Mining in Logarithmic Space with Variable Difficulty

Loïc Miller, Dorian Pacaud, Nathanael Derousseaux, Emmanuelle Anceaume, Romaric Ludinard

Mining in Logarithmic Space with Variable Difficulty

Loïc Miller[1], Dorian Pacaud[1], Nathanël Derousseaux–Lebert[1], Emmanuelle Anceaume[2], and Romaric Ludinard[1]

[1] IMT Atlantique, IRISA
[2] CNRS, IRISA

Abstract. This paper addresses both the storage and communication complexity issues of the blockchain. Techniques exist to optimize application data, whose size may grow or shrink over time. On the other hand, consensus data is vital to guarantee the trust we have in the blockchain, and is kept in its entirety. The Non-Interactive Proof of Proof-of-Work primitive addresses this scalability issue assuming a fixed computational power. We present for the first time a construction that provably satisfies the requirements of a non-interactive proof of proof-of-work that is secure against a 1/3 adversary in a dynamic environment. Succinctness, security and onlineness of the scheme are proven, while experimental results confirm the exponential reduction of the Bitcoin blockchain.

Keywords: blockchain · Proof-of-Work · logspace mining · variable difficulty

1 Introduction

Blockchain immutability, i.e. its ability to remain a permanent, indelible, and unalterable history of transactions, is a feature that brings an unprecedented level of trust to the stored data. In Proof-of-Work (PoW) based blockchains [1], immutability results from *i)* cryptography, i.e. transactions are cryptographically sealed in blocks, which are themselves cryptographically linked together in a sequence starting from the unique genesis block, *ii)* block creation difficulty, i.e. as blocks go deeper in the sequence it becomes infeasible to replace them by another sequence of blocks, *iii)* distribution, i.e. each party maintains a copy of the adopted sequence of blocks, and *iv)* a deterministic rule that allows parties to decide which among several sequence of blocks is the blockchain. Cryptographic fingerprints, PoW nonce, and block mining difficulty are sealed in each block header, and because of their essential role in contributing to the construction of one unique immutable sequence of blocks, they are called consensus data. Most existing protocols require consensus data to be locally stored and sent over the network to allow newcomers to synchronize with the system. Consensus data grows linearly over time, hindering the long-term feasibility and thus adoption of blockchain technology. Note that application data (e.g, accounts or UTXOs, smart contract state evolution) are application dependent and belong to the core

of each block. Application data may grow or shrink as time progresses, and so multiple techniques exist and have been well deployed in practice to securely optimize it, e.g., SNAP [2], Layer 2 constructions [3, 4], side chains [5, 6, 7], or compression of multiple transactions into smaller ones [8]. These methods are significant to compress application data, but inapplicable to consensus data. Non-interactive Proofs of Proof-of-Work (NIPoPoWs) have been proposed by Kiayias *et al.* [9] to address the scalability issue of consensus data. A NIPoPoW is designed as a compact proof of the total amount of work that has gone into building a PoW-based blockchain, and should convince in one interaction that it represents a secure and succinct summary of an honest underlying blockchain. Its construction relies on the properties of hash distribution, and consists in sampling a polylogarithmic number of well chosen blocks in the total number of blocks of the original blockchain to estimate its size [9, 10, 11]. However, the security of their construction [9] is proven assuming that all the blocks of the original blockchain have been mined with the same mining difficulty. This is a strong assumption that does not match permissionless blockchains.

Envisioning NIPoPoWs compliant with variable mining difficulty raises numerous challenges. Firstly, this calls for quantifying the extent to which a block is worth sampling (i.e. defining its sampling level): two blocks with very close hash values may not be equally representative due to their respective mining difficulty, and inversely, two blocks showing completely different hash values may have the same sampling level. Secondly it requires to securely dimension the number K of blocks per sampling level to guarantee that, with high probability, the adversary cannot show more difficulty in sampled blocks than what the honest parties can do at any sampling level. Thirdly, this demands to be resilient to *low difficulty attacks*, an attack in which the adversary exploits the verifiers' inability to check the veracity of "old" blocks' mining difficulty. An adversarial strategy is to secretly mine a blockchain whose "old" blocks are computationally easy blocks, and for more recent blocks to match their mining difficulty with the one of the honest blocks. Fourthly, this requires to be able to compare blockchains whose sizes are extremely different. We are not aware of any NIPoPoW construction that proposes to face those challenges. This is our objective.

Our contributions. We present a NIPoPoW construction that:

- Provably handles proof-of-work blockchains whose mining difficulty varies with the system computational power;
- Provably compresses proof-of-work blockchains: $O(\text{poly}\log(b))$ blocks of information (b is the number of blocks of the blockchain) are sufficient to mine new blocks and to synchronize all the nodes of the system;
- Provably achieves security against a Byzantine adversary controlling strictly less than 1/3 of the computational power;
- Provably dimension our security parameter K;
- Experimentally shows that when $|\mathcal{C}| \to \infty$ (where $|\mathcal{C}|$ is equal to the number of blocks of $\mathcal{C}$), we have $2K\log(|\mathcal{C}|) \leq NIPoPoW(|\mathcal{C}|) \leq 3K\log(|\mathcal{C}|)$, where $NIPoPoW(|\mathcal{C}|)$ represents the number of blocks of the NIPoPoW.

In the remainder of the paper, Section 2 details the related work with a particular emphasis on Kiayias *et al.*'s NIPoPoW scheme. Section 3 presents the model of the system, and Section 4 formally specifies the addressed problem. Section 5 proposes our instantiation of NIPoPoWs. Section 6 provides proofs for security, succinctness, onlineness, and the dimensioning of the security parameters. We conclude in Section 7.

2 Related Work

2.1 Light clients

The problem of blockchain becoming of considerable size was initially predicted by Satoshi Nakamoto in the original paper that introduced Bitcoin [1]. He offered a simple solution, *Simplified Payment Verification (SPV)*, that only requires a client to store block headers and leave out transactions. Still, the amount of data that needs to be downloaded from the network grows linearly with the size of the blockchain. FlyClient [12] allows a succinct and secure construction of proofs in a setting with variable difficulty. They make use of Merkle mountain ranges to reference the whole previous blockchain from every block. If a full node has a proof and mines a new block on top of it, they cannot create a new optimal proof without holding the whole chain. CoinPrune [13] still requires to store the entire chain of block headers prior to the pruning point. Another approach to build succinct proofs is to rely on SNARKS (Succinct Non-Interactive Argument of Knowledge). Coda [14] is such a construction. Coda compresses a chain to polylogarithmic size and updates the proof with new blocks. However, leveraging SNARKs requires a trusted setup for the common reference string.

2.2 Non-Interactive Proofs of Proof-of-Work

A Non-interactive Proof of Proof-of-Work (NIPoPoW) primitive aims at constructing a proof that is representative of the size of the original blockchain. Such a primitive has been proposed and instantiated by Kiayias *et al.* [9], and elegantly relies on the idea that sampling a small set of well chosen blocks is sufficient to estimate the size of the original blockchain [9, 10, 11]. To be kept as a sample, a block needs to satisfy a specific property on its cryptographic hash. The distribution of hash values is stochastic, and thus some blocks end up with hash values significantly below the mining target T. Blocks that hash to a value less than $T/(2^\ell)$ are called ℓ-superblocks [10], where T is the mining target and $\ell \geq 0$ is called the level of the block. The notion of ℓ-superblock reflects *block rarity*. In expectation, for a blockchain $\mathcal{C}$ of n blocks, only one block of $\mathcal{C}$ is a $(\log(n))$-superblock, two blocks of $\mathcal{C}$ are $(\log(n)-1)$-superblocks, ..., and all the blocks of $\mathcal{C}$ are 0-superblocks. The construction proposed by Kiayias *et al.* [9] consists in keeping the $2K$ most recent blocks of each level, where K is a security parameter, whose value by rule of thumb is set proportionally to the common

Table 1: Compression schemes. $|\mathcal{C}|$: number of blocks in chain $\mathcal{C}$, c: block header size, t_x: size of block transactions, k: common prefix parameter, a: snapshot size, χ: size of the uncompressed sub-chain.

	Storage	Communication	Online	Variable Difficulty	Adv.
BTC SPV	$\|\mathcal{C}\|c + \log(t_x)$	$\|\mathcal{C}\|c + \log(t_x)$		✓	1/2
Kiayias [10]	$\|\mathcal{C}\|c + kt_x + a$	$\text{poly}\log(\|\mathcal{C}\|)c + kt_x + a$			1/3
FlyClient [12]	$\|\mathcal{C}\|c + kt_x + a$	$\text{poly}\log(\|\mathcal{C}\|)c + kt_x + a$		✓	1/2
Kiayias [9]	$\text{poly}\log(\|\mathcal{C}\|)c + kt_x + a$	$\text{poly}\log(\|\mathcal{C}\|)c + kt_x + a$	✓		1/3
Jain [15]	$\text{poly}\log(\|\mathcal{C}\|)c + kt_x + a$	$\text{poly}\log(\|\mathcal{C}\|)c + kt_x + a$	✓		1/2
This work	$\text{poly}\log(\|\mathcal{C}\|)c + (\chi + k)t_x + a$	$\text{poly}\log(\|\mathcal{C}\|)c + (\chi + k)t_x + a$	✓	✓	1/3

prefix parameter k [3]. Their proof is resilient to a 1/3-adversary. The authors in [15] extend Kiayias *et al.* [9]'s scheme to obtain a proof that is correct in presence of a 1/2-bounded PPT adversary. Their main idea is to attach increasing weights $W_\beta(\ell)$ to ℓ-superblocks, making them "diamond-blocks" so that the selected proof is the heaviest. Because the adversary has minority mining power, they cannot create a heavier sequence of diamond blocks faster than the honest parties, for the same reason that an adversary cannot create a longer regular blockchain faster than the honest parties create one. Still, Jain et al. [15] prove the correctness of their construction in a static setting, that is assuming that the mining difficulty is constant.

Table 1 summarizes the storage and communication complexity of these constructions, indicates whether or not the construction is sufficient to create new blocks and to synchronize any newcomer, whether it fits a dynamic environment, and finally whether it is robust against a 1/2 or a 1/3-bounded PPT adversary. This work improves on existing solutions by providing a scheme that is both online, i.e. one can mine blocks directly from the proof, and works in a variable difficulty setting against a 1/3-bounded PPT adversary. The storage and communication costs of our scheme are comparable to Kiayias *et al.* [9] and Jain *et al.* [15]. We add one constant, keeping our scheme in polylogarithmic complexity.

3 Model of the system

We consider proof-of-work blockchains, that is blockchains whose block construction requires solving a resource consuming calculation [16]. We specifically focus on consensus data (i.e., application data is not considered as it is application dependent). We assume that each block contains a snapshot of the application state. Proofs of correctness of our construction relies on the model adopted by Kiayias et al [9]. Specifically, we consider a *synchronous* setting where time is quantized into discrete rounds [9] during which each party can send a message

[3] For any two blockchains $\mathcal{C}_1$ and $\mathcal{C}_2$, with $|\mathcal{C}_1| \leq |\mathcal{C}_2|$, we say that k is the common prefix of both blockchains, if $\mathcal{C}_1$, from which the k most recent blocks have been removed, is a prefix of $\mathcal{C}_2$.

to other parties, receive the messages sent to it during the round, and execute computational steps based on the received messages. We assume the presence of an adversary that models the behaviors of adversarial parties. We note a series $n = \{n_r\}_{r\in\mathbb{N}}$ to represent the total number of participants of the system over time, $t = \{t_r\}_{r\in\mathbb{N}}$ of which are adversarial. The relative proportion of malicious participants is bounded by a variable δ, typically called the advantage of honest parties, and we have $\forall r : t_r \leq (1-\delta)(n_r - t_r)$. Each party is allowed to make ρ queries to the cryptographic hash function in every round to create a block (specifically to find an adequate nonce). We say that a block b is valid if its header contains a nonce, along with non-double spending transactions, that hashes to a value h below a given mining target. The cryptographic hash function `h(.)` behaves as an ideal random function, and is modelled as a random oracle [17]. It produces a constant length κ output, where κ is a security parameter typically equal to 256. The adversary can query the cryptographic hash function up to $t_r \times \rho$ times per round [18]. Note f the probability that at least one of the honest queries is successful, i.e. the probability that at least one honest party successfully mines block during one round.

We suppose that the adversary is a *rushing adversary* in the sense that they can observe what the honest parties have done during the round before using their computational power at the end of the round. The adversary is also a *Sybil adversary* as they can inject as many additional messages as they wish by faking multiple identities. We limit the adversary to a probabilistic polynomial-time Turing machine that behaves arbitrarily. The adversary may thus not follow the prescribed algorithms, but it remains computationally bounded. Hence, it cannot, in a polynomial number of steps of time or space, forge honest party signatures or break the hash function and signature scheme with all but negligible probability. Therefore, we name our adversary the *1/3-bounded PPT adversary*. Any party following the prescribed protocol is called an *honest* party. In the following, blockchains will often be denoted as $\mathcal{C}, \ldots, \mathcal{C}'$, while blocks will be denoted by $b, \ldots, b'$.

4 The addressed problem

The addressed problem is the construction of a non-interactive proof that is representative of the underlying PoW-based blockchain $\mathcal{C}$. Such a proof is denoted by Π and is created with the NIPoPoW primitive. This primitive consists of two operations. Given a blockchain $\mathcal{C}$, $\texttt{Compress}(\mathcal{C})$ creates the proof Π, and given a set of proofs $\Pi = \{\Pi_1, \ldots, \Pi_n\}$, operation $\texttt{Compare}(\Pi_1, \ldots, \Pi_n)$ compares all the $\Pi_1, \ldots, \Pi_n$ and outputs one proof $\Pi_i \in \Pi$. The objective of this work is to instantiate both operations such that for any 1/3-bounded PPT adversary, the following properties hold:

Security: For any set of proofs $\Pi = \{\Pi_1, \ldots, \Pi_n\}$ such that among them at least one has been provided by an honest party, $\texttt{Compare}(\Pi_1, \ldots, \Pi_n)$ returns Π_i, representative of the most recent honest chain that accumulated the most work;

Succinctness: For any blockchain $\mathcal{C}$ maintained by an honest party, we have $|\texttt{Compress}(\mathcal{C})| = O(\text{poly}\log|\mathcal{C}|)$;

Onlineness. For any blockchain $\mathcal{C}$ maintained by an honest party, we have $\texttt{Compress}(\mathcal{C} \cdot b) = \texttt{Compress}(\Pi \cdot b)$;

where $\mathcal{C} \cdot b$ (resp. $\Pi \cdot b$) indicates that block b is appended to $\mathcal{C}$ (resp. proof Π). *Onlineness* is crucial for scalability, as it states that parties can mine blocks directly from the proof. The underlying blockchain does not need to be maintained, i.e., the proof is self-contained.

5 NIPoPoWs in a variable setting

5.1 Variable mining difficulty

Let $\mathcal{C}$ be some PoW-based blockchain. To guarantee that on average the time interval between any two successive block creations is constant, despite unpredictable variations of the global hashing power, miners in PoW-based blockchains periodically recalculate the block mining difficulty. The period of time between any two successive recalculations is called an epoch. In Bitcoin, an epoch is made of $m = 2016$ blocks, i.e. two weeks. The mining difficulty at epoch i represents how much more difficult it is to mine a block at epoch i than it was when the genesis block of the blockchain was created. The mining difficulty is inversely proportional to the mining target [1]. This last notion refers to the PoW inequation satisfied by any block b of a PoW-based blockchain, i.e., $\mathtt{h}(b) \leq T_i$, where T_i represents the mining target at epoch i. Let $\|b\|$ refer to the difficulty sealed in b's header. If $\|b\| = 1/T_i$, block b is valid for epoch i. By abuse of notation we write $b \in T_i$ if b was mined during epoch i.

5.2 Evaluating the significance of a block

We revisit the definition of block level, introduced in Kiayias *et al* [9]. The level of a block is an indicator of its *significance*: the higher the level of a block is, the more it is representative of the number of blocks of $\mathcal{C}$. The level of a block is a projection of its hash value over its mining period onto the unit interval $[0, 1]$. A level may contain blocks whose mining target is different, which just reflects the fact that those blocks are equally representative of the original blockchain, but have been created in epochs that involved different computational powers.

Definition 1. *(Level of a block) Let κ be the security parameter of the mining process, then for any block $b \in \mathcal{C}$ such that $b \in T_i$, the level of b is defined as*

$$\texttt{Level}(b) = \max\left\{\ell \in [\![0, \kappa - 1]\!] \;\middle|\; \frac{\mathtt{h}(b)}{T_i} \leq \frac{1}{2^\ell}\right\}.$$

By definition, a block of level ℓ is also a block of level ℓ' for all $0 \leq \ell' < \ell$. We say that the level of block b is ℓ if ℓ is the largest value for which $\mathtt{h}(b)/T_i \leq 1/2^\ell$

holds. Figure 2 in Appendix A shows the number of blocks as a function of their level. By convention, the genesis block has an infinite level. A block of level ℓ is called a ℓ-block. On average, one needs to create 2^ℓ blocks to obtain one block of level ℓ. By sampling only the last blocks of each level, one will be able to exponentially compress the blockchain.

5.3 Construction of a NIPoPoW

NIPoPoWs requires two operations: the `Compress`() operation that builds the NIPoPoW, and the `Compare`() operation that selects among a set of collected NIPoPoWs the one that reflects the "best" underlying blockchain. Our instantiation of these operations are close to the ones proposed in [9]. This is very interesting because it keeps the elegance of the original construction, and it makes ours adapted to both static and variable difficulties. Apart from the block level function which guarantees the scalability of our construction, our instantiation ensures that any verifier is capable of auditing the difficulty of the NIPoPoW blocks, prevents low difficulty attacks, and copes with the extremely rare scenario in which NIPoPoWs have no blocks in common, i.e., NIPoPoWs that reflect blockchains whose sizes are extremely different.

Notations used in the algorithms We note $\mathcal{C}$ an interlinked blockchain, with $\mathcal{C}[i]$ denoting its i^{th} element. $\mathcal{C}[:j]$ denotes blocks from the genesis blocks to the j^{th} block exclusive, and $\mathcal{C}[i:]$ denotes blocks from the i^{th} element inclusive to the end of $\mathcal{C}$. $\mathcal{C}[i:j]$ denotes blocks from the i^{th} element inclusive to the j^{th} block exclusive. Block indices i and j can be replaced by blocks A and Z. We then write $\mathcal{C}\{A:Z\}$ to designate the chain from block A inclusive to Z exclusive. Again, any end can be omitted. A negative index means to take blocks from the end, $\mathcal{C}[-1]$ denotes the tip of $\mathcal{C}$. We write $\mathcal{C}\uparrow^\mu$ to mean the subsequence of $\mathcal{C}$ containing only its μ-blocks. The $\mathcal{C}\uparrow$ operator is absolute: $(\mathcal{C}\uparrow^\mu)\uparrow^{\mu+i} = \mathcal{C}\uparrow^{\mu+i}$. Since $\mathcal{C}$ is interlinked, $\mathcal{C}\uparrow^\mu$ is a chain too. Given two chains $\mathcal{C}_1$ and $\mathcal{C}_2$, $\mathcal{C}_1 \subseteq \mathcal{C}_2$ means that all of $\mathcal{C}_1$'s blocks are in $\mathcal{C}_2$. We denote $\mathcal{C}_1 \cup \mathcal{C}_2$ the chain consisting of all blocks in either chains. $\mathcal{C}_1 \cap \mathcal{C}_2$ denotes the chain consisting of blocks only in both chains. We note $\mathcal{C}_1 \setminus \mathcal{C}_2$ the chain consisting of blocks in $\mathcal{C}_1$ but not $\mathcal{C}_2$. The chain filtering operators $[\cdot]$, $\{\cdot\}$, $\uparrow$ have precedence over the $\cup$, $\cap$ and $\setminus$ operators.

Compressing a blockchain The `Compress`() algorithm aims at sampling a polylogarithmic number of well chosen blocks from the blockchain. Only those blocks will appear in the NIPoPoW, i.e. all the other ones will be pruned. This requires every block header to keep pointers to the last preceding block of every level so that sampled blocks form a valid chain. The algorithm is parameterized by the security parameters K, χ and k. Parameter K is dimensioned to be large enough so that, for a blockchain made of $|\mathcal{C}|$ blocks, K guarantees that with any high probability a 1/3-adversary cannot exhibit more difficulty in its own K sampled blocks of level ℓ, for all $0 \leq \ell \leq \log|\mathcal{C}|$, than what the honest parties can show. Parameter χ represents the minimum number of blocks necessary for any

verifier to check that the most recent part of the NIPoPoW exhibits a correctly computed mining difficulty. Finally, parameter k represents the common prefix parameter [19]. Once a block is anchored more than k blocks deep in a blockchain, this block has a negligible probability to be pruned from the main blockchain by a fork. Note that in the static construction (e.g., [9, 15]) it is assumed that all the blocks of the blockchain have been created with the same constant difficulty. Thus, there is no need for an uncompressed part in those cases.

Specifically, the $\texttt{Compress}_{K,\chi,k}()$ algorithm first separates the blockchain into three sub-chains. The *unstable*, *uncompressed*, and *compressed* sub-chains. The unstable sub-chain contains the k most recent blocks of the blockchain. The term unstable refers to the fact that those k most recent blocks have a non negligible probability to be pruned during a fork resolution. The analytical dimensioning of the common prefix parameter k by Garay *et al.* [19] in a variable setting shows that k is proportional to the ratio m/τ, where m is the length of an epoch and τ is the dampening factor in Bitcoin's mining target recalculation. A conservative evaluation gives $k = 323$. Those k blocks are returned as such by the $\texttt{Compress}_{K,\chi,k}()$ algorithm. The uncompressed sub-chain contains the χ most recent blocks of the blockchain after having removed the k most recent ones. We set $\chi = 2m$ as this is the minimal value to keep all blocks of at least the most recent epoch (recall that an epoch contains 2016 blocks). Those χ blocks are also returned as such by the algorithm. Finally, the compressed sub-chain is built by sampling at least $2K$ but no more than $4K$ blocks per level of the blockchain. Specifically, the algorithm determines the highest level ℓ of the blockchain, that is the highest level that contains at least $2K$ ℓ-blocks. Note that the few blocks whose level is larger than ℓ are also kept at level ℓ. This guarantees that appending new blocks to the proof progressively leads to higher block levels. Then for each lower level $0 \leq \mu < \ell$, the $2K$ most recent μ-blocks are kept, as well as all μ-blocks coming after the K^{th} block of level $\mu+1$, counting from the end. This guarantees that any two consecutive levels of the construction intersect in at least K blocks. The compressed sub-chain is also returned by the algorithm, as well as the highest level ℓ. Hence, the NIPoPoW consists of the chain of blocks obtained by appending those three sub-chains. Due to space constraints, pseudo-code of the $\texttt{Compress}_{K,\chi,k}(\mathcal{C})$ algorithm appears in Appendix B. Figure 3 in Appendix A shows the compression of the first blocks of the Bitcoin blockchain.

Comparison algorithm The $\texttt{Compare}()$ algorithm is fed with NIPoPoWs. Note that it also accepts full blockchains as input, since its first step is to $\texttt{Compress}()$ each of its inputs to be able to compare them on an equal basis. The algorithm is parameterized by the same parameters as the $\texttt{Compress}_{K,\chi,k}()$ one. It compares iteratively its next NIPoPoW (i.e., the next input) with the best current one. Specifically, given two NIPoPoWs to be pairwise compared, the algorithm looks for the smallest level μ at which both NIPoPoWs share a common block b. Block b is called the *last common ancestor* (LCA) of both NIPoPoWs. The existence or not of a LCA between any two NIPoPoWs provides crucial infor-

mation on their underlying full blockchains. Firstly, it indicates that their sizes, which in average are equal respectively to $2K2^{\ell}$ and $2K2^{\ell'}$, for some $\ell, \ell' \geq 0$ (see Definition 1) are very close to one another, that is $\ell = \ell'$. Secondly, a LCA at level μ guarantees that the length of the common prefix of both underlying blockchains is in average equal to $2K(2^{\ell} - 2^{\mu-1})$ blocks. Note that if $\mu = 0$ then both underlying blockchains are almost similar except possibly for their most recent $2K + \chi + k$ blocks. Hence, two blockchains are comparable if their NIPoPoWs share a LCA. Thus, just comparing the accumulated difficulty of both NIPoPoWs from the LCA block is indicative of the best underlying blockchain. Specifically, the $\texttt{Compare}_{K,\chi,k}()$ algorithm computes, for both NIPoPoWs, the sum of difficulties of blocks following block b in the compressed sub-chain plus the sum of difficulties of all the blocks in the uncompressed and the unstable sub-chains. The best NIPoPoW is the one that accumulates the most difficulty. The algorithm iterates the same pairwise comparison process on the next inputs (if any). If each pairwise comparison exhibits a LCA, then the algorithm returns the best current NIPoPoW. On the other hand, the absence of common ancestor between any two NIPoPoWs reveals strongly dissimilar underlying blockchains. Such a dissimilarity is either due to the presence of a fork that has never been resolved since nearly the inception of both blockchains. The other reason is a long-lasting obsolescence of one of the two underlying blockchain. Assuming the absence of network partition, the former case indicates that one of the two NIPoPoWs is adversarial while the second case reflects a blockchain that has almost never been updated with new blocks. The most efficient way that allows a verifier to handle such a situation is to create and diffuse a transaction including a privately generated random beacon and then observing the network for at least K/f rounds. The synchrony assumptions and the adversarial model guarantees that any honest transaction sent in a round is received by all honest parties in the same round, and that on average it takes K/f rounds to create K blocks, where K is our security parameter and f is the probability for honest parties to create at least one block during one round. After K/f rounds, the verifier considers all the blockchains that exhibit a block containing its beacon, and picks the heaviest one. The verifier can safely keep the NIPoPoW generated from the chain with the most difficulty, as the existence of the beacon prevents the adversary from mining blocks in advance. For space limitation reasons, pseudo-code of the $\texttt{Compare}_{K,\chi,k}(\mathcal{C}_1, \ldots, \mathcal{C}_n)$ algorithm is given in Appendix B.

6 Analysis

6.1 Preliminary definitions

Our analysis relies on Garay *et al.*'s Bitcoin Backbone with chains of variable difficulty model [19], as well as concepts introduced in Kiayias *et al.*'s protocol [9]. The analysis strongly relies on Garay *et al.*'s notion of typical executions [19]. An execution E of the protocol generates a sequence S of rounds. An execution E is typical if the considered random variables do not deviate too much from their expected values, *i.e.* at distance ε of their expected value. From Garay

et al. [19] we have $|S| > m/(16\tau f)$, where m represents the number of blocks defining an epoch, i.e. $m = 2016$ for Bitcoin, f is the probability for honest parties to create at least one block in a round, and τ is the dampening filter in Bitcoin target's recalculation function. A round r is successful if the honest parties succeed in creating at least one block during r, and is uniquely successful if the honest parties succeed in creating exactly one block during r. Given a chain $\mathcal{C}$, and $d \in \mathbb{R}^+$, $d \in C[i]$ if $\|\mathcal{C}[:i]\| \leq \|\mathcal{C}[:i]\| + d < \|\mathcal{C}[:i+1]\|$. We denote by Q_r the random variable that represents the difficulty of the block created during a uniquely successful round r. If r is not uniquely successful then $Q_r = 0$. By extension, $\sum_{r \in S} Q_r$ represents the accumulated difficulty obtained during an execution of S rounds. Quantity $\sum_{j \in J} A_j$ represents the accumulated difficulty obtained by the adversarial parties over a set of queries J. Considering a set of queries rather than a set of rounds allows the adversary to target specific rounds during which it queries the hash function. Furthermore, by adopting a similar approach as the one of Kiayias *et al.* [9], we generalize our results by using the notion of block property satisfying a predicate P on its hash output $h \in \{0,1\}^k$ over its mining target T. A P-block is a block such that $P(h/T)$ = true. Probability ξ_P quantifies the probability that a block satisfies predicate P. By abuse of notation, we denote by $\mathcal{C} \uparrow^P$ the set of P-blocks of $\mathcal{C}$. Finally, we denote by ϕ the quantity $\rho/2^\kappa$. For self-containment reasons, Appendix C provides the formalisation of all the notions we use in our proofs. More details can be found in Garay et al. [19]. Appendix D summarizes all the variables used in this article.

6.2 Suppression of honest difficulty

The following technical lemmas will be deeply used to prove the correctness of our solution. The two first ones show the condition under which the adversary may impose its chain to honest parties (i.e., can suppress some of the honest difficulty in the blockchain). On the other hand, Lemma 3 states that in executions long enough, the honest difficulty of the adopted blockchain dominates the adversarial one. Proofs of these lemmas extend the ones presented in Kiayias *et al.* [9]. Observe that in the constant mining difficulty setting [9] the height of a blockchain (i.e., its number of blocks) is equivalent to its accumulated difficulty as all the blocks are created with the same difficulty. In the variable mining difficulty setting, there is some discrepancy between the height of a blockchain and its accumulated difficulty and so to compare $\mathcal{C}$ and $\mathcal{C}'$, we need to consider the different possible relationships between the height of a chain and its associated difficulty. For space constraint reasons, we present the proofs of these three technical lemmas to Appendix E.

Lemma 1 (Pairing). *Consider a execution with an honest party, an adversary and two chains $\mathcal{C}$,$\mathcal{C}'$. For any pair of distinct blocks $\mathcal{C}[i]$ and $\mathcal{C}'[i']$ such that $\exists d \in \mathbb{R}^+, d \in \mathcal{C}[i]$ and $d \in \mathcal{C}'[i']$, if $\mathcal{C}[i]$ was computed by an honest party in a uniquely successful round, then $\mathcal{C}'[i']$ was computed by the adversary.*

Lemma 2 (Suppression). *If r is a uniquely successful round and the corresponding block b does not belong to the chain of an honest party at a later round, then there is a set of consecutive rounds S and a set J of adversarial queries in S such that $r \in S$ and $\sum_{r\in S} Q_r \leq \sum_{j\in J} A_j$.*

Lemma 3 (Unsuppressibility). *Given a typical execution with an honest party and an adversary, every set of consecutive rounds U has a subset S of uniquely successful rounds, such that the following conditions hold:*

1. $\sum_{r\in S} Q_r \geq \sum_{r\in U} Q_r - 2\sum_{j\in U} A_j - (1+\varepsilon)\phi \sum_{r\in 2\frac{m}{16\tau f}} t_r$,
2. *After the last round in S the blocks corresponding to S belong to the chain of every honest party.*

6.3 Security, succinctness and onlineness

Intuitively, the P-block Common-Prefix Lemma shows that while the adversary is able to accumulate difficulty in its private blockchain for a given period of time, at some point the honest parties will accumulate more difficulty than what the adversary has achieved to do.

Lemma 4 (P-block Common-Prefix). *Given a typical execution with an honest party and a $1/3$-bounded PPT adversary, such that a chain $\mathcal{C}$ is adopted by the honest party at round r, and there exists another chain $\mathcal{C}'$ such that P-blocks of $\mathcal{C}' \setminus (\mathcal{C}' \cap \mathcal{C})$ have at least $2(1-\varepsilon)(1+\varepsilon)\xi_P(|W'|/2^\kappa - \phi m\gamma t/16\tau f)$ difficulty, then with overwhelming probability, we have $\|\mathcal{C} \uparrow^P\| > \|\mathcal{C}' \uparrow^P\|$.*

Proof. Let us consider an an execution that satisfies the assumptions of the lemma. Let r^* be the round during which the last honest block on $\mathcal{C}^* = \mathcal{C} \cap \mathcal{C}'$ was computed. If no such block exists, we set $r^* = 0$. Let us consider the set of rounds $S = \{i : r^* < i \leq r\}$ as well as two subsets $S_1 = \{i : r^* < i < r^* + m/16\tau f\}, S_2 = \{i : r^* + m/16\tau f \leq i \leq r\}$. We will study the execution during the rounds in S. Let us denote by W' the set of adversarial queries on $\mathcal{C}' \setminus \mathcal{C}^*$ at some round $r \in S_2$. Finally, we denote by W the rest of the adversarial queries, *i.e.* those executed on $\mathcal{C}' \setminus \mathcal{C}^*$ at round $r \in S_1$ or on $\mathcal{C} \setminus \mathcal{C}^*$ at any round $r \in S$.

Using Lemma 3, there is at least $\sum_{r\in S} Q_r - 2\sum_{w\in W} A_w - (1+\varepsilon)\phi \sum_{r\in 2\frac{m}{16\tau f}} t_r$ difficulty that the adversary cannot suppress on $\mathcal{C}$. Each of those rounds contributing difficulty produce a P-block independently with probability ξ_P. Thus with overwhelming probability, the total quantity of difficulty in P-blocks on $\mathcal{C} \setminus \mathcal{C}^*$ is at worst A_P, where A_P is defined in Equation (1).

$$A_P = (1-\varepsilon)\xi_P \cdot \left[\sum_{r\in S} Q_r - 2\sum_{w\in W} A_w\right] - (1-\varepsilon)\xi_P(1+\varepsilon)\phi \sum_{r\in 2\frac{m}{16\tau f}} t_r \qquad (1)$$

On the other hand, the difficulty quantity of P-blocks on $\mathcal{C}' \setminus \mathcal{C}^*$ is at most the difficulty in P-blocks from the W' queries and the queries executed during

S_1. The latter can be shown to be at most $(1+\varepsilon)\xi_P\phi\sum_{r\in\frac{m}{16\tau f}}(n_r-t_r)$ difficulty. The former, in a typical execution, are at most $(1+\varepsilon)\xi_P\sum_{w'\in W'}A_{w'}$ difficulty. Thus, the difficulty in P-blocks on $\mathcal{C}'\setminus\mathcal{C}^*$ is at most B_P, where

$$B_P=(1+\varepsilon)\xi_P\sum_{w'\in W'}A_{w'}+(1+\varepsilon)\xi_P\phi\sum_{r\in m/16\tau f}(n_r-t_r) \tag{2}$$

We show that $A_P-B_P\geq(1-\varepsilon)\xi_P(\sum_{r\in S}Q_r-2\sum_{j\in J}A_j)$, where this lower bound is positive in a typical execution in which the power of the adversary is bounded below 1/3 of the total power. We have $A_P-B_P=\xi_P((1-\varepsilon)\left(\sum_{r\in S}Q_r-2\sum_{j\in J}A_j\right)+(1-3\varepsilon)\sum_{w'\in W'}A_{w'}-(1-\varepsilon)(1+\varepsilon)\phi\sum_{2m/16\tau f}t_r-(1+\varepsilon)\phi\sum_{m/16\tau f}(n_r-t_r))$. Since $0<\xi_P\leq 1$, this is equivalent to showing that

$$(1-3\varepsilon)\sum_{w'\in W'}A_{w'}-(1-\varepsilon)(1+\varepsilon)\phi\sum_{2m/16\tau f}t_r-(1+\varepsilon)\phi\sum_{m/16\tau f}(n_r-t_r)\geq 0$$

To get the required amount of difficulty on $\mathcal{C}'$, we need to find the amount a of difficulty such that $(1+\varepsilon)\sum_{w'\in W'}A_{w'}+(1+\varepsilon)\phi\sum_{r\in m/16\tau f}(n_r-t_r)\geq a$. The inequality is verified with $a=2(1-\varepsilon)\sum_{w'\in W'}A_{w'}-(1-\varepsilon)(1+\varepsilon)\phi\sum_{2m/16\tau f}t_r$. Following Definition 8 (Typical Execution) and Fact 1 (see Appendix C) from [19], and re-injecting the probability ξ_P, we obtain the bound B_P, i.e., $B_P\geq 2(1-\varepsilon)(1+\varepsilon)\xi_P(|W'|/2^\kappa-\phi m\gamma t/16\tau f)$. This completes the proof. □

Theorem 1 (Security). *Given a typical execution with an honest party and a 1/3-bounded PPT adversary, such that at round r the honest party has chain $\mathcal{C}$, and its corresponding proof $\Pi=\texttt{Compress}_{K,\chi,k}(\mathcal{C})$, and the adversary has chain $\mathcal{C}'$, and its corresponding proof $\Pi'=\texttt{Compress}_{K,\chi,k}(\mathcal{C}')$ and let $\Pi^*=\texttt{Compare}_{K,\chi,k}(\Pi,\Pi')$ be the proof accepted by the verifier, then, with overwhelming probability, we have $\|\Pi^*\{(\Pi^*\cap\mathcal{C})[-1]:\}\|\geq\|\mathcal{C}\{(\Pi^*\cap\mathcal{C})[-1]:\}\|$.*

Proof. Let M the set of levels where both chains shared at least one block, $M=\{\mu\in\mathbb{N}\mid\Pi\uparrow^\mu\cap\Pi'\uparrow^\mu\neq\emptyset\}$, and let $\mu=\min(M)$ if $M\neq\emptyset$, $\mu=\perp$ otherwise (Section 5.3). Three cases must be considered according to μ's value.

- $\mu=0$**:** Recall that the verifier chooses the heaviest proof. $(\Pi^*\cap\mathcal{C})[-1]$ refers to the last block shared by Π^* and $\mathcal{C}$, and $\Pi^*\{(\Pi^*\cap\mathcal{C})[-1]:\}$ refers to this very same block and the set of blocks in Π^* following it. Since we have $\mu=0$, this last common block belongs to the last $2K+\chi+k$ blocks of both chain $\mathcal{C}$ and proof Π^*. At round r, the honest party has $\mathcal{C}$, *i.e.* $\|\mathcal{C}\|\geq\|\mathcal{C}'\|$. Thus, the only possibility to have $\Pi^*=\Pi'$ is that the adversary successfully mined a block on top of Π, which is a valid extension heavier than the honest proof.
- $\mu>0$**:** By definition of μ, $\Pi\uparrow^{\mu-1}\cap\Pi'\uparrow^{\mu-1}=\emptyset$. By construction, both proofs hold at least $2K(\mu-1)+\chi+k$ blocks after their common block b (Section 5.3). By Lemma 4, Π is accepted.
- $\mu=\perp$**:** In this case, the proofs do not share a common block. We thus use the beacon method to create this reference block. After K/f rounds, we are able to pick the chain containing the beacon which has the most work, and thus determine the value of Π^* accordingly. □

Theorem 2 (Succinctness). *For any infinite sequence S of rounds and for any subsequence $S_j \subseteq S$ of consecutive rounds, any honest miner stores $O(K^2 \log(r))$ blocks at round $r \in S_j$.*

Proof. Consider a proof Π generated by an honest prover from an underlying chain $\mathcal{C}$, and suppose for contradiction that $|\Pi| \in \omega(K \log(r))$. Consider $(\mathcal{D}, X, \Omega, \ell) = \texttt{Compress}(\mathcal{C})$. By assumption on Π, we have $\sum_{\mu \in \mathbb{N}} |\mathcal{D}[\mu]| \in \omega(K \log(r))$, since χ, k are constant. Let $\ell = \max\{\mu \in \mathbb{N} \mid \mathcal{D}[\mu] \neq \emptyset\}$, and let us consider some $\mu \leq \ell$ such that $\mathcal{D}[\mu] \in \omega(K \log(r))$ and thus $\mathcal{D}[\mu] \in \Omega(K^2 \log(r))$. Note r_0, r_1 the rounds in which $\mathcal{D}[\mu][0]$, $\mathcal{D}[\mu][-1]$ were created respectively. Consider U the set of consecutive rounds between r_0 and r_1. We have $|U| \geq |\mathcal{D}[\mu]| \geq K$. By definition of μ, at least K^2 blocks must have been created during U, among which $(1-\varepsilon)K^2/2 \geq 2K$ are $(\mu+1)$-blocks which belongs to $\mathcal{D}[\mu+1]$ as well. The case $\mu = \ell$ contradicts the maximality of ℓ, and thus the assumption $\Pi \in \omega(K \log(r))$. If $\mu < \ell$, then we have $\mathcal{D}[\mu] \in \Omega(K^2 \log(r))$, and $\mathcal{D}[\mu+1] \in O(K)$. Since $\mathcal{D}[\mu] = \mathcal{C}[: -\chi - k] \uparrow^{\mu} [-2K :] \cup \mathcal{C}[: -\chi - k] \uparrow^{\mu} \{\mathcal{D}[\mu+1][-K] :\}$ and $|\mathcal{C}[: -\chi - k] \uparrow^{\mu} [-2K :]| = 2K$, we have $\mathcal{C}[: -\chi - k] \uparrow^{\mu} \{\mathcal{D}[\mu+1][-K] :\} \in \Omega(K^2 \log(r))$. Since $\mathcal{D}[\mu+1] \in O(K)$, $|\mathcal{D}[\mu]| \leq 2K + O(K)$ which contradicts $\mathcal{D}[\mu] \in \Omega(K^2 \log(r))$, completing the proof. □

Theorem 3 (Onlineness). *Let $\Pi = \texttt{Compress}_{K,\chi,k}(\mathcal{C})$ generated about an honest chain $\mathcal{C}$, and let b a block mined on top of $\mathcal{C}$. We have $\texttt{Compress}_{K,\chi,k}(\mathcal{C} \cdot b) = \texttt{Compress}_{K,\chi,k}(\Pi \cdot b)$.*

Proof. Recall that given $\mathcal{C}$ (resp. Π) and a block b, $\mathcal{C} \cdot b$ (resp. $\Pi \cdot b$) refers to the concatenation of b to $\mathcal{C}$ (resp. to Π). Given $(\mathcal{D}, X, \Omega, \mu) = \texttt{Compress}(\mathcal{C})$, $\mathcal{D}$ gathers $2K$ last blocks from ν-blocks of the compressed sub-chain and $\mu = \max_{\mu \in \mathbb{N}} \Pi \uparrow^{\mu} \neq \emptyset$. Let ℓ the level of block b. Obviously, for any level $\ell < \nu \leq \mu$, we have $(\mathcal{C} \cdot b) \uparrow^{\nu} = \mathcal{C} \uparrow^{\nu} = \Pi \uparrow^{\nu} = (\Pi \cdot b) \uparrow^{\nu}$. In addition, for any level $0 < \nu \leq \ell$, we have $\Pi \uparrow^{\nu} \subseteq \mathcal{C} \uparrow^{\nu}$ and thus $(\Pi \cdot b) \uparrow^{\nu} \subseteq (\mathcal{C} \cdot b) \uparrow^{\nu}$. Let us consider a level $0 \leq \nu \leq \ell$. Recall that *i)* the $\texttt{Compress}()$ algorithm determine a pivot $b^{\nu}_{\mathcal{C}}$, the K-th most recent block at level $\nu+1$ of a chain $\mathcal{C}$, *ii)* any block of level ν is also a block of levels lower than ν. As a consequence, for any level $0 \leq \nu \leq \ell$, we have $b^{\nu}_{\mathcal{C} \cdot b} = b^{\nu}_{\Pi \cdot b}$, and thus $(\mathcal{C} \cdot b)\{b^{\nu}_{\mathcal{C}} :\} = (\Pi \cdot b)\{b^{\nu}_{\Pi} :\}$. Similarly, for any level $0 \leq \nu \leq \ell$, the same argument holds for blocks preceding pivot b^{ν}_{Π}, we have $(\Pi \cdot b) \uparrow^{\mu} \{: b^{\nu}_{\Pi}\} \subseteq (\mathcal{C} \cdot b) \uparrow^{\mu} \{: b^{\nu}_{\mathcal{C}}\}$, which completes the proof. □

6.4 Dimensioning the security parameter K

All the properties of our NIPoPoW instantiation, namely Security (Theorem 1), Succinctness (Theorem 2) and Onlineness (Theorem 3) rely on security parameters K, χ and k. We dimension k and χ in Section 5.3. Security parameter K must guarantee that with any high probability a 1/3-adversary cannot exhibit more difficulty in its own K sampled blocks than what the honest parties can do. This ensures that the $\texttt{Compare}_{K,\chi,k}()$ algorithm, when fed with a set of NIPoPoWs such that among them at least one is an honestly generated one,

will return the best NIPoPoW, i.e. the NIPoPoW that represents the blockchain with the largest accumulated difficulty.

We model the protocol as a competition between an honest party and the adversary. We denote by Π the honest NIPoPoW and by Π' the adversarial one. We assume that the competition takes place within an epoch which means that the honest target T is constant during the competition. The competition starts at the first time where both chains fork. We denote by $b^\star$ the last common block: $b = (\Pi \cap \Pi')[-1]$. At each step of the competition, a unique block is created, either by the honest party or the adversary. We assume that the adversary may potentially mine blocks at a different difficulty from that of the honest parties. We denote by $\alpha = a/b$ the ratio between honest and adversarial blocks in terms of difficulty. We refer by n the number of blocks in the honest chain that do not belong to the adversarial chain, we have $n = |\Pi\{b^\star :\}|$. We are interested in determining the earliest step in which the adversarial chain overcomes the honest one, after n blocks have been appended to the honest chain. We denote this event by C_n^α.

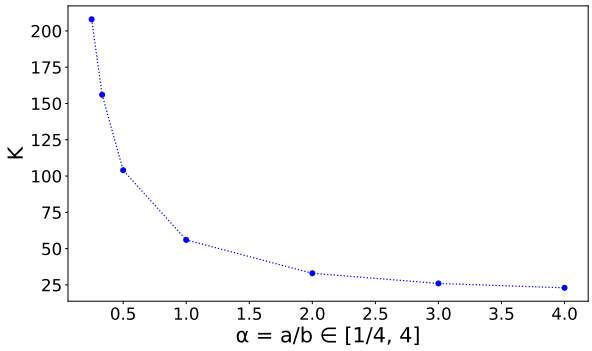
To determine $\mathbb{P}(C_n^\alpha)$, we use the classical gambler's ruin problem [20] and the Poisson distribution. The classical gambler's ruin problem provides us $P_{ruin}(M)$, the probability that the adversary will catch up, given an initial gap M ($M \in \mathbb{N}$) in the quantity of work between his chain and the honest chain. Following [20], two cases must be considered. If the initial gap M is less than b, the quantity of difficulty of an adversarial block, i.e., $M < b$, the adversary has already caught up, in which case we have $P_{ruin}(M) = 1$. On the contrary, if $M \geq b$, we have $P_{ruin}(M) < 1$. We provide details on the derivation of P_{ruin} in Appendix F.

Consider now the step where the honest party appends its n-th block (presenting a difficulty of a) on its chain. We denote by I_n the time interval during which the honest party produces n blocks. Regarding the adversary, he may have appended $i \geq 0$ blocks (each one having b difficulty) on its chain. Following [21], random variable X represents the number of blocks produced by the adversary. X follows a Poisson distribution with parameter λ, which represents the average number of blocks the adversary produces during I_n. The difference of accumulated difficulties between both chains is t given by $na - ib$. Since $\{X = i; \lambda\}_{i \geq 0}$ is a complete system of events with $\mathbb{P}(X = i; \lambda) \neq 0$ for $i \geq 0$ and $P_{ruin}(na - ib) = \mathbb{P}(C_n^\alpha \mid \{X = i; \lambda\})$ for $i \geq 0$, we can determine $\mathbb{P}(C_n^\alpha)$ using the law of total probability:

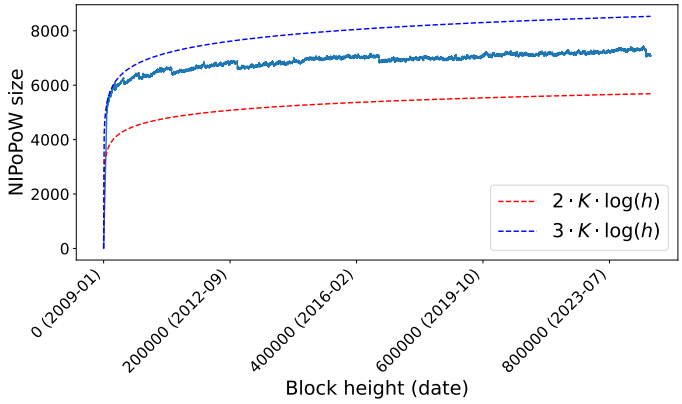
$$\mathbb{P}(C_n^\alpha) = \sum_{i=0}^{\infty} \mathbb{P}(X = i; \lambda) \cdot P_{ruin}(na - ib)$$

$$\mathbb{P}(C_n^\alpha) = 1 - \sum_{i=0}^{\lfloor n\alpha - 1 \rfloor} \frac{\lambda^i e^{-\lambda}}{i!} \left(1 - P_{ruin}(na - ib)\right).$$

Given the expression of $\mathbb{P}(C_n^\alpha)$, and a security parameter ε, we can define K as follows $K = \inf_{n \geq 0} \mathbb{P}(C_n^\alpha) < \varepsilon$. Figure 1a plots the value of K as a function of α with $\varepsilon = 10^{-6}$. A secure implementation should take the largest value of K to cope with a variation of $\alpha = 1/4$, i.e. $K = 208$. Finally we



(a) Parameter K as a function of α in presence of a 1/3 adversary.



(b) Evolution of the number of blocks kept in the proof over blockchain length.

Fig. 1: Parameter K as a function of the choice adversarial target, and chain compression on Bitcoin Mainnet (865,042 blocks as of 2024-10-11).

have experimentally compressed the current Bitcoin blockchain with the settings $K = 208, \chi = 4032, k = 323$. Figure 1b shows the obtained NIPoPoW size in the number of blocks. Its size varies polylogarithmically with the size of Bitcoin's blockchain length, which illustrates the succinctness property.

Our code is publicly available here for the calculation of K, and here for the implementation of our scheme in Python. The Python implementation uses the Bitcoin blockchain historical data to generate the proofs.

7 Conclusion

We have presented an instantiation of a NIPoPoW that handles variable mining difficulty. This relies on an original block level definition and a precise dimension-

ing of the security parameter K that defends the system against a 1/3-adversary, and in particular against low difficulty attacks.

As future works, we intend to extend the solution proposed in [15]. We are pretty confident that handling block difficulties and weight should improve the resiliency of our solution to a 1/2-adversary. Our scheme deeply relies on the stochastic and cryptographic properties of the hash function in the PoW. Given scalability properties of NIPoPoWs, we intend to study their instantiation to DAG-based PoW-based blockchains. Another challenging objective is to instantiate them to PoS-based permissionless blockchains (e.g. [22, 23]) by exploiting the randomness present in blocks.

8 References

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

[2] SNAP. **2020**. Ethereum Snapshot Protocol.

[3] Poon, J. and Dryja, T. **2016**. The Bitcoin Lightning Network: Scalable Off-Chain Instant Payments.

[4] Kiayias, A. and Litos, O. S. T. **2020**. A composable security treatment of the lightning network. In: *IEEE Computer Security Foundations Symposium*. CSF.

[5] Back, A., Corallo, M., Dashjr, L., Friedenbach, M., Maxwell, G., Miller, A., Poelstra, A., Timón, J. and Wuille, P. **2014**. Enabling blockchain innovations with pegged sidechains.

[6] Nick, J., Poelstra, A. and Sanders, G. **2020**. Liquid: A Bitcoin Sidechain.

[7] Kiayias, A., Gazi, P. and Zindros, D. **2019**. Proof-of-stake sidechains. In: *IEEE Symposium on Security and Privacy*. S&P.

[8] Chepurnoy, A., Srinivasan, S. and Zhang, Y. **2018**. EDRAX: A Cryptocurrency with Stateless Transaction Validation.

[9] Kiayias, A., Leonardos, N. and Zindros, D. **2021**. Mining in Logarithmic Space. In: *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*. CCS.

[10] Kiayias, A., Miller, A. and Zindros, D. **2020**. Non-interactive Proofs of Proof-of-Work. In: *Financial Cryptography and Data Security*. Cham.

[11] Karantias, K., Kiayias, A. and Zindros, D. **2020**. Compact storage of superblocks for NIPoPoW applications. In: *Mathematical Research for Blockchain Economy*.

[12] Bünz, B., Kiffer, L., Luu, L. and Zamani, M. **2020**. FlyClient: Super-Light Clients for Cryptocurrencies. In: *IEEE Symposium on Security and Privacy*. S&P.

[13] Matzutt, R., Kalde, B., Pennekamp, J., Drichel, A., Henze, M. and Wehrle, K. **2021**. CoinPrune: Shrinking Bitcoin's Blockchain Retrospectively. *IEEE Transactions on Network and Service Management* 18.3.

[14] Bonneau, J., Meckler, I., Rao, V. and Shapiro, E. **2020**. Coda: Decentralized Cryptocurrency at Scale.

[15] Jain, A., Anceaume, E. and Gujar, S. **2023**. Extending The Boundaries and Exploring The Limits Of Blockchain Compression. In: *IEEE 43th Symposium on Reliable and Distributed Systems*. SRDS.

[16] Back, A. **2002**. Hashcash - A Denial of Service Counter-Measure.

[17] Bellare, M. and Rogaway, P. **1993**. Random oracles are practical: A paradigm for designing efficient protocols. In: *ACM conference on Computer and Communications Security*. CCS.

[18] Garay, J., Kiayias, A. and Leonardos, N. **2015**. The Bitcoin Backbone Protocol: Analysis and Applications. In: *Advances in Cryptology*. EUROCRYPT.

[19] Garay, J., Kiayias, A. and Leonardos, N. **2017**. The Bitcoin Backbone Protocol with Chains of Variable Difficulty. In: *Annual International Cryptology Conference*.

[20] Katriel, G. **2013**. Gambler's ruin probability—A general formula. *Statistics & Probability Letters* 83.10.

[21] Ozisik, A. and Levine, B. **2017**. An Explanation of Nakamoto's Analysis of Double-spend Attacks.

[22] Buterin, V. and Griffith, V. **2017**. Casper the friendly finality gadget. *arXiv preprint arXiv:1710.09437*.

[23] Gilad, Y., Hemo, R., Micali, S., Vlachos, G. and Zeldovich, N. **2017**. Algorand: Scaling Byzantine Agreements for Cryptocurrencies. In: *Proceedings of the 26th Symposium on the ACM Operating Systems Principles*. SOSP.

A Additional figures

Figure 2 has been plotted from the Bitcoin blockchain data. It shows the number of blocks that match a given block level (see Definition 1).

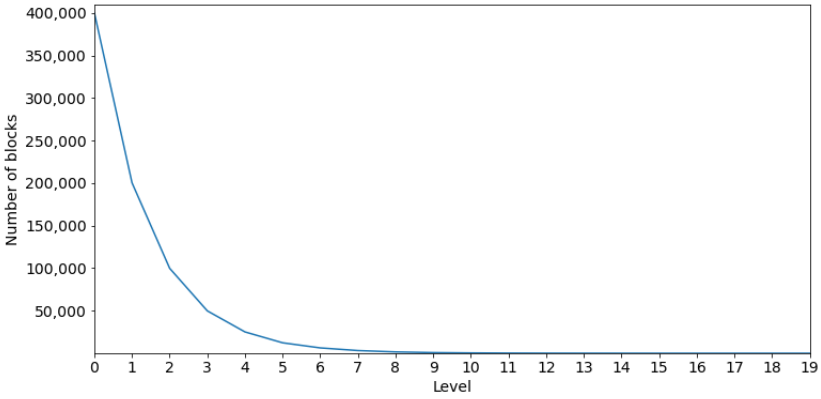
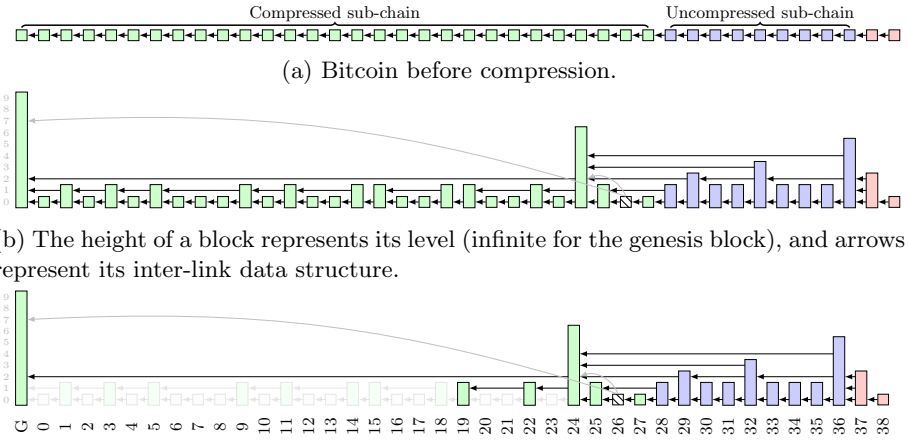


Fig. 2: Number of blocks as a function of their level (see Definition 1).

Figure 3 reflects the outcome of the compression algorithm run on the first 38 blocks of the Bitcoin blockchain. Compression parameters are $K = 2$, $\chi = 9$ and $k = 2$. Figure 3a depicts the chain according to these parameters. The unstable sub-chain (in red) gathers the last $k = 2$ blocks. The uncompressed sub-chain (in blue) gathers the following $\chi = 9$ blocks. The compressed sub-chain is depicted in green. Figure 3b depicts the very same chain according to block levels as defined in Definition 1, and their references to previous block at each level. Finally, Figure 3c show the result of the compression algorithm on this chain : Level 1 is the highest level ℓ that contains at least $2K$ blocks. Note that for readability reasons, links from each block to the genesis block have been omitted. The NIPoPoW proof is made of the sequence of blocks $= \langle G, 19, 22, 24, 25, 26, 27, 28, \ldots, 36\rangle$, where the sequence of blocks $\langle G, 19, 22, 24, 25\rangle$ represents the block sequence of level 1, and the sequence $\langle 24, 25, 26, 27\rangle$ the block sequence of level 0.



(a) Bitcoin before compression.

(b) The height of a block represents its level (infinite for the genesis block), and arrows represent its inter-link data structure.

(c) The first 38 blocks of Bitcoin after compression.

Fig. 3: Compression scheme on the first 38 blocks of Bitcoin ($K = 2$, $\chi = 9$ and $k = 2$).

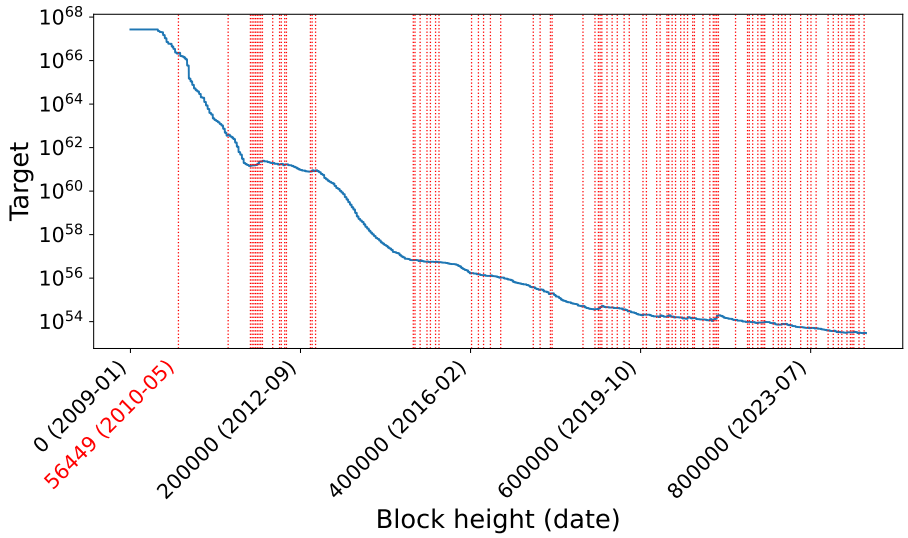


Fig. 4: Bitcoin target variation over time (865,042 blocks as of 2024-10-11). Vertical dashed lines indicate a increase of the target compared to the previous epoch, and thus a drop in difficulty. The first drop occurred in 2010.

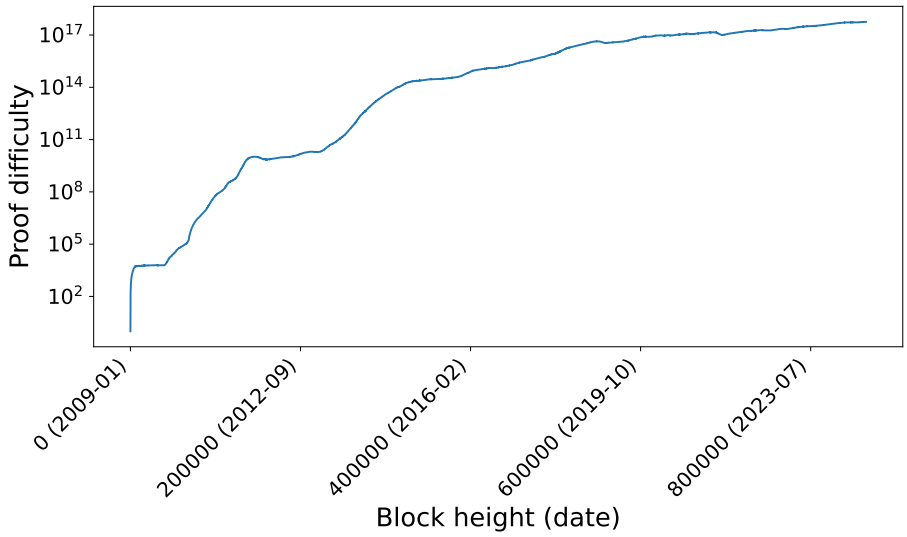


Fig. 5: Variation of the total difficulty of the proof (logscale) over time, when applied to Bitcoin (865,042 blocks as of 2024-10-11).

B Algorithms

Input : $\mathcal{C}$, which is either a regular or a compressed chain
Output: tuple $(\mathcal{D}, X, \Omega, \ell)$ where
Ω is the unstable sub-chain of $\mathcal{C}$ of size k,
X is the uncompressed sub-chain of $\mathcal{C}$ of size χ,
$\mathcal{D}$ is the compressed chain,
and ℓ is the highest level of $\mathcal{C}$

function $\texttt{Compress}_{K,\chi,k}(\mathcal{C})$:
  $\mathcal{D} \leftarrow \emptyset$
  $\Omega \leftarrow \mathcal{C}[-k:]$ // Unstable sub-chain
  $X \leftarrow \mathcal{C}[-\chi - k : -k]$ // Uncompressed sub-chain
  $\mathcal{C}^\star \leftarrow \mathcal{C}[: -\chi - k]$ // To be compressed part
  if $|\mathcal{C}^\star| \geq 2K$**:** // The chain is long enough
    $\ell \leftarrow \max\{\mu : |\mathcal{C}^\star \uparrow^\mu| \geq 2K\}$ // Get the highest level ℓ
    $\mathcal{D}[\ell] \leftarrow \mathcal{C}^\star \uparrow^\ell$ // Keep all the blocks of level $\geq \ell$
    for $\mu \leftarrow \ell - 1$ **down to** 0**:** // For each subsequent level
      $b^\star \leftarrow \mathcal{C}^\star \uparrow^{\mu+1} [-K]$ // Determine the pivot block $b^\star$
      $\mathcal{D}[\mu] \leftarrow \mathcal{C}^\star \uparrow^\mu [-2K:] \cup \mathcal{C}^\star \uparrow^\mu \{b^\star :\}$ // Keep the $2K$ most recent μ-blocks plus all the most recent μ-blocks starting from $b^\star$
  else: // The chain to be compressed is too short
    $\ell \leftarrow 0$
    $\mathcal{D}[0] \leftarrow \mathcal{C}^\star$
  return $(\mathcal{D}, X, \Omega, \ell)$

Algorithm 1: Chain compression algorithm.

```
Input  : C_1, ..., C_n: either a regular or a compressed chains
Output: tuple (D, X, Ω, ℓ) where
          Ω is the unstable sub-chain of the best chain,
          X is the uncompressed sub-chain of the best chain,
          D is the compressed chain of the best chain,
          and ℓ is the highest level of the best chain
function Compare_{K,χ,k}(C_1, ..., C_n):
    (D, X, Ω, ℓ) ← ⊥                    // Best valid history among C_0 ... C_i
    i ← 0
    while (D, X, Ω, ℓ) is ⊥ ∧ i ≤ n:
        if valid(C_i):
            (D, X, Ω, ℓ) ← Compress_{K,χ,k}(C_i)
        i ← i + 1
    while ¬((D, X, Ω, ℓ) is ⊥) ∧ i ≤ n:
        if valid(C_i):    // Compare current best valid to the next one
            (D', X', Ω', ℓ') ← Compress_{K,χ,k}(C_i)
            M ← {μ ∈ ℕ | D[μ] ∩ D'[μ] ≠ ∅}
            if M = ∅:
                (D, X, Ω, ℓ) ← ⊥                    // no block in common
            else:
                μ ← min M
                b ← (D[μ] ∩ D'[μ])[−1]    // best ≠ C_j and fork on block b
                if ‖D'{b :} · X' · Ω'‖ > ‖D{b :} · X · Ω‖:
                    (D, X, Ω, ℓ) ← (D', X', Ω', ℓ')       // updates, up to C_j
        i ← i + 1
    if (D, X, Ω, ℓ) is ⊥:
        beacon ← generate_beacon()
        send_tx_beacon (beacon)
    else:
        return (D, X, Ω, ℓ)
```

Algorithm 2: Compressed chains comparison function algorithm.

C Definitions from The Bitcoin backbone protocol with chains of variable difficulty

We consider a synchronous protocol working in rounds, with a series $n = \{n_r\}_{r\in\mathbb{N}}$ representing the total number of participants of the system, $t = \{t_r\}_{r\in\mathbb{N}}$ of which are adversarial. We have $\forall r : t_r \leq (1-\delta)(n_r - t_r)$, here with $\delta \geq 0.5$ since we consider a 1/3 adversary. Each party has ρ queries it can make to the hash function. Garay *et al.* call this the *dynamic ρ-bounded synchronous setting*. Appendix D contains a summary on variables and their use.

The analysis strongly relies on Garay *et al.*'s notion of typical executions [19]. A typical execution is an execution of the protocol where variables do not deviate too much from their expected values. A typical execution must span at least $m/16\tau f$ rounds, where m represents the number of blocks mined during an epoch, f is the probability for honest miners to mine at least one block in a round, and τ is the dampening filter in Bitcoin target's recalculation function. A round r is successful is the honest parties succeed in mining at least one block during r, and is uniquely successful if the honest parties succeed in mining exactly one block during r. The quantity Q_r is equal to the difficulty of the block mined during a uniquely successful round. If the honest parties mine several blocks or no blocks during round r then $Q_r = 0$. $\sum_{r\in S} Q_r(E)$ refers to the difficulty obtained during the uniquely successful rounds of a set of S rounds. Likewise, $\sum_{r\in S} D_r(E)$ refers to the difficulty obtained during the successful rounds of a set of S rounds. Quantity $\sum_{j\in J} A_j(E)$ represents the amount of difficulty obtained by the adversarial parties over a set of queries J. Considering a set of queries rather than a set of rounds allows the adversary to specifically target rounds during which it queries the hash function.

The Bitcoin Backbone model [19] also provides us with some properties that an execution can have. Intuitively, a (γ, s)-respecting sequence limits the variation of the number of participants to a factor of γ during s rounds. For example, a $(2, 5)$-respecting sequence means the number of participants can at most double every 5 rounds. We need to limit the variation of the number of parties because our properties would not hold for arbitrary sequence of parties.

Definition 2 ([19], Definition 1 - (γ, s)-respecting sequence). *For $\gamma \in \mathbb{R}^+$, we call a sequence $(n_r)_{r\in\mathbb{N}}$ (γ, s)-respecting if for any set S of at most s consecutive rounds, $max_{r\in S}\ n_r \leq \gamma \cdot min_{r\in S}\ n_r$.*

Note that any sequence of parties can be (γ, s)-respecting, but when we say in the following to consider a (γ, s)-respecting environment, we specifically refer to the sequence of honest parties $(n_r - t_r)_{r\in\mathbb{N}}$.

We also define (η, θ)-good executions, which are intuitively a limit on the amount of "luck" parties have when mining a block. Initial parameters of Bitcoin are set so that one block is mined every 10 minutes on average, which is modeled as the block production rate f. An (η, θ)-good execution gives an idea on how well parties approximate f, as it gives upper and lower bounds for f.

Definition 3 ([19], Definition 5 - (η,θ)-good execution). *Consider an execution E and constants $\eta \in (0,1]$ and $\theta \in [1,\infty)$. A target-recalculation point r in a chain C in E is (η,θ)-good if the new target T satisfies $\eta f \leq f(T, n_r - t_r) \leq \theta f$. A chain C in E is (η,θ)-good if all its target-recalculation points are (η,θ)-good. A round r is (η,θ)-good in E if $\eta f \leq f(T_r^{min}, n_r - t_r)$ and $f(T_r^{max}, n_r - t_r) \leq \theta f$. We say that E is (η,θ)-good if every round in E was (η,θ)-good.*

We define $T^{(r,\eta)}$ as the minimum target for all honest parties in a (η,θ)-good round. $T^{(S,\eta)}$ is then the minimum target for all honest parties in all rounds $r \in S$. We also define $\phi = \rho/2^\kappa$ for convenience.

Definition 4 ([19], Definition 8 - Typical execution). *An execution E is $(\varepsilon,\eta,\theta)$-typical if the following hold:*

(a) If, for any set S of consecutive rounds, $\phi T^{(S,\eta)} \sum_{r\in S}(n_r - t_r) \geq \frac{\eta m}{16\tau\gamma}$, then

$$\begin{aligned} &\sum_{r\in S} Q_r(E) > \sum_{r\in S} \mathbf{E}[Q_r | \varepsilon_{r-1} = E_{r-1}] - \varepsilon(1-\theta f)\phi \sum_{r\in S}(n_r - t_r) \\ \text{and } &\sum_{r\in S} D_r(E) < (1+\varepsilon)\phi \sum_{r\in S}(n_r - t_r) \end{aligned} \tag{3}$$

(b) For any set J indexing a set of consecutive queries of the adversary we have

$$\sum_{j\in J} A_j(E) < (1+\varepsilon)2^{-\kappa}|J| \tag{4}$$

and the blocks with targets less than $\tau T^{(J)}$ that the adversary acquired are less than $\frac{\eta(1-\varepsilon)(1-\theta)f}{32\tau^2\gamma} \cdot m$

(c) No insertions, no copies, and no predictions occurred in E.

Remark 1 ([19], Remark 4). Note that if J indexes the queries of the adversary in a set S of consecutive rounds, then $|J| = \rho \sum_{r\in S} t_r$ and the inequality in Definition 4(b) reads $\sum_{j\in J} A_j(E) < (1+\varepsilon)\phi \sum_{r\in S} t_r$.

Proposition 1 ([19], Proposition 3). *Assume E is a typical execution in a (γ, s)-respecting environment. For any set S of consecutive rounds with $|S| \geq \frac{m}{16\tau f}$,*

$$\sum_{r\in S} D_r(E) < (1+\varepsilon)\phi \sum_{r\in S}(n_r - t_r) \tag{5}$$

If in addition, E is (η,θ)-good, then

$$\sum_{r\in S} Q_r > (1-\varepsilon)(1-\theta f)\phi \sum_{r\in S}(n_r - t_r) \tag{6}$$

and any block computed by an honest party at any round r corresponds to target at least $T^{(r,\eta)}$, and so contributes to the random variables D_r and Q_r (if the r was uniquely successful).

Definition 5 (Q-block [9], Definition 1). *A block property is a predicate Q defined on a hash output $h \in \{0,1\}^\kappa$. Given a block property Q, a valid block with hash h is called a Q-block if $Q(h)$ is true.*

Using those definitions, we now prove the security of our protocol. Our proof closely follows that of Kiayias *et al.*'s proof of security, but is adapted to the variable difficulty setting.

D Variables of interest

This appendix gives the reader a table summarizing variables used in the paper, their domain of definition and their meaning.

Table 2: Summary of used variables, their domain of definition and their meaning

	Domain of definition	Description
n_r	$\forall r : n_r \in \mathbb{N}$	Number of total parties in round r.
t_r	$\forall r : t_r \in \mathbb{N}$	Number of adversarial parties in round r.
ρ	$\rho \in \mathbb{N}$	Number of queries of each party.
ϕ	$\phi \in \mathbb{Q}_{\geq 0}$	Convenience notation. $\phi = \rho/2^\kappa$.
δ	$\delta \in (0, 1)$	Advantage of honest parties, $\forall r : t_r \leq (1 - \delta)(n_r - t_r)$.
m	$m \in \mathbb{N}$	Length of an epoch in blocks, $m = 2016$ for Bitcoin.
f	$f \in (0, 1)$	Target probability of at least one honest party mining a block in a round.
τ	$\tau \in \mathbb{R}$	Dampening filter, $\tau = 4$ for Bitcoin.
(γ, s)	$\gamma \in \mathbb{R}, s \in \mathbb{N}$	Bound on variation of the number of parties. See Definition 2.
(η, θ)	$\eta \in (0, 1], \theta \in [1, \infty)$	Lower and upper bound on f. See Definition 3.
ε	$\varepsilon \in (0, 1)$	Quality of concentration of random variables in typical executions. See Definition 4.
Q_r	$Q_r \in \mathbb{R}$	Difficulty of honest block mined in uniquely successful round r.
D_r	$D_r \in \mathbb{R}$	Maximum difficulty among honest blocks mined in round r.
A_j	$A_j \in \mathbb{R}$	Difficulty of adversarial block mined in query j.
κ	$\kappa \in \mathbb{N}$	Size parameter of the output of the hash function, found as 2^κ. $\kappa = 256$ for Bitcoin.
k	$k \in \mathbb{N}$	Common Prefix parameter. Length of the unstable sub-chain of the proof.
χ	$\chi \in \mathbb{N}$	Length of the uncompressed sub-chain of the proof.
K	$K \in \mathbb{N}$	Security parameter of our protocol.
ℓ	$\ell \in \mathbb{N}$	Level of a block or a proof.
α	$\alpha \in [\frac{1}{4}, 4]$	Ratio between the difficulty reward of the honest parties a and the adversary b. $\alpha = a/b$.
p	$p \in [0, 1]$	Fraction of honest computing power. $p + q = 1$, and $p = \frac{2}{3}$ in our setting.
q	$q \in [0, 1]$	Fraction of adversarial computing power. $p + q = 1$, and $q = \frac{1}{3}$ in our setting.

E Proofs

Proofs for the technical lemmas of Section 6.

Lemma 5 (Pairing). *Consider a execution with an honest party, an adversary and two chains $\mathcal{C}$,$\mathcal{C}'$. For any pair of distinct blocks $\mathcal{C}[i]$ and $\mathcal{C}'[i']$ such that $\exists d \in \mathbb{R}^+, d \in \mathcal{C}[i]$ and $d \in \mathcal{C}'[i']$, if $\mathcal{C}[i]$ was computed by an honest party in a uniquely successful round, then $\mathcal{C}'[i']$ was computed by the adversary.*

Proof. Let r be the uniquely successful round where $\mathcal{C}[i]$ was computed, and $d \in \mathcal{C}[i]$ and $d \in \mathcal{C}'[i']$. We show that no honest party would have computed $\mathcal{C}'[i']$. Due to the synchronous setting, every block computed in a round is received during that round. Consider $r' > r$. Since parties are aware of $\mathcal{C}[i]$ and by definition of d, $\|\mathcal{C}'[:i']\| \leq d < \|\mathcal{C}[:i+1]\|$, no honest party would have computed $\mathcal{C}'[i']$ at r'. Similarly, if an honest party computed $\mathcal{C}'[i']$ at some round $r' < r$, then no honest party would have extended $\mathcal{C}[i-1]$ at round r. Since r is uniquely successful, $\mathcal{C}'[i']$ cannot have been computed by an honest party at round r. □

Lemma 6 (Suppression). *If r is a uniquely successful round and the corresponding block b does not belong to the chain of an honest party at a later round, then there is a set of consecutive rounds S and a set J of adversarial queries in S such that $r \in S$ and $\sum_{r \in S} Q_r \leq \sum_{j \in J} A_j$.*

Proof. Let us consider an execution in which an honest party maintains $\mathcal{C}$, and let $b \in \mathcal{C}$ a uniquely successful block mined at round r. Let $\mathcal{C}[u_0] = \mathcal{C}'[u_0]$ be the last honest block on the common prefix of $\mathcal{C}$ and $\mathcal{C}'$, and r_0 be the round at which the honest party created $C[u_0]$ (if $C[u_0]$ is the genesis block, then we $r_0 = 0$). Consider $r_1 > r$ the first round after r in which an honest party adopts $\mathcal{C}'$ such that $b \notin \mathcal{C}'$. We build a sequence S of consecutive rounds $S = \{r' : r_0 < r' < r_1\}$ and we show that the set of adversarial queries J exhibit more difficulty than the ones of the honest party during S. From Lemma 1, we have that any uniquely successful contribution on $\mathcal{C}$ (resp. $\mathcal{C}'$) of the honest parties between r_0+1 and r_1-1 is matched by an equal adversarial contribution on $\mathcal{C}'$ (resp. $\mathcal{C}$). If the adversary did not match difficulty contributions between r and $r_1 - 1$, an honest party mining on $\mathcal{C}$ would not have adopted $\mathcal{C}'$ at r_1, which contradicts the initial assumption. Likewise, if the adversary did not match difficulty contributions between r_0+1 and $r-1$, block b would have been included in $\mathcal{C}'$. Thus, $\sum_{r \in S} Q_r \leq \sum_{j \in J} A_j$.

Lemma 7 (Unsuppressibility). *Given a typical execution with an honest party and an adversary, every set of consecutive rounds U has a subset S of uniquely successful rounds, such that the following conditions hold:*

1. $\sum_{r \in S} Q_r \geq \sum_{r \in U} Q_r - 2\sum_{j \in U} A_j - (1+\varepsilon)\phi \sum_{r \in 2\frac{m}{16\tau f}} t_r$,
2. *after the last round in S the blocks corresponding to S belong to the chain of every honest party.*

Proof. Let U' be the set of consecutive rounds that contains U and also the $\frac{m}{16\tau f}$ rounds that come before and after U. By Lemma 2, we may take S to contain all those uniquely successful rounds $r \in U$ such that for any set of consecutive rounds $S' \subseteq U'$ containing r, $\sum_{r\in S} Q_r > \sum_{j\in J} A_j$. Note that, in a (η, θ)-good typical execution in a (γ, s)-respecting environment, no such S' may contain elements outside U'.

We need to prove this statement: $\sum_{r\in U} Q_r - \sum_{r\in S} Q_r \leq 2\sum_{j\in U} A_j + (1 + \varepsilon)\phi \sum_{r\in 2\frac{m}{16\tau f}} t_r$. Let us focus on the uniquely successful rounds not in S. Consider a collection $\mathcal{T}$ of sets of consecutive rounds with the following properties.

- $\forall T \in \mathcal{T}, \sum_{r\in T} Q_r \leq \sum_{j\in T} A_j$.
- $\forall r \in U \setminus S$, there is a $T \in \mathcal{T}$ that contains r.
- $|\mathcal{T}|$ is minimum among all collections with the above properties.

We now observe that the minimality condition on $\mathcal{T}$ implies that no round r with $A_r > 0$ belongs to more than two sets of $\mathcal{T}$. If that was the case, then there would be three sets T_1, T_2, T_3 in $\mathcal{T}$ with $T_1 \cap T_2 \cap T_3 \neq \emptyset$. But then, we could keep the two sets with the leftmost and rightmost endpoints, contradicting the minimality of $\mathcal{T}$. No round in $U' \setminus U$ belongs to more than one set of $\mathcal{T}$. Thus,

$$\sum_{r\in U} Q_r - \sum_{r\in S} Q_r = \sum_{r\in U\setminus S} Q_r \leq \sum_{T\in\mathcal{T}} \sum_{r\in T} Q_r \leq \sum_{T\in\mathcal{T}} \sum_{j\in T} A_j \leq 2\sum_{j\in U} A_j + \sum_{j\in U'\setminus U} A_j$$

The third inequality holds because every round in which the adversary was successful is counted at most twice inside U and at most once outside U (by the discussion above the inequalities). Finally, using $|U' \setminus U| \leq 2m/16\tau f$ and $\sum_{j\in J} A_j < (1 + \varepsilon)\phi \sum_{r\in S} t_r$ we obtain the stated bound. □

F Dimensioning parameter K

All the properties of our NIPoPoW instantiation, namely Security (Theorem 1), Succinctness (Theorem 2) and Onlineness (Theorem 3) rely on security parameters K, χ and k. We dimension k and χ in Section 5.3. Security parameter K must guarantee that with any high probability a 1/3-adversary cannot exhibit more difficulty in its own K sampled blocks than what the honest parties can do. This ensures that the $\texttt{Compare}_{K,\chi,k}()$ algorithm, when fed with a set of NIPoPoWs such that among them at least one is an honestly generated one, will return the best NIPoPoW, i.e. the NIPoPoW that represents the blockchain with the largest accumulated difficulty.

We model the protocol as a competition between an honest party and the adversary. We denote by Π the honest NIPoPoW and by Π' the adversarial one. We assume that the competition takes place within an epoch which means that the honest target T is constant during the competition. The competition starts at the first time where both chains fork. We denote by $b^\star$ the last common block: $b = (\Pi \cap \Pi')[-1]$. At each step of the competition, a unique block is created, either by the honest party or the adversary. We assume that the adversary may potentially mine blocks at a different difficulty from that of the honest parties. We denote by $\alpha = a/b$ the ratio between honest and adversarial blocks in terms of difficulty.

We refer by n the number of blocks in the honest chain that do not belong to the adversarial chain, we have $n = |\Pi\{b^\star :\}|$. We are interested in determining the earliest step in which the adversarial chain overcomes the honest one, after n blocks have been appended to the honest chain. We denote this event by C_n^α.

To derive our formula for $\mathbb{P}(C_n^\alpha)$ we will use the classical Gambler's ruin problem [20] and the Poisson distribution. The classical Gambler's ruin provides the probability, denoted $P_{ruin}(M)$, for the adversary to catch up with the honest chain given an initial gap M between his chain and the honest one. By combining this with the Poisson distribution, we will ultimately obtain $\mathbb{P}(C_n^\alpha)$.

F.1 Gambler's ruin application

We consider the classical Gambler's ruin problem [20]. We model the evolution of the gambler's payoff along games by an homogeneous discrete-time Markov chain $X = \{X_k, k \geq 0\}$, where $M \in \mathbb{N}$ is the initial wealth of the gambler ($X_0 = M$) with

$$\mathbb{P}(X_k = j) = p_j, \quad -\nu \leq j < \infty$$

where ν is the maximal possible loss with the assumption that $p_{-\nu} \neq 0$. The gambler must stop playing when his wealth is less than ν, in which case he is ruined. We want to find $P_{ruin}(M)$, the probability of ruin which depends on the payoff distribution $\{p_j\}_{-\nu \leq 0 < \infty}$ and on the initial wealth M. We know that $P_{ruin}(M) = 1$ if the expected value of X is non positive, so we assume:

$$\mathbb{E}(X) = \sum_{k=-\nu}^{\infty} kp_k > 0 \tag{7}$$

Another assumption made in [20] is that $M \geq \nu$, otherwise the gambler is already ruined and $\forall M < \nu, P_{ruin}(M) = 1$. We also need to define the generating function:

$$p(z) = \sum_{k=-\nu}^{\infty} p_k z^k \tag{8}$$

We can now draw an analogy between the blockchain and the Gambler's ruin problem in the following way. First, we take M as an initial gap between the honest chain and the adversary's chain in terms of quantity of work. Recall that by assumption, these two proofs share a least one block and are thus comparable. Recall also that we are only looking at the cases when exactly one block is created, either by the adversary or by the honest party. Thus, we have at each iteration k, $X_k = a$ with $P(X_k = a) = p_a$ and the gap increases by a, or $X_k = -b$ with $P(X_k = -b) = p_{-b}$ and the gap is reduced by b. We now have that $\nu = b$ and Relation (8) becomes

$$p(z) = p_a z^a + p_{-b} z^{-b} \tag{9}$$

Let p be the mining power of the honest and q the mining power of the adversary such that $p + q = 1$. When the adversary mines with the target αT, we define p_a and p_{-b} to be the probabilities that the next block is mined by the adversary and the honest miners, respectively. We consider a time interval during which z blocks mined with target T (i.e., when $\alpha = 1$) are created. Among the z blocks, qz of them have been created by the adversary and pz one by the honest party. Now if the adversary mines with target αT, the adversary will mine αqz blocks during the very same interval so that the total number of blocks during the time interval will no longer be z but $\alpha qz + pz$ with αqz being the number of blocks belonging to the adversary and pz to the honest party. Finally, finding the expressions for p_a and p_{-b} comes down to determining the proportion of blocks belonging to the adversary and to the honest party among the $\alpha qz + pz$ blocks mined in total during the considered time interval.
We obtain:

$$p_a = \frac{\alpha qz}{pz + \alpha qz} = \frac{q\alpha}{p + q\alpha} \qquad p_{-b} = \frac{pz}{pz + \alpha qz} = \frac{p}{p + q\alpha}$$

For integers $n > 0$, $r \geq 0$, the complete symmetric polynomial of order r in the variables $z_1, \ldots, z_n$ is defined as the sum of all products of the variables $z_1, \ldots, z_n$ of degree r, that is:

$$\Phi_{n,r}(z_1, \ldots, z_n) = \sum_{\substack{i_j \geq 0, \\ i_1 + \cdots + i_n = r}} \prod_{j=1}^{n} z_j^{i_j}. \tag{10}$$

We can now give Theorem 4 from [20] which provides a formula for P_{ruin} the probability of ruin. Since M is in our case the initial gap between the two chains, $P_{ruin}(M)$ corresponds to the probability for the adversary to catch up with the honest chain given an initial gap M.

Theorem 4. *The equation $p(z) = 1$ has ν solutions (counting multiplicities) in the unit disk $|z| < 1$ of the complex plane, which we denote as η_j (for $1 \leq j \leq \nu$). The probability of ruin is given by*

$$P_{ruin}(M) = \sum_{n=1}^{\nu} \Phi_{n,M-n+1}(\eta_1, \ldots, \eta_n) \prod_{j=1}^{n-1}(1 - \eta_j).$$

When the roots $\eta_1, \ldots, \eta_\nu$ are distinct, we can use the following alternative expression:

$$P_{ruin}(M) = \sum_{j=1}^{\nu} \eta_j^M \prod_{\substack{i=1 \\ i \neq j}}^{\nu} \frac{1 - \eta_i}{\eta_j - \eta_i}.$$

F.2 Combining Gambler's ruin problem with the Poisson distribution

Inspired by [21] we derive the general formula for $\mathbb{P}(C_n^\alpha)$ combining the gambler's ruin problem with the Poisson distribution. The gap we use for P_{ruin} is $(na - ib)$ for $i \geq 0$ which corresponds to all the possible gaps between the two chains.

Let X be a random variable representing the number of blocks produced by the adversary, and let λ denote the Poisson parameter, which corresponds to the average number of blocks produced by the adversary during the time interval I_n, where I_n is the period in which the honest party produces n blocks. The Poisson distribution gives us $\mathbb{P}(\{X = i; \lambda\})$, the probability that the adversary produces i blocks during I_n. Since $\{\{X = i; \lambda\}; i \geq 0\}$ is a complete system of events with $\mathbb{P}(\{X = i; \lambda\}) \neq 0$ and $P_{ruin}(na - ib) = \mathbb{P}(C_n^\alpha \mid \{X = i; \lambda\})$, we can determine $\mathbb{P}(C_n^\alpha)$ using the law of total probability:

$$\begin{aligned}
\mathbb{P}(C_n^\alpha) &= \sum_{i=0}^{\infty} \mathbb{P}(X = i; \lambda) \cdot \mathbb{P}(C \mid \{X = i; \lambda\}) = \sum_{i=0}^{\infty} \mathbb{P}(X = i; \lambda) \cdot P_{ruin}(na - ib) \\
&= \sum_{i=0}^{\lfloor n\alpha - 1 \rfloor} \frac{\lambda^i e^{-\lambda}}{i!} P_{ruin}(na - ib) + \sum_{k=\lceil n\alpha - 1 \rceil}^{\infty} \frac{\lambda^k e^{-\lambda}}{i!} \\
&= 1 - (1 - \sum_{i=0}^{\lfloor n\alpha - 1 \rfloor} \frac{\lambda^i e^{-\lambda}}{i!} P_{ruin}(na - ib) - (e^{-\lambda} \sum_{i=0}^{\infty} \frac{\lambda^i}{i!} - \sum_{i=0}^{\lfloor n\alpha - 1 \rfloor} \frac{\lambda^i e^{-\lambda}}{i!})) \\
&= 1 - \sum_{i=0}^{\lfloor n\alpha - 1 \rfloor} \frac{\lambda^k e^{-\lambda}}{i!} (1 - (P_{ruin}(na - ib))
\end{aligned}$$