μ be the highest level with $S(\mathcal{P}, \mu) \geq \delta B$. This means that, for all $\mu' \leq \mu$,

$$|\mathcal{P}\{L_f(\mu') : L_l(\mu') + 1\} \uparrow^{\mu'}| \geq k + a \log(\delta B).$$

By Lemma 3, we get that for every $\mu' \leq \mu$,

$$|\mathcal{P}\{L_f(\mu') : L_l(\mu') + 1\} \uparrow^{\mu'}| \leq (1 + \delta^2) (L_l(\mu') - L_f(\mu') + 1),$$

except with probability

$$2e^{-\frac{k+a \log(\delta B)}{2} \delta^2} = \frac{\text{negl}(k)}{B^{a\delta^2/2}}.$$

From Lemma 9, the number of level-ranges in \mathcal{P} is at most $\log B / \log 2$, except with probability $\text{negl}(k)/B^{a/4}$. Therefore, by a union bound,

$$S(\mathcal{P}, 0) + |\mathcal{P}\{B' : \}| \leq ((1 + \delta)^2 + \delta)B,$$

except with probability,

$$\log B \cdot \frac{\text{negl}(k)}{B^{a\delta^2/2}} + \frac{\text{negl}(k)}{B^{a/4}} = \frac{\text{negl}(k)}{B^{a\delta^2/2-1}}.$$

Next, we prove Theorem 1.

E.1 Proof of Theorem 1

Proof (Proof of Theorem 1). The proof is by induction on the length of the honest blockchain $B^{(1)}$.

As the base case, consider $B^{(1)} = ck$. Since no trimming has occurred up to this point, the probability of a trim-attack up to this point is 0.

As the induction hypothesis, assume that up to chain-length B , the adversary wasn't able to create a fork from beyond a honest node's trimming point. That is,

$$\begin{aligned} \forall t \text{ such that } B_t^{(1)} < B, \text{ Compare } (\mathcal{P}_t^{(1)}, \mathcal{P}_t^{(2)}) = \mathcal{P}_t^{(2)} \text{ implies that} \\ \text{LCA}(\mathcal{P}_t^{(1)}, \mathcal{P}_t^{(2)}) \geq B_t^{(1)} - \Delta(S(\mathcal{P}_t^{(1)}, 0) + |\mathcal{P}\{B_t^{(1)} : \}|). \end{aligned} \quad (7)$$

For the induction step, consider a time t when the length of the honest chain is $B_t^{(1)} = B$. As a contradiction, assume that the adversary has created a secret chain $\mathcal{P}_t^{(2)}$ such that, $\text{Compare}(\mathcal{P}^{(1)}, \mathcal{P}^{(2)}) = \mathcal{P}^{(2)}$, and $b = \text{LCA}(\mathcal{P}^{(1)}, \mathcal{P}^{(2)}) < B - \Delta(S(\mathcal{P}_t^{(1)}, 0) + |\mathcal{P}\{B_t^{(1)} : \}|)$.

We now split into two cases based on the location of b :

Case 1: $b \geq \delta B$ By superchain quality of $\mathcal{P}^{(1)}$, we have,

$$\begin{aligned} S(\mathcal{P}^{(1)}, \mu^{(1)}) &\geq (1 - \delta)b, \\ &\geq (1 - \delta)\delta B. \end{aligned} \quad (8)$$

Moreover, since $\mathcal{P}^{(2)}$ shares blocks with $\mathcal{P}^{(1)}$ up to block b , we have,

$$\begin{aligned} S(\mathcal{P}^{(2)}, \mu^{(2)}) &\geq (1 - \delta)b, \\ &\geq (1 - \delta)\delta B. \end{aligned} \quad (9)$$

As a consequence of Lemma 5, the weight of $\mathcal{P}^{(2)}$ between blocks b and b' is upper bounded by M .

Next, we define a block $b'' \geq b'$ which is such that all level ranges in $\mathcal{P}^{(2)}\{b'' : \}$ are of size at least $c'g(\mathcal{P}^{(1)}, \mu^{(1)})$, for some $1 < c' < c$. In order to ensure this, b'' can be defined as,

$$b'' = \begin{cases} b', & \text{if } |\mathcal{P}^{(2)}\{b' + 1 : L_l(\mu'^{(2)}) + 1\} \uparrow^{\mu'^{(2)}}| \\ & \geq c'(k + a \log((1 - \delta)\delta B)), \\ L_l^{(2)}(\mu'^{(2)}) + 1, & \text{otherwise,} \end{cases} \quad (10)$$

where $\mu'^{(2)}$ is the level range in which block b' resides in $\mathcal{P}^{(2)}$. If $b' > b$, then $\mu'^{(2)} \leq \max\{\hat{\mu}^{(1)}, \mu^{(1)} - 1\}$. Therefore, the weight of $\mathcal{P}^{(2)}$ between b' and b'' is upper bounded by $c'M$.

Let b^* be the earliest block such that all blocks in $\mathcal{P}^{(2)}\{b^* + 1 : \}$ were mined by the adversary. Then, by the induction hypothesis,

$$b^* - b'' \leq \Delta(S(\mathcal{P}_t^{(1)}, 0) + |\mathcal{P}_t^{(1)}\{B_t^{(1)} : \}|).$$

The above statement is proved as follows. Assume there was a time t^* when b^* was in a honest node's chain, but at a later time t the block after b'' in $\mathcal{P}_t^{(2)}$ is not in the honest node's chain \mathcal{C}_t . By the induction hypothesis,

$$\begin{aligned} b^* - b'' &\leq \Delta(S(\mathcal{P}_{t^*}^{(1)}, 0) + |\mathcal{P}_{t^*}^{(1)}\{B_t^{(1)} : \}|), \\ &\leq \Delta(S(\mathcal{P}_t^{(1)}, 0) + |\mathcal{P}_t^{(1)}\{B_t^{(1)} : \}|). \end{aligned}$$

Given, $k' = k - a \log(((1 + \delta^2) + \delta)/\delta)$. Therefore, from Lemma 6, except with probability $\frac{\text{negl}(k)}{B^{a\delta^2/2-1}}$,

$$\Delta(S(\mathcal{P}_t^{(1)}, 0) + |\mathcal{P}_t^{(1)}\{B_t^{(1)} : \}|) \leq g(\mathcal{P}^{(1)}, \mu^{(1)}).$$

Therefore,

$$\text{Weight}(\mathcal{P}^{(2)}, b^*) \geq \frac{c' - 1}{c'} \frac{c - 1 - c'}{c} \text{Weight}(\mathcal{P}^{(1)}, b).$$

Since, each level range in $\mathcal{P}^{(2)}\{b^* : \}$ contains at least $g(\mathcal{P}^{(1)}, \mu^{(1)})$ number of corresponding superblocks, we have that except with probability $\text{negl}(k) \log B/B^{a\delta^2/2}$, the time required T required by the adversary is lower bounded as,

$$T \geq \frac{(1 - \delta)}{\lambda_a} \frac{c' - 1}{c'} \frac{c - 1 - c'}{c} \text{Weight}(\mathcal{P}^{(1)}, b).$$

In this time, except with probability $\text{negl}(k)/B^{a\delta^2/2}$, the honest chain growth is lower bounded as,

$$|\mathcal{C}\{b : \}| \geq (1 - \delta)^2 \frac{\lambda_h}{\lambda_a} \frac{c' - 1}{c'} \frac{c - 1 - c'}{c} \text{Weight}(\mathcal{P}^{(1)}, b).$$

We know from superquality property that $\text{Weight}(\mathcal{P}^{(1)}, b) \geq (1 - \delta)|\mathcal{C}\{b : \}|$. Therefore, we reach a contradiction if,

$$(1 - \delta)^3 \frac{\lambda_h}{\lambda_a} \frac{c' - 1}{c'} \frac{c - 1 - c'}{c} > 1. \quad (11)$$

Case 2: $b < \delta B$ The reason to consider this case separately is because when $b < \delta B$, the corresponding level-range containing b may not have enough number of blocks for us to obtain a strong concentration bound. Therefore, we jump ahead to a level which contains enough superblocks. Let $\tilde{\mu}^{(2)}$ be the highest-level in $\mathcal{P}^{(2)}$ such that,

$$S(\mathcal{P}^{(2)}, \tilde{\mu}^{(2)}) \geq \delta B.$$

In case b' falls in the level range of $\tilde{\mu}^{(2)}$ or below, then the same analysis for Case 1 holds. In particular, if the relation in (11) is satisfied, we reach a contradiction except with probability $\frac{\text{negl}(k)}{B^{a\delta^2/2-1}}$.

Otherwise, if b' falls in a level-range higher than $\tilde{\mu}^{(2)}$, we set,

$$b'' = L_f^{(2)}(\tilde{\mu}^{(2)}).$$

Since $b < \delta B$, we have,

$$\begin{aligned} \text{Weight}(\mathcal{P}^{(2)}, b'') &\geq (1 - \delta) \text{Weight}(\mathcal{P}^{(2)}, b), \\ &\stackrel{(x)}{\geq} (1 - \delta) \text{Weight}(\mathcal{P}^{(1)}, b), \\ &\stackrel{(y)}{\geq} (1 - \delta)^3 B. \end{aligned}$$

(x) follows because we assume that $\mathcal{P}^{(2)}$ beats $\mathcal{P}^{(1)}$. (y) follows by superchain quality of the honest chain and the fact that $b < \delta B$.

By the same argument of the induction hypothesis used in Case 1, except with probability $\frac{\text{negl}(k)}{B^{a\delta^2/2-1}}$, $(c-1)/c$ fraction of the blocks in $\mathcal{P}^{(2)}\{b'' : \}$ were mined by the adversary. Again, except with probability $\frac{\text{negl}(k)}{B^{a\delta^2/2}}$, by the time that the adversary mines those blocks, the length of the honest chain would be,

$$|\mathcal{C}\{b+1 : \}| \geq (1-\delta)^5 \frac{\lambda_h}{\lambda_a} \frac{c-1}{c} B.$$

The above equation is a contradiction if,

$$(1-\delta)^5 \frac{\lambda_h}{\lambda_a} \frac{c-1}{c} > 1. \quad (12)$$

_____ End Case 2 _____

There are at most $B - \Delta(\mathcal{P}^{(1)})$ unique choices that can be made for block b . For each choice, the conclusions of Case 1 and Case 2 hold except with probability $\frac{\text{negl}(k)}{B^{a\delta^2/2-1}}$. Therefore, taking a union bound over all choices of b , the conclusions hold except with probability $\frac{\text{negl}(k)}{B^{a\delta^2/2-2}}$. This completes the induction step.

The induction step at chain-length B is violated with probability $\frac{\text{negl}(k)}{B^{a\delta^2/2-2}}$. Therefore, summing over all chain lengths, we have,

$$\begin{aligned} \mathbb{P}(\text{trim} - \text{attacked}) &\leq \sum_{B=1}^{\infty} \frac{\text{negl}(k)}{B^{a\delta^2/2-2}}, \\ &= \text{negl}(k), \end{aligned} \quad (13)$$

where the last equation follows since $a\delta^2/2 - 2 \geq 2$.

E.2 Proofs of Other Theorems:

Proof (Proof of Corollary 1). We have proved in Theorem 1 that the adversary cannot create a longer fork from before a honest node's trimming point except with probability $\text{negl}(k)$. Since a honest node retains all blocks after the trimming point, comparing two trimmed chains that differ only after the trimming point is equivalent to comparing the underlying chains (see line 11 and 12 in Algorithm 2).

Proof (Proof Sketch of Theorem 2). In order to pass through the **state – verify** protocol in Algorithm 3, the adversary has to produce a malicious state at the trimming point of an honest node, such that it verifies against all the states corresponding to the untrimmed chain tail. In Section B.3, we explained that the adversary's process of generating a malicious state sequence is a Poisson process of rate λ_s , where $\lambda_s \ll \lambda_a$. Therefore, from the same argument as in Theorem 1, we can prove that there exists no time when the adversary can create such a malicious state.

Proof (Proof of Corollary 2). This is a corollary of Theorems 1 and 2. Since, the probability that there exists a time when the system is either **trim – attacked** or **state – attacked** is very small, the probability that an arriving node sees the system attacked in either way is negligible in k .

Proof (Proof Sketch of Theorem 3). Assume that all relevant quantities concentrate strongly around their means. First, the highest level-range would be $\mu_h = O(\log B)$. This is because, a level μ_h is 2^{μ_h} times harder to find than a level-0 superblock.

Next, a level range of level- μ would contain at most $O(\log B)$ μ -superblocks. Because, if there were more, then it could be trimmed to level- $(\mu + 1)$. Since we also retain suffixes of length $O(\log B)$ blocks of every lower level, level-range μ would contain about $O(\log^2 B)$ blocks. Adding across all the μ_h number of level ranges, the trimmed portion of \mathcal{P} would contain about $O(\log^3 B)$ blocks. And, the untrimmed tail would contain at most $O(\log B)$ blocks.

Since an interlink contains a link to the previous superblock of every level, the size of the interlink would be $O(\log B)$, thus making the size of each block $O(\log B)$. Therefore, the total storage requirement would be $O(\log^4 B)$. These statements are proved more rigorously in appendix H.

F Trimming Attack

Here we prove Theorem 4.

Proof (Proof of Theorem 4). Let $f(x) = c_0 \log(1 + x)$, for some small constant c_0 . Define a sequence $(\alpha_i)_i$ as follows:

$$\begin{aligned} \alpha_0 &= 0, \\ \alpha_1 &= 1, \\ \alpha_i &= 2f\left(\sum_{j=1}^{i-1} \alpha_j\right), \quad i \geq 2. \end{aligned}$$

The adversarial attack follows a sequence of attempts. The i^{th} attempt begins when $|\mathcal{C}_t| = \sum_{j=0}^{i-1} \alpha_j$. At this point, the adversary abandons the previous attempt and starts mining on a secret chain \mathcal{D}_t forking away from \mathcal{C}_t at this tip. The attempt lasts until one of \mathcal{C}_t or \mathcal{D}_t reach a length of $\left(\sum_{j=0}^i \alpha_j\right) + 1$. If \mathcal{D} reaches the length first, then the adversary waits until \mathcal{C} reaches a length of $\sum_{j=0}^i \alpha_j$, at which point it publishes \mathcal{D} . Since, \mathcal{D} will be longer than \mathcal{C} , the honest nodes will adapt \mathcal{D} . The fact that this is a successful trimming attack follows from the following claim:

Claim 5 *For all large enough i , $\alpha_i \geq f(\sum_{j=1}^i \alpha_j)$*

From the definition of the attack, the length of the honest chain when adversary publishes his secret chain is $\sum_{j=1}^i \alpha_j$. Recall, $f(\sum_{j=1}^i \alpha_j)$ is an upperbound on the number of blocks between the trimming point and the tip of the chain for the honest node, since $B - B' = o(\log B)$. Therefore, the above claim shows that the adversary has successfully created a fork from before a trimming point of any node. To prove the claim, for large enough i we have,

$$\begin{aligned} \alpha_i &= f\left(\sum_{j=1}^{i-1} \alpha_j\right) + f\left(\sum_{j=1}^{i-1} \alpha_j\right), \\ &\stackrel{(a)}{\geq} f\left(\sum_{j=1}^{i-1} \alpha_j\right) + f(\alpha_i), \\ &\stackrel{(b)}{\geq} f\left(\sum_{j=1}^i \alpha_j\right). \end{aligned}$$

(a) follows because $2c_0 \log(z) \leq z$. (b) follows by the sub-additivity of $\log(1+z)$.

The below two claims are needed to compute the probability of a successful attack attempt.

Claim 6 $\alpha_i \leq c_1 \log i$, for all $i \geq 2$ and $c_1 = 1 + 4c_0$.

Assuming $c_0 \leq 1/4$, $1 \leq c_1 \leq 2$. We will prove the above claim by induction. As the base case, $\alpha_2 = 2c_0 \log(1 + \alpha_1) = 2c_0 \log 2 \leq c_1 \log 2$. Assume that the claim is true for all $j \leq i-1$. Then,

$$\begin{aligned} \alpha_i &= 2c_0 \log\left(1 + \sum_{j=1}^{i-1} \alpha_j\right), \\ &\leq 2c_0 \log(1 + (i-1)c_1 \log i), \\ &\stackrel{(a)}{\leq} 2c_0 \log(ic_1 \log i), \\ &= 2c_0 \log i + 2c_0 \log c_1 + 2c_0 \log \log i, \\ &\stackrel{(b)}{\leq} 2c_0 \log i + 2c_0 \log i + \log i, \\ &= (4c_0 + 1) \log i. \end{aligned}$$

(a) follows since $c_1 \geq 1$. (b) follows since $i \geq 2 \geq c_1$.

The probability of the adversary's i^{th} attempt succeeding is $e^{-c_2 \alpha_i}$, for some constant c_2 [37].

Claim 7 $\sum_{i=1}^{\infty} e^{-c_2 \alpha_i} = \infty$.

To prove the claim,

$$\begin{aligned}
\sum_{i=2}^{\infty} e^{-c_2 \alpha_i} &= \sum_{i=2}^{\infty} e^{-2c_0 c_2 \log(1 + \sum_{j=1}^{i-1} \alpha_j)}, \\
&\stackrel{(a)}{\geq} \sum_{i=2}^{\infty} e^{-\log(1 + \sum_{j=1}^{i-1} \alpha_j)}, \\
&= \sum_{i=2}^{\infty} \frac{1}{1 + \sum_{j=1}^{i-1} \alpha_j}, \\
&\stackrel{(b)}{\geq} \sum_{i=2}^{\infty} \frac{1}{1 + \sum_{j=1}^{i-1} c_1 \log j}, \\
&\stackrel{(c)}{=} \sum_{i=2}^{\infty} \frac{1}{1 + c_1 i \log i}, \\
&= \infty.
\end{aligned}$$

- (a) follows since c_0 can be made as small as desired. (b) follows from Claim 6.
(c) can be verified to diverge using the integral test.

Since the attack-attempts are carried over disjoint sections of the blockchain, all the attempts are mutually independent. Therefore, from Claim 7 and the second Borel-Cantelli Lemma, we have,

$$\mathbb{P}(\text{trim} - \text{attacked}) = 1.$$

G Dominant Superchain

We will use the following concentration bound for sums of independent, but not necessarily identically distributed, Bernoulli random variables. It can be proved by an elementary application of Chernoff bound and can be found in [14].

Lemma 7. *Let $\{X_i\}_{i=1}^n$ be a sequence of independent random variables with $X_i \sim \text{Bernoulli}(p_i)$. Let $S_n = \sum_{i=1}^n X_i$, and $s_n = \sum_{i=1}^n p_i$. Then, for $0 < \varepsilon < 1$,*

$$\mathbb{P}(S_n \leq (1 - \varepsilon)s_n) \leq e^{-\frac{\varepsilon^2}{2}s_n}.$$

We now prove the high likelihood of dominant superchains.

Lemma 8. *Let \mathcal{C} be a blockchain and $\mathcal{C} \uparrow^\mu$ be its corresponding level- μ upchain. Then, for any $k \in \mathbb{N}$ and $g \geq k + a \log(|\mathcal{C}|)$. Then, if $a \geq 8$, $\mathcal{C} \uparrow^\mu$ is a dominant superchain with high probability in k and $|\mathcal{C}|$.*

Proof. Consider a starting block $b_1 \geq \mathcal{C}[0]$ and ending block $b_2 \leq \mathcal{C}[-1]$.

First, observe that if $\mathcal{C}[b_1 : b_2 + 1] \uparrow^\mu \geq 1$, then any trimmed chain \mathcal{P} starting at b_1 and ending at b_2 , and with its highest level range being at most μ , will

contain at least one level- μ superblock. This is because a level- μ superblock, call it b_3 , is also a level- μ' superblock for any $\mu' \leq \mu$. Therefore, the interlink data-structure ensures that there is a level range μ' , which contains a link to b_3 .

Let \mathcal{P}' be the trimmed chain of the highest weight starting at b_1 and ending at b_2 , and with its highest level-range being at most μ . Assume that it satisfies the premise as defined in equation 3. That is,

$$2^{\mu_1} |\mathcal{P}'\{l : L'_l(\mu') + 1\} \uparrow^{\mu_1}| + \sum_{\mu'' < \mu'} 2^{\mu''} |\mathcal{P}'\{L'_f(\mu'') : L'_l(\mu'') + 1\} \uparrow^{\mu''}| \geq 2^\mu g.$$

It is useful to re-arrange the equation by dividing both sides by 2^μ .

$$2^{\mu_1 - \mu} |\mathcal{P}'\{l : L'_l(\mu') + 1\} \uparrow^{\mu_1}| + \sum_{\mu'' < \mu'} 2^{\mu'' - \mu} |\mathcal{P}'\{L'_f(\mu'') : L'_l(\mu'') + 1\} \uparrow^{\mu''}| \geq g. \quad (14)$$

Given that a block is a level- μ'' superblock, let X be an indicator variable for it also to be a level- μ superblock. Then, $\mathbb{P}(X = 1) = 2^{\mu'' - \mu}$. Therefore, we can associate a sequence of Bernoulli random variables $\{X_i\}_{i=1}^n$ with \mathcal{P} , where X_i indicates whether the i^{th} superblock as taken in (14) is a level- μ superblock. Denoting $\mathbb{P}(X_i = 1) = p_i$, equation (14) says that,

$$s_n = \sum_{i=1}^n p_i \geq g.$$

Now, using Lemma 7, we can lower bound $S_n = \sum_{i=1}^n X_i$, as,

$$\begin{aligned} \mathbb{P}(S_n \leq 1) &\leq e^{-\frac{(1-1/g)^2}{2}g}, \\ &\leq e^{-g/4}, \\ &\leq \frac{\text{negl}(k)}{B^{a/4}}, \end{aligned}$$

where we denote $B = |C|$. There are totally B ways to choose b_1 , and then $B - b_1$ ways to choose b_2 . Therefore, there are totally at most B^2 ways to choose b_1 and b_2 . Taking a union bound over all possible starting points b_1 and ending points b_2 , we conclude there exists at least one level- μ superblock in \mathcal{C} with high probability if $a \geq 8$.

H Succinctness

Lemma 9. *When the length of the blockchain \mathcal{C} is B , the highest-level of superblock in the corresponding trimmed chain \mathcal{P} is at most $\log B / \log 2$ except with probability $\text{negl}(k) / B^{a/4}$.*

Proof. Let $\mu = \log B / \log 2$. Then, weight of 1 level- μ superblock is $2^\mu = B$. Therefore, $g(\mathcal{P}, \mu) \geq B$.

Also, $|\mathcal{C} \uparrow^\mu| \sim \text{Poisson}(1)$. Therefore, applying the Poisson concentration bound,

$$\begin{aligned} \mathbb{P}(|\mathcal{C} \uparrow^\mu| \geq g(\mathcal{P}, \mu)) &\leq \mathbb{P}(|\mathcal{C} \uparrow^\mu| \geq 1 + (k-1) + a \log B), \\ &\leq e^{-\frac{k-1+a \log B}{4}}, \\ &= \frac{\text{negl}(k)}{B^{a/4}}. \end{aligned}$$

The proof is complete by observing that if $|\mathcal{C} \uparrow^\mu| < g(\mathcal{P}, \mu)$, then the chain won't be trimmed to this level.

Next, we show that individual level ranges aren't too big.

Lemma 10. *Let \mathcal{P} be a trimmed chain of an underlying chain \mathcal{C} obtained by the trimming protocol. Let L_f and L_l be the associated level range functions. Then, for every level μ , $|\mathcal{P}\{L_f(\mu) + 1 : L_l(\mu)\} \uparrow^\mu| \leq 4f(\mathcal{P}, \mu)$, with high probability.*

Proof. As a contradiction, assume that there were a level μ , such that $|\mathcal{P}\{L_f(\mu) + 1 : L_l(\mu)\} \uparrow^\mu| \leq 4f(\mathcal{P}, \mu)$. Then, from Lemma 3, with high probability, we would have $|\mathcal{P}\{L_f(\mu) + 1 : L_l(\mu)\} \uparrow^{\mu+1}| \leq f(\mathcal{P}, \mu)$. This means that the trimming protocol would have trimmed this range to level $\mu + 1$.

Now we are ready to prove the optimistic succinctness theorem.

Proof (Proof Theorem 3). From Lemma 9 we know that the highest level is $O(\log B)$ with high probability. And, from Lemma 10, the number of μ -level superblocks in level range μ is $O(\log B)$ with high probability.

According to the construction of level range μ in the trimming protocol, it contains $O(\log B)$ superblock of level- μ' for each level $\mu' < \mu$. Therefore, each level range contains $O(\log^2 B)$ blocks with high probability. And, since there are at most $O(\log B)$ level ranges, the number of blocks in the trimmed section of \mathcal{P} is $O(\log^3 B)$ blocks. Since, the untrimmed tail consists of $O(\log B)$ blocks, \mathcal{P} totally contains $O(\log^3 B)$ blocks.

Each block has the interlink, which in turn contains a link to the previous superblock of every level. Since there are at most $O(\log B)$ level of superblocks, the size of the interlink is $O(\log B)$. This makes the overall size of \mathcal{P} as $O(\log^4 B)$.

I Our Protocol in the Random Oracle Model

In our analysis, we have modelled PoW mining as calls to an ideal random functionality which returns a “success” with a small probability, and “failure” otherwise, independent of everything else. However, in current PoW systems, mining is performed by making calls to a random oracle hash function, H :

$\{0,1\}^* \rightarrow \{0,1\}^\kappa$, where κ is a security parameter. Since the output space of H is finite, we cannot have our formulation of **trim – attacked** in Definition 1 as a security model. This is because **trim – attacked** considers time from 0 to infinity, during which time the hash function H is going to experience collisions with probability 1. In particular, in order to ensure that H doesn’t experience collisions with high probability, one can only run the blockchain for $\text{poly}(\kappa)$ time (or, $\text{poly}(\kappa)$ rounds in the discrete time model)⁶.

In [30], the authors denote our ideal random functionality as \mathcal{F}_{tree} . They prove that when a blockchain is run for $\text{poly}(\kappa)$ rounds, the \mathcal{F}_{tree} model and the random oracle model are statistically close (see Lemma 5.1 in [30]). Using this, they prove that if a blockchain protocol is secure under the \mathcal{F}_{tree} model, it is secure under the random oracle model for $\text{poly}(\kappa)$ number of rounds as well. We can employ this result in transferring our security definitions in Section 5 to the more realistic random oracle model for hashing. For example, Theorem 1 would imply that an adversary without a majority of the mining power cannot create a longer-fork from before a honest node’s trimming point at any point during the $\text{poly}(\kappa)$ rounds of blockchain execution (instead of infinite number of rounds). Similar arguments hold under the Bitcoin Backbone model[23] as well.

J Optimizing Storage with Stateless Blockchains

Our trimming protocol optimizes the amount of cold storage required to represent the PoW in the blockchain. In this section, we describe how our work can be interfaced to other methods that optimize the storage of the *state*. In hybrid nodes, states need to be stored both in cold and hot storage. The state at the trimming point, $\text{state}(B'_t)$, needs to be stored in cold-storage in order to recover the state at the chain tip when forks are created. Conversely, the state at the tip of the chain, $\text{state}(B_t)$, needs to be stored in hot-storage in order to perform transaction and block validation. Along with the length of the blockchain, the size of the blockchain-state is also rapidly increasing with time. For instance, at the time of writing Bitcoin’s UTXO state is almost 4GB in size, which motivates lessening high storage and verification time burdens by extending our work to optimize both the state’s cold-storage at the trimming point, and its hot-storage at the chain tip.

So far, we have described that storing state-commitments in the blocks enables one to securely establish the state of a trimmed chain, by use of the **state-verify** protocol described in Algorithm 3. In a different line of work, called *stateless blockchains*, state-commitments are used to reduce the amount of hot-storage required for transaction validation, thus speeding up validation. In this section, we illustrate that when hybrid nodes are used in stateless-blockchains, then state commitments can serve a dual purpose: 1) they can be used to establish the state of the trimmed chain; 2) and, they can be used to perform stateless transaction validation.

⁶ $\text{poly}(\kappa)$ means polynomial in κ .

Stateless validation, first proposed by Todd [38], is a scheme where nodes validating transactions store a short cryptographic state-commitment rather than the entire state. A client then provides a membership proof that the node can verify against the commitment. Recently, several constructions have been developed to perform stateless validation in both the UTXO and account-based state models using various cryptographic primitives [10, 13, 4, 39, 34, 1, 19]. As an example, the following is how stateless validation of an account-based state would work via *vector commitments*, a primitive allowing one to commit to an ordered sequence of values in a way that allows one to later open the commitment at a specific position in the sequence, with proof that the i^{th} value is the i^{th} committed message (see [8]). Suppose an account is represented by a tuple of the form mapping a public key to a balance: (pk_i, v_i) , and that party 1 want to transfer an amount m to party 2. Then, the transaction submitted by party 1 takes the form $tx = (pk_1, v_1, pk_2, m, \pi, \sigma)$ where pk_1 is the public key of the source account, pk_2 is the public key of the destination account, v_1 is the initial balance of pk_1 , m is the amount being transferred, π is a proof of membership of (pk_1, v_1) in the commitment, and σ is a signature of (v_1, pk_2, m, π) under pk_1 . In order to validate a transaction, a node checks that $m < v_1$, the validity of the signature, and the validity of the proof with respect to the public commitment $y(B_t)$. If the checks passes, the commitment is updated to reflect the change in the two accounts: $(pk_1, -m)$ and $(pk_2, +m)$. With each such update, users update their proofs π so that they remain valid with respect to the latest commitment.

As a first step, observe that stateless blockchains optimize hot-storage by obviating the need to store the state at the tip of the chain in the RAM. At this point, the state at the trimming point is still stored in the cold storage. We can use this to provide an interface from trimming to stateless blockchains: Firstly, if the proof for some client becomes outdated past the trimming point because it was offline for a long while, then the client can recompute its latest proof by querying a hybrid node for $\text{state}(B'_t)$ and the untrimmed chain tail. Secondly, if a fork in the chain makes the proofs of several clients invalid, they can all recompute their proofs by contacting the hybrid node similarly.

Now, we describe a way to optimize the cold-storage of the state at the trimming point as well. In order to do this, we need to make an assumption on the clients. At every point, the client needs to store its membership proof corresponding to every block in the untrimmed tail of hybrid nodes. This enforces a couple of constraints on clients. First, the storage requirements of a client would scale logarithmically in the length of the blockchain as well, since the length of the chain-tail grows logarithmically. Second, the clients need to be online very often so that their proofs never get outdated. If this is too cumbersome, clients can alternatively delegate the job of saving and updating proofs to proof-serving nodes [10]. In this case, the hybrid node need not store the entire state at the trimming point either. This is because, even in the event that a fork is created in the untrimmed-tail, all the clients would have their proofs corresponding to the forking point. In this case, a joining hybrid node would no longer need to

use the **state-verify** protocol, since just the state commitment suffices. The rest of the hybrid node's protocols proceed as before.