Preuves non-interactives : la nouvelle ère des chaînes compressées

Dorian Pacaud, Loïc Miller, Emmanuelle Anceaume, Romaric Ludinard

Preuves non-interactives : la nouvelle ère des chaînes compressées

Dorian Pacaud[1] et Loïc Miller[1] et Emmanuelle Anceaume[2] et Romaric Ludinard[1]

[1]*IMT Atlantique / IRISA* [2]*CNRS / IRISA*

Un des défis majeurs de la blockchain réside dans la gestion de la complexité liée à la communication et au stockage. Pour garantir la sécurité de la blockchain, il est nécessaire de conserver intégralement les données de consensus, dont la taille augmente linéairement avec la taille de la chaîne, ce qui compromet la pérennité de la blockchain. Les Preuves Non-Interactives de Preuve de Travail (NIPoPoWs) apportent une solution à ce problème, à condition que le système conserve un nombre constant de participants. Nous proposons pour la première fois une construction qui répond rigoureusement aux exigences d'une NIPoPoW, capable de résister à un adversaire contrôlant jusqu'à un tiers des ressources dans un environnement dynamique. Nous montrons la concision, la sécurité et l'actualisabilité de ce système, tandis que nos résultats expérimentaux confirment une réduction exponentielle de la taille de la blockchain Bitcoin.

Mots-clefs : blockchain, preuve de travail, compression, difficulté variable

1 Introduction

In its current form, blockchain technology does not scale. One aspect of scalability difficulty comes from the data required to be stored and sent over the network when nodes need to synchronize with each other. Blockchains maintain two types of data: the application data and the consensus data, both included in each block. The first one includes everything that is part of the block data itself, such as transactions or account balances. Consensus data includes the cryptographic fingerprints, the nonce, and the block mining difficulty. In this work, we focus on Proof-of-Work (PoW) chains [Nak08] and consensus data. Consensus data grows linearly over time, hindering the long-term feasibility and thus adoption of blockchain technology. While there exist several works that permit the compression of application data [SNA20, PD16], they are not applicable for the consensus data. To address the scalability issue of consensus data, a new construction named NIPoPoW for Non-Interactive Proof of Proof-of-Work has been recently proposed [KLZ21, JAG23]. Based on the probabilistic properties of the hash function and the definition of block levels, this construction enables the sampling of well-chosen blocks to create a compressed chain with a polylogarithmic length relative to the entire chain. An honest prover can then securely help a new node to bootstrap by sending only this polylogarithmic sample. However, this construction is proven secure under the assumption that all blocks have been mined with the same mining difficulty.

Based on this construction, we propose a NIPoPoW compliant with variable difficulty [MPD+24]. Our construction compresses a PoW blockchain to a polylogarithmic number of blocks. This compressed proof is sufficient for miners to mine new blocks and for bootstrapping parties to synchronize in a non interactive way. We prove correctness of our construction against a Byzantine adversary owning at any time strictly less than $1/3$ of the system's computational power. We illustrate the performance of our construction by compressing the Bitcoin blockchain.

2 Model of the system and properties

We consider PoW blockchains and make the assumption that each block contains a snapshot of the application state. We adopt a synchronous setting with discrete rounds during which each party can send and receive messages from other parties, and execute computational steps based on the received messages. We suppose the presence of an adversary who is limited to a probabilistic polynomial-time Turing machine that behaves arbitrarily. The adversary may not follow the protocol, but has bounded computational power, i.e., at any time the adversary owns less than $1/3$ of the total hashing power. This is a $1/3$-bounded PPT adversary. Any party following the prescribed protocol is called an honest party.

The NIPoPoW primitive we propose consists of two operations: Given a chain C, $Compress(C)$ produces a proof Π, and $Compare(C_1, \ldots, C_n)$ outputs the proof C_i which has accumulated the most work from $C_1, \ldots, C_n$. Let $|C|$ be the length of the chain C (that is, the number of blocks in C) and $C \cdot b$ (resp. $\Pi \cdot b$) be the chain C (resp. Π) to which is appended a valid block b. The correctness of the NIPoPoW primitive in the presence of a $1/3$-bounded PPT adversary must guarantee the following properties.

Theorem 2.1 (Security) *For any set of proofs $\Pi_1, \ldots, \Pi_n$ such that among them at least one has been provided by an honest party, $Compare(\Pi_1, \ldots, \Pi_n)$ returns Π_i, representative of the honest blockchain that has accumulated the most work.*

Theorem 2.2 (Succinctness) *For any blockchain C of an honest party, $|Compress(C)| = O(polylog|C|)$.*

Theorem 2.3 (Onlineness) *For any blockchain C of an honest party, $Compress(C \cdot b) = Compress(\Pi \cdot b)$,*

Proposition 2.3 states that if a party knows only the proof Π of an underlying chain C, this party can mine directly on top of Π. It ensures scalability as parties no longer need to store the underlying chain.

3 NIPoPoW in a variable setting

The number of participants in the system is not constant, so miners have to recalculate the mining target such that the inter-block delay remains reasonable. A long inter-block delay reduces transaction throughput, whereas a short inter-block delay will hurt liveness because of the increased number of forks. The recalculation of the mining target occurs at the end of each epoch, i.e., every 2016 blocks for Bitcoin.

Definition 1 (Block mining difficulty) *For any valid block b, i.e., such that $h(b) < T$ where h is the hash function and T the current mining target of b, the mining difficulty of block b is $\frac{1}{T}$.*

The intuition of our construction is based on the definition of block level introduced by Kiayias *et al.* [KLZ21]. At epoch i, when a miner finds a nonce such that $h(b) < T_i$ where T_i is the mining target of epoch i, the nonce could also by chance verify $h(b) < T_i/2^\ell$ for some integer ℓ.

Definition 2 (Level of a block) *Let C be a blockchain, and block $b \in C$ such that b was mined during epoch i. We say that the level of block b is equal to ℓ if it verifies*

$$\frac{h(b)}{T_i} \leq \frac{1}{2^\ell}$$

We say that a block of level ℓ is an ℓ-block. Note that by definition, a ℓ-block is also a ℓ'-block for all $0 \leq \ell' < \ell$. The probability that a valid block is of level ℓ is $1/2^\ell$. To obtain an ℓ-block, 2^ℓ blocks must have been mined on average around this ℓ-block.

Construction of a NIPoPoW. NIPoPoWs are composed of two operations. The $Compress(C)$ operation builds the NIPoPoW Π of C, while the $Compare(\Pi_1, \ldots, \Pi_n)$ operation returns the proof Π_i that has accumulated the most work among $\Pi_1, \ldots, \Pi_n$. The former is executed by a prover, while the latter is executed by a verifier. As detailed below, the $Compress(C)$ operation sub-samples blocks, which in a variable difficulty setting, prevents block difficulties from being verified. Therefore this paves the way to adversarial strategies, in particular, to low-difficulty attacks. A low-difficulty attack consists for an adversarial prover in secretly mining a blockchain whose “old blocks” have difficulties much lower than old honest blocks (their mining is therefore very fast, which allows the prover to illegitimately get high level blocks), and recent blocks have a difficulty similar to current honest blocks (to guarantee that the suffix of the adversarial proof will appear legitimate compared to an honest proof).

Compression. The $Compress()$ algorithm samples a logarithmic number of well-chosen blocks from the chain. Specifically, the algorithm first computes the maximum level ℓ of the blockchain, i.e., the maximum level at which the blockchain contains at least $2K$ blocks of level ℓ, with K the security sampling parameter. Next, the algorithm keeps the $2K$ last blocks of each level μ below ℓ down to level 0. The algorithm also keeps all the μ-blocks after the K^{th} block of level $\mu + 1$. The K-th most recent $\mu + 1$-block b^{μ}_{C} is called the μ-pivot block of C. The NIPoPoW Π will therefore be composed of all the most recent blocks of the blockchain and of very few "old" blocks. Our construction requires that each block header keep pointers to the last preceding block of every level to ensure that a NIPoPoW also forms a chain, i.e., a totally ordered sequence of blocks. Each block header contains no more than $\log |C|$ pointers (see Figure 1).

Comparison. When a newcomer or a not fully synchronized entity v wishes to synchronize with the most recent and honestly computed NIPoPoW, v first sends a $Sync$ request with a private random number v of its choice. Each miner u, upon receiving $Sync$ requests, inserts all private random numbers the miner has not already inserted into the block b under creation. Once created, u broadcasts b, appends b to u's local NIPoPoW Π, compresses $\Pi \cdot b$ to get an up-to-date NIPoPoW Π', and broadcasts Π'. Newcomer v will accept NIPoPoW Π' only if Π' contains v's own random number. Such a NIPoPoW is called legitimate, and the block that contains v's random number is called v's legitimate block. After receiving legitimate NIPoPoWs $\Pi_1, \ldots, \Pi_n$, v executes $Compare(\Pi_1, \ldots, \Pi_n)$ operation to get the most recent and honest NIPoPoW among $\Pi_1, \ldots, \Pi_n$. Note that we assume that among all the legitimate received NIPoPoWs, at least one has been created by a honest prover. Specifically, the $Compare()$ operation compares legitimate NIPoPoWs Π_i and Π_j pairwise and iteratively, keeping the best NIPoPoW after each pairwise comparison. To compare Π_i and Π_j, the algorithm first determines the smallest level μ at which both Π_i and Π_j share a common block b. We call b the Last Common Ancestor (LCA). In the case where there exists an LCA, it means that both underlying chains of Π_i and Π_j are close in number of blocks. By construction, either Π_i or Π_j has at least K μ-blocks after the LCA block b. Thus, the algorithm simply keeps the NIPoPoW that has accumulated the most work in its sampled μ-blocks, defending against low difficulty attacks. The security proof shows that this is sufficient to obtain the best honest NIPoPoW. This comes from the fact that the adversary cannot produce K blocks worth of difficulty faster than the honest parties. On the other hand, the absence of LCA between Π_i and Π_j characterizes a length difference between both underlying blockchains, which may be representative of a brute force low-difficulty attack, in which all the blocks of the adversarial blockhain have been mined with a low difficulty. To detect such an attack, newcomer v waits for the K blocks $b_{i_1} \ldots b_{i_K}$ that are appended to v's legitimate block b_i of Π_i (resp. the K blocks $b_{j_1} \ldots b_{j_K}$ that are appended to v's legitimate block b_j of Π_j). If v receives $b_{i_1} \ldots b_{i_K}$ before $b_{j_1} \ldots b_{j_K}$, then Π_i is taken for the best NIPoPoW between Π_i and Π_j, and Π_j otherwise.

4 Analysis and results

Relying on Garay *et al.*'s model [GKL17], we prove that our solution satisfies Theorems 2.1, 2.2 and 2.3. Let C (resp. C') be a honest (resp. adversarial) PoW-blockchain.

For any $\mu \in \mathbb{N}$, we denote by $D_H(\mu)$ (resp. $D_A(\mu)$) the accumulated difficulty of all μ-blocks in subchain $C \setminus (C \cap C')$ (resp. $C' \setminus (C \cap C')$).

Proposition 1 (μ-block common prefix) *For any $\mu \in \mathbb{N}$, there exists a certain quantity D^{μ}_{A} on subchain $C' \setminus (C \cap C')$ such that if $D_A(\mu) \geq D^{\mu}_{A}$, then with any high probability, blockchain C has accumulated more difficulty than C' at level μ i.e. $D_H(\mu) > D_A(\mu)$.*

The quantity D^{μ}_{A} decays exponentially with μ [MPD+24].

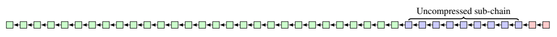
Dimensioning the security parameter K. We model our protocol as a competition between the honest party and an adversary [Kat13, OL17]. We study the evolution of both chains from the forking block and calculate the probability p_{K_A} that $D_A(0) > D_H(0)$ after the adversary has provided K_A blocks and p_{K_H} the probability that $D_A(0) > D_H(0)$ after the honest party has provided K_H blocks. For $p = max(p_{K_A}, p_{K_H}) < \epsilon$, where $\epsilon = 10^{-6}$, we obtain $K = max(K_A, K_H) = 208$. It means that when any party has mined K blocks after the forking block, then $D_H(0) > D_A(0)$ with probability $1 - \epsilon$.

Security. Selecting the heaviest NIPoPoW needs to consider three cases. First, if the two NIPoPoWs do not share an LCA, the verifier uses the block that includes his private random number as a beacon and waits for K blocks to be appended. By the dimensioning of K, the verifier chooses the honest NIPoPoW. Second, if there exists an LCA between both NIPoPoWs and the LCA is at level $\mu = 0$, then the classic common prefix property is applied. Finally, if there exists an LCA between both NIPoPoWs and the LCA is at level $\mu > 0$, then the honest NIPoPoW is selected. Indeed, by combining Proposition 1 with the dimensioning of K and the fact that D_A^{μ} decays exponentially with μ, we obtain with high probability that $D_H(\mu) > D_A(\mu)$.

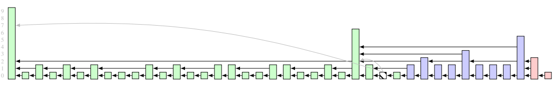
Succinctness. For any blockchain C, $|C|/2^{\ell}$ blocks are ℓ-blocks on average, with $\ell = 0, \dots, \log|C|$. There are therefore approximately $\log(|C|)$ different levels. By construction, half of ℓ-blocks are also $(\ell + 1)$-blocks, thus there are strictly less than $4K$ blocks after the ℓ-pivot block b_C^{ℓ} for each level ℓ, $\ell = 0, \dots, \log|C|$. Hence approximately $2K$ ℓ-blocks are sampled for each level ℓ (never less than $2K$ and always strictly less than $4K$). Consequently, a NIPoPoW contains $O(2K. \log(|C|))$ blocks. In addition, each ℓ-block b, $\ell = 0, \dots, \log|C|$, points to no more than $\log(|C|)$ blocks (each one being the last μ-block that precedes b, with $\mu = 0, \dots, \log|C|$. This gives $|Compress(C)| = O(polylog|C|)$.

Onlineness. The $Compress()$ operation applied on a chain C samples the $2K$ most recent ℓ-blocks, with $\ell = 0, \dots, \log|C|$. The operation determines for each level ν a pivot block b_C^{ν} which is the K^{th} most recent block of C. Since the K most recent blocks are the same for both $C.b$ and $\Pi.b$, we have $b_{C\cdot b}^{\nu} = b_{\Pi\cdot b}^{\nu}$. In addition, all the blocks in $\Pi \cdot b$ are included in $C \cdot b$, thus $Compress(C \cdot b) = Compress(\Pi \cdot b)$.

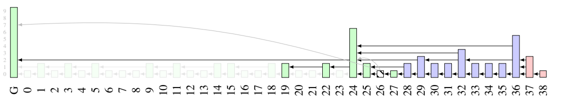
We experimentally show (see Figure 2) that when $|C| \to \infty$, $2K\log(|C|) \leq NIPoPoW(|C|) \leq 3K\log(|C|)$, where $NIPoPoW(|C|)$ represents the number of ℓ-blocks of the NIPoPoW, with $\ell = 0, \dots, \log|C|$.



(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.

FIGURE 1: Compression scheme on the first 38 blocks of Bitcoin ($K = 2$, $\chi = 9$ and $k = 2$).

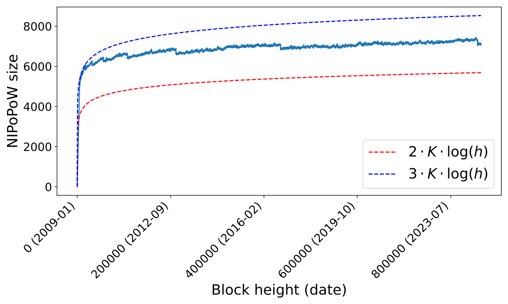


FIGURE 2: Evolution of the number of blocks kept in the proof over the Bitcoin blockchain length ($K = 208$, $\chi = 4032$ and $k = 323$).

5 Conclusion

In this work, we have proposed an instantiation of a NIPoPoW capable of handling variable mining difficulty. This improves upon previous existing instantiations [KLZ21, JAG23]. The proposed solution is resilient against a 1/3-adversary and, in particular, against low-difficulty attacks. Such an attack takes advantage of both variable difficulties and sampling. The adversary strategically mines their "old" blocks with a low-difficulty, making them numerous, and its "recent" ones with difficulties similar to the honest used ones, to hide their attack. As future works, we intend to propose a construction that is tolerant to a 1/2-adversary. This would make it fully compliant with existing PoW-based blockchains. We also intend to study their instantiation to DAG-based PoW-based blockchains.

References

[GKL17] Juan Garay, Aggelos Kiayias, and Nikos Leonardos. The bitcoin backbone protocol with chains of variable difficulty. In *Annual International Cryptology Conference*, pages 291–323. Springer, 2017.

[JAG23] Anurag Jain, Emmanuelle Anceaume, and Sujit Gujar. Extending the boundaries and exploring the limits of blockchain compression. In *IEEE 43th Symposium on Reliable and Distributed Systems*, SRDS, 2023.

[Kat13] Guy Katriel. Gamblers ruin probability—A general formula. *Statistics & Probability Letters*, 83(10):2205–2210, 2013.

[KLZ21] Aggelos Kiayias, Nikos Leonardos, and Dionysis Zindros. Mining in logarithmic space. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, CCS, pages 3487–3501, 2021.

[MPD$^+$24] Loïc Miller, Dorian Pacaud, Nathanael Derousseaux, Emmanuelle Anceaume, and Romaric Ludinard. Mining in Logarithmic Space with Variable Difficulty. working paper or preprint, November 2024.

[Nak08] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008.

[OL17] A Pinar Ozisik and Brian Neil Levine. An explanation of nakamoto's analysis of double-spend attacks. *arXiv preprint arXiv:1701.03977*, 2017.

[PD16] J. Poon and T. Dryja. The bitcoin lightning network: Scalable off-chain instant payments, 2016.

[SNA20] SNAP. Ethereum snapshot protocol, 2020.