Extending The Boundaries and Exploring The Limits Of Blockchain Compression

Anurag Jain, Emmanuelle Anceaume, Sujit Gujar

Extending The Boundaries and Exploring The Limits Of Blockchain Compression

Anurag Jain
International Institute of Information Technology (IIIT), Hyderabad, India
Emmanuelle Anceaume
CNRS, IRISA, Rennes, France
Sujit Gujar
International Institute of Information Technology (IIIT), Hyderabad, India

***Abstract*—The long-term feasibility of blockchain technology is hindered by the inability of existing blockchain protocols to prune the consensus data leading to constantly growing storage and communication requirements. Kiayias et al. have proposed Non-Interactive-Proofs-of-Proof-of-Works (NIPoPoWs) as a mechanism to reduce the storage and communication complexity of blockchains to $O(\text{poly}\log(n))$. However, their protocol is only resilient to an adversary that may control strictly less than a third of the total computational power, which is a reduction from the security guaranteed by Bitcoin and other existing Proof-of-based blockchains. We present an improvement to the Kiayias et al. proposal, which is resilient against an adversary that may control less than half of the total computational power while operating in $O(\text{poly}\log(n))$ storage and communication complexity. Additionally, we present a novel proof that establishes a lower bound of $O(\log(n))$ on the storage and communication complexity of any PoW-based blockchain protocol.**

***Index Terms*—Blockchains; Proof-of-work; NIPoPoWs.**

I. Introduction

Blockchains are proposed as the panacea for developing decentralized applications and handling decentralized finance. Blockchain technology needs to be capable of handling the entire world's economic data for such lofty ambitions to come true. The immutable nature of blockchain indicates that the storage requirements are constantly growing, with every block being retained for eternity. This further increases the communication requirements for a new honest party to join the system as it would need to download the entire blockchain from the network in order to start mining[1]. In a blockchain system, each party must independently maintain a copy of the blockchain to mine new blocks and verify transactions. The data stored by a party can be divided into two types: *Application data*, which consists of all information required to verify the validity of a transaction, e.g., UTXO (Unspent Transaction Output), account data, and smart contract states, and *Consensus data*, which consists of all information required, other than the application data, for the complete participation of a party to maintain its copy of the blockchain. A new honest party that joins the blockchain system must synchronize with both types of data in order to initiate mining. In a Proof-of-Work (PoW) blockchain protocol, the consensus data can grow linearly with every new block mined. This may cause issues with both storing the consensus data and communicating the same for bootstrapping a new party into the system. Multiple techniques exist to optimize application data that may grow or shrink as time goes and have been well deployed in practice. For instance, Ethereum uses snapshots, with the SNAP protocol run in parallel with the consensus protocol [19], that periodically summarize all active data, discarding inactive data that has been overwritten or mutated. Moving transactions and smart contract execution off-chain in Layer 2 constructions (e.g., [18], [13]), side chains (e.g., [2], [17], [10]) or compression of multiple transactions into smaller ones, e.g., [6], are significant methods to compress transaction data. However, these solutions do not compress consensus data. All PoW headers must still be sent and stored. Compressing consensus data is still an area of active research since the consensus data in existing protocols increases linearly with time.

Kiayias et al. [12] propose an elegant scheme to create succinct proofs about the fact that proof-of-work took place without presenting every proof-of-work nonce of the blockchain. These succinct proofs are generated once, and do not require multiple queries to convince any party. As such, these proofs are non-interactive, and gave rise to *Non-Interactive Proofs-of-Proof-of-Works* (NIPoPoWs). By relying on NIPoPoWs, for a blockchain containing n blocks, any party is only required to store consensus data corresponding to $O(\text{poly}\log(n))$ blocks, and for any party that wishes to join the system, it only needs to download the consensus data corresponding to $O(\text{poly}\log(n))$ set of blocks to commence the mining operation. To the best of our knowledge, Kiayias et al.'s proposal [12] is state of the art in terms of the storage and communication complexity. It promises to yield an order of magnitude reduction in both storage and communication requirements over existing PoW-based blockchain protocols. Indeed, to create succinct representations of work, NIPoPoWs rely on the probability distribution of the hash values of block headers in the blockchain and use this *a priori* to sub-sample the blocks: instead of sending the full chain, a party sends sampled blocks as a representative of the underlying work.

[1]In a permissionless blockchain every party, i.e., miner and non miner must download the full blockchain if they want to verify (non miner) and mine new blocks (miner). The ones that do not need to download the full blockchain are Simplified Payment Verification (SPV) parties, but such parties still need to download block headers and cannot check the security of the blockchain.

Sampled blocks, also called superblocks, contain a PoW that is "stronger than needed": A superblock of level ℓ is a block whose hash is less than $T/2^\ell$ where T is the current target and $\ell \geq 1$. To achieve secure reduction, Kiayias et al. [12] analyze their solution against *suppression attacks*. The objective of such an attack is to bias the succinct representation of the total work devoted to the construction of the blockchain by "suppressing" superblocks. This consists for the adversary in trying to fork the honest sampled chain to prune superblocks. As demonstrated in their paper, their succinct construction creates a security threat and can only guarantee tolerance against a Byzantine adversary that may control strictly less than a third of the total computational power, reducing security from the original Bitcoin protocol.

In this work, we address this important issue and present Gems, a scheme to construct a succinct representation of the blockchain using NIPoPoWs that also operates in $O(\text{poly}\log(n))$ storage complexity and $O(\text{poly}\log(n))$ communication complexity and which provably achieves security against a Byzantine adversary that controls strictly less than half of the total computational power. The main idea of our solution is *(i)* to assign increasing weights to ℓ-superblocks for $\ell > \beta$, where β is a threshold parameter (such superblocks are called *ℓ-diamond* blocks), and *(ii)* to modify the chain selection rule so that the selected succinct chain is the one that accumulates the largest weight. With the modified chain selection rule, it is improbable for an attacker controlling less than half of the total hashing power to bias the honest subsample of the blockchain by suppressing ℓ-diamond blocks. The crucial point of our solution in terms of security is that an adversary cannot fake this set of diamond-blocks without actually providing work. 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. This is shown in Section VIII. We thereby improve the security of Kiayias et al.'s scheme by tolerating a Byzantine adversary that may control up to half of the total hashing power.

The limitation of both our and Kiayias et al.'s solutions is that they are only proven to operate securely in a setting with constant difficulty. Tackling the problem of blockchain compression operating in a $O(\text{poly}\log(n))$ storage complexity and $O(\text{poly}\log(n))$ communication complexity with variable difficulty is an open problem.

On the other hand, our scheme that achieves optimal security while still operating in $O(\text{poly}\log(n))$ storage and communication complexity leaves the open question raised by Kiayias et al [12] of whether such a scheme is optimal or if there is further room for improvement. We progress on this by presenting a proof using information theory to establish a lower bound on the limit of blockchain compression.

Contributions of this work.

- We propose Gems, a NIPoPoW protocol that operates in $O(\text{poly}\log(n))$ storage complexity and $O(\text{poly}\log(n))$ communication complexity, which through a novel weight assignment scheme, makes it resilient to a Byzantine adversary that may control up to half of the total hashing power;
- For the first time, we provide proofs of boundaries of blockchain compression by showing that it is not possible to compress a PoW blockchain beyond $O(\log(n))$ storage and communication complexity without introducing additional trust assumptions.

Organization of the paper. Section II is dedicated to related work, while Section III presents the assumptions on the system, and Section IV presents the safety and liveness properties of permissionless blockchains. Section V presents the main lines of the Kiayias et al.'s approach to compress PoW-blockchains with NIPoPoWs. Section VI firt shows why selecting superblocks as a function of their weight guarantees with overwhelming probability their unsuppressibility, and second proposes a weight assignment schema. Section VII presents Gems' chain compression comparison algorithms. Section VIII analyzes the security of Gems. In Section IX, we present a lower bound on the storage space and communication complexities for any permissionless PoW-based blockchain protocol. Section X presents some future work.

II. Related Work

The problem of blockchain becoming of considerable size was initially predicted by Satoshi Nakomoto in the original paper that introduced Bitcoin [16]. He offered a simple solution of a *Simplified Payment Verification (SPV)* that only requires a client to store the 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. An alternative would be for SPV clients to embed hardcoded checkpoints but that would introduce additional trust assumptions. Flyclient [5] 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 proof without holding the whole chain. Thus, logarithmic space mining is not possible with this scheme. CoinPrune [15] still requires to store the entire chain of block header prior to the pruning point.

Another approach to built succinct proofs is to rely on SNARKS (for Succinct Non-Interactive Argument of Knowledge). Coda [4] 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.

Kiayias et al. [11] introduced and formalized an interactive proof mechanism, *Proofs-of-Proof-of-Work* (PoPoW) based on superblocks that allows a client to verify a chain in sublinear time and communication complexity. The authors later showed the existence of an attack on the scheme and proposed a non-interactive alternative (NIPoPoWs) [12]. However, the proposed solution did not address the size of the blockchain that needed to be stored by any miner. The authors in [12]

further used NIPoPoWs to develop a scheme that also allows the miners to operate in $O(\text{poly}\log(n))$ storage and communication complexity while reducing the security tolerance to a Byzantine adversary that controls strictly less than a third of the total computation power and limiting itself to operate in an environment with a fixed difficulty. We build upon their solution to present a scheme with improved security.

III. Model of the System

We consider a static setting where time is quantized into discrete rounds [7], [12], during which each party can send a message to each of its neighbours, receives the messages sent to it during the round, and executes computational steps based on the received messages. Note that computational steps other than hashing are treated as instantaneous. This reflects a *synchronous* network. We assume the presence of a Byzantine or malicious adversary that may control strictly less than half of the total amount of computational power currently available in the system. This model, named the "Computational Threshold Adversary" [1], is an alternative to the common Threshold Adversary Model. Each party is allowed to make q queries to a cryptographic hash function in every round. The adversary controls up to t parties. For this reason, the adversary can query the cryptographic hash function up to $t \times q$ times per round [7]. 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 but remains computationally bounded. Hence, it cannot, in a polynomial number of steps or time or space, forge honest parties' signatures or break the hash function and signature scheme with all but negligible probability. Therefore, we term our adversary as the *1/2-bounded PPT adversary*. Any party following the prescribed protocol is called a *honest* party.

IV. Properties of permissionless blockchains

We consider *proof-of-work* (PoW) based permissionless blockchains which achieve Nakamoto-based consensus [7], [20] by requesting the parties to contribute a limited resource such as hashing power. A robust blockchain protocol must ensure the following safety and liveness properties [7]:

Common prefix with parameter $k \in \mathbb{N}$: If any two honest parties have a valid transaction that appears in a block that is at least k blocks away from the end of their blockchain, then this transaction will appear at the same position in both blockchains, with overwhelming probability.

Chain growth with parameters $u \in \mathbb{N}$: if all honest parties try to insert a valid transaction in their blockchain for u consecutive rounds, the transaction shall be accepted by any honest party by the end of the last round of the set of u rounds with overwhelming probability.

V. Non-Interactive Proofs-of-Proof-of-Works

A. Intuition

The proof-of-work schema requires to generate a "proof" of investment of a limited resource such as hash power. Every party that wants to append a valid block to the blockchain is required to find a *nonce*, along with non-double spending transactions, that hashes to a value below a given target. The hash function $\mathcal{H}$ is modelled as a random oracle [3], i.e., behaves likes an ideal random function, and produces constant length output. Since the distribution of hash values is stochastic, some blocks end up with hash values significantly below the target. Such blocks are called superblocks.

Definition V.1 (ℓ-superblock ([12]))**.** A block that hashes to a value less than $T/(2^\ell)$ is said to be a ℓ-superblock, where T is the current target value and $\ell \geq 0$.

Note that every ℓ-superblock is also a ℓ'-superblock for any $\ell' \leq \ell$. By convention the genesis block is an ∞-superblock.

Non-Interactive Proofs-of-Proof-of-Works (NIPoPoWs) compress a PoW-based blockchain by subsampling its blocks [11]. The working principle behind this compression lies in the assumption that a sub-sample of the blocks, i.e., the ℓ-superblocks, with $\ell \geq 0$, can be sufficient to estimate the size of the original distribution of block headers. In a long enough execution of a PoW blockchain, on average, $n/2^\ell$ of the blocks are ℓ-superblocks, with $\ell \geq 0$ and n the number of blocks of the original chain. The key idea is to sub-sample the blocks in the blockchain such that the sub-sampled chain represents the original chain; any difference in the original blockchain results in different sub-sampled blockchains [9], [12], [14]. The compression scheme requires every block header to store pointers to the last ℓ-superblock that precedes it at every level $\ell \geq 0$ in order to ensure that the subsampled blocks also form a valid chain, i.e., a totally ordered sequence of valid blocks. Figure 1a illustrates how blocks at each level are linked together (note that we have omitted all the links to the genesis block for clarity reasons). For instance the 32-th block points back to the 31-th, 30-th, 28-th, 24-th and 16-th blocks, each one representing the last ℓ-superblock with $\ell = 0, 1, 2, 3$ and 4 respectively. On the other hand, the 33-th block points back solely to the 32-th block as this latter block is a 4-superblock (and thus also a ℓ-superblock, with $\ell \leq 4$). A chain of n blocks will contain superblocks at $O(\log(n))$ levels in average. Hence, the space and communication complexity of NIPoPoW is $O(\text{poly}\log(n))$. The proposal by Kiayias et al. [12] offers the best-known compression of PoW blockchains so far. It achieves $O(\text{poly}\log(n)c + kd + a)$ storage and communication costs while allowing parties to mine new blocks based on this compressed blockchain, where c is the block header size, k is the common prefix parameter, d is the size of application data per block, and a is the size of application data. In Section IX, we show that any such compression needs at least $O(\log(n))$ blocks without additional trust assumptions.

B. Algorithmic Ingredients of the NIPoPoW

Any scheme for operating and compressing blockchains requires to design (i) a *chain compression* algorithm and (ii) a *compressed chain comparison* algorithm to determine which compressed chain to be retained in the case of forks.

1) Chain Compression Algorithm: The Kiayias et al.'s chain compression algorithm (see Algorithm 1 [12]) is parameterized by a security parameter m and the common prefix parameter k. The algorithm compresses the blockchain except for the k most recent blocks, called *unstable* blocks, which are added to the compressed chain at the end of the compression process. The compression algorithm works as follows: for the highest levels ℓ that contain less than $2m$ ℓ-superblocks, it retains all these ℓ-superblocks, but for each level μ, $\ell > \mu \geq 0$, it retains at least the $2m$ most recent μ-superblocks so that m of them are both μ-superblocks and $(\mu+1)$-superblocks. Figure 1b illustrates this algorithm, where χ is the unstable part of the proof and $\mathcal{D}$ its stable part, i.e., all the ℓ-superblocks that at some time t have been retained by the compression algorithm at each level $\ell \geq 0$. Π represents an instance of NIPoPoW proof, that is the chain made of $\mathcal{D}$ and χ.

2) Compressed Chain Comparison Algorithm: Let $\Pi_1, \ldots, \Pi_j$ be the different compressed blockchains that a new party receives. This party applies the comparison algorithm on $\Pi_1, \ldots, \Pi_j$ pairwise. The algorithm first verifies their syntactic validity, extracts for both of them the stable parts $\mathcal{D}_i$ and $\mathcal{D}'_i$ and the unstable ones χ_i and χ'_i, and then selects the largest common part between $\mathcal{D}_i$ and $\mathcal{D}'_i$, i.e., the latest block that is common to both $\mathcal{D}_i$ and $\mathcal{D}'_i$ at the minimum level μ. If such a common block b exists, then the proof with the greatest number of blocks after b wins the comparison. Otherwise, if no such block b exists and the greatest levels of $\mathcal{D}_i$ and $\mathcal{D}'_i$ are different, then the algorithm selects the one with the greatest level. Finally if their greatest levels are equal, then they represent both a chain that has involved the same computational power, consequently any of them can be accepted as a correct proof.

C. Security

The high-level intuition behind the security of Kiayias et al.'s proposal [12] is that the common prefix property holds true not only for the proof but also at every level of the proof. So the adversary cannot possibly produce a blockchain that forks superblocks at a higher level. It is thus safe to accept the chain having the greatest number of superblocks at the level where we find the fork. However, the common prefix property of Kiayias et al.'s proposal [12] only holds true if the adversary cannot possibly produce a chain containing a larger number of ℓ-superblocks than what the honest parties can do. Since the security of the protocol relies heavily on these superblocks, it can be shown that the optimal attack for the adversary is to "suppress" these superblocks by forking the chain. Lemma 6.7 of [12] shows that an adversary with at least a third of the total hashing power can possibly suppress all the superblocks. Hence, their protocol can be proven to be secure only if the adversary is allowed to control less than one third of the total hashing power, which is a reduction from the security guaranteed by Bitcoin [7].

VI. Gems Blockchain: Weight intuition

The notion of unsuppressible blocks is fundamental to guarantee the quality of the compressed chain, i.e., to ensure that the distribution of superblocks within the blockchain has not been adversarially biased in the compressed chain. Unfortunately, Kiayias' proposal uses the default blockchain selection rule that is oblivious to the presence of superblocks. With Gems, suppression of superblocks is hindered. We designate a small fraction of the superblocks with level $\ell \geq \beta$ as *ℓ−diamond blocks* and assign increasing weight $W(\ell)$ to them as per their level. The system parameter β quantifies the trade-off between storage and communication costs and increasing weights. The chain selection rule relies on block cumulative weight. The intuition of unsuppressibility comes from the fact that an adversary that does not control the majority of the total hashing power cannot produce a sampled chain that has more weight than the honest sampled chain containing a ℓ-diamond block without including another ℓ-diamond block in their adversarial sampled chain.

A. Typical Execution Constraints

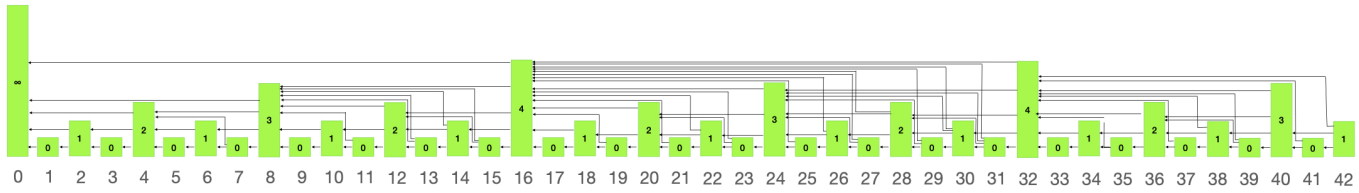
Since the mining of blocks is a probabilistic process, it can often happen that the adversary luckily manages to mine more blocks than the honest network during short periods of time. So similarly to Garay et al. [7], we consider long enough executions, so that the adversary cannot be always lucky during these long executions. Specifically, we consider executions with at least λ consecutive rounds so that the honest parties mine more blocks than the adversary with a probability at least $1 - e^{-\Omega(\epsilon^2 \lambda f + \kappa)}$, where $\epsilon \in (0,1)$, κ is the length of hash function outputs (i.e., $\kappa = 256$), λ satisfies $\lambda \geq 2/f$, f being the probability that at least one honest party successfully mines a block during one round, and $\epsilon \in (0,1)$. Let δ be the advantage of honest parties, with $\delta \leq 1 - t/(N-t)$, where N represents the number of mining parties, out of which t are controlled by the adversary. Both δ and f relate as $3(f+\epsilon) < \delta < 1$. Now, for any set S of rounds, let $Y(S)$ be the random variable representing the number of rounds in S in which the honest parties produced exactly one block. Let $Z(S)$ be the random variable representing the number of blocks produced by the adversary in the set of rounds S. Lemma VI.1 provides guarantees on the power of the adversary during long enough executions.

Lemma VI.1 ([7], Lemma 11)**.** *For any set S of at least λ consecutive rounds, we have*

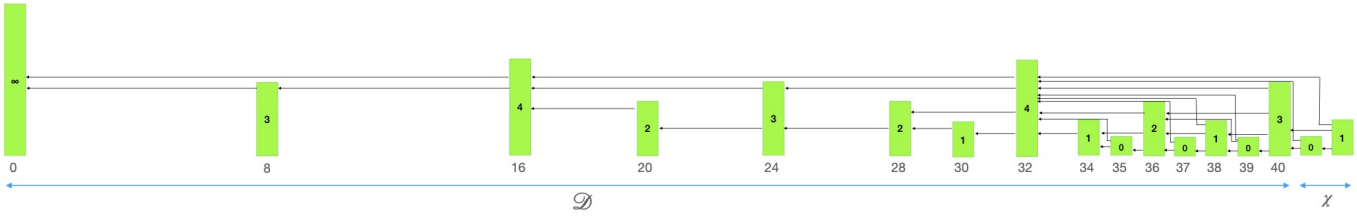
(a) $(1 - \frac{\delta}{3}) f|S| < (1-\epsilon) f(1-f)|S| < Y(S)$,

(b) $Z(S) < \frac{t}{n-t} \frac{f}{1-f} |S| + \epsilon f |S| \leq (1 - 2\frac{\delta}{3}) f|S|$,

(c) $Z(S) < Y(S)$.



(a) The probabilistic hierarchical blockchain. Higher levels have achieved a higher difficulty during mining.



(b) View of the blockchain after compression at time t.

Figure 1: Illustration of Kiayias et al.'s [12] compression scheme. Note that we have omitted a link from each block to the genesis block for clarity reason. All blocks are labelled by their level. Remember that a block at level ℓ, for any $\ell > 0$ is also a block at level $\ell - 1$. Compression parameters are $m = 3$ and $k = 2$ (Figure (b)). Level 3 is the highest level ℓ that contains at least $2m$ ℓ-superblocks. The set of 3-superblocks = $\{0, 8, 16, 24, 32, 40\}$, the one of 2-superblocks = $\{20, 24, 28, 36, 40\}$, 1-superblocks = $\{30, 32, 34, 38, 40\}$ and the set of 0-superblocks = $\{35, 36, 37, 38, 39, 40\}$.

B. Warm-up: Assigning common weight to superblocks

We show that attaching a large enough weight to blocks essentially makes them "unsuppressible". Prior to discussing how blocks are assigned weight, we present a series of lemmas that demonstrate the positive effect of attaching weights to blocks on their security. In the next section, we shall generalize these results to devise a weight assignment policy specific to each level of superblocks. Lemma VI.2 provides an upper-bound on the number of blocks mined by the adversary in an execution that contains less than λ rounds.

Lemma VI.2. *For any set S of consecutive rounds such that $|S| \leq \lambda$, $Z(S) < (1 - (2\delta)/3)f\lambda$.*

Proof. We prove this lemma by contradiction. Let U be a typical execution with $|U| = \lambda$, and let $S \subset U$, i.e., $|S| < \lambda$. Suppose by contradiction that $Z(S) \geq (1 - (2\delta)/3)f\lambda$.
By Lemma VI.1, we have

$$Z(U) < \big(1 - \frac{2\delta}{3}\big) f|U|.$$

By definition of execution U, we have

$$\begin{aligned} Z(U) &= Z(S) + Z(U \setminus S) \\ &\geq Z(S). \end{aligned}$$

By assumption of the proof, we get

$$\begin{aligned} Z(U) &\geq \big(1 - \frac{2\delta}{3}\big) f\lambda \\ &= \big(1 - \frac{2\delta}{3}\big) f|U|. \end{aligned}$$

However, from Lemma VI.1 we have that for any set of consecutive rounds S' with $|S'| \geq \lambda$, $Z(S') < (1 - (2\delta)/3)f|S'|$. Hence, execution U cannot be a typical execution, which contradicts the assumption, and completes the proof. □

Lemma VI.3. *If S is a sequence of consecutive rounds, then, $Y(S \setminus \{r\}) \geq Y(S) - 1$ for any round $r \in S$.*

Proof. The set $S \setminus \{r\}$ only discards the round r which can contribute a maximum of one uniquely successful round in $Y(S)$.

$$Y(S) = Y(S \setminus \{r\}) + Y(\{r\}),$$

and thus

$$\begin{aligned} Y(S \setminus \{r\}) &= Y(S) - Y(\{r\}) \\ &\geq Y(S) - 1, \end{aligned}$$

as by definition $Y(\{r\}) \leq 1$. This completes the proof. □

Lemma VI.4 provides an upper bound on the number of blocks the adversary can successfully mine in addition to the ones mined by the honest parties.

Lemma VI.4. *For any sequence S of rounds, $|S| \geq 1$, and for any round $r \in S$, we have*

$$\max_{\forall\ S} \big(Z(S) - Y(S \setminus \{r\})\big) < \left(1 - \frac{\delta}{3}\right) f\lambda.$$

Proof. Two cases exist:
Case I: $|S| < \lambda$.
From Lemma VI.2, $Z(S) < (1 - (2\delta)/3)f\lambda$. By definition, $Y(S) \geq 0$. Hence, $\big(Z(S) - Y(S \setminus \{r\}) < \big(1 - \frac{\delta}{3}\big) f\lambda$.
Case II: $|S| \geq \lambda$.
From the conditions of typical executions we know that for

$|S| \geq \lambda$, $Z(S) < (1-\delta/3)f|S|$ and by Lemmas VI.3 and VI.1, we have $Y(S \setminus \{r\}) \geq (1-\delta/3)f|S| - 1$. Thus,

$$Z(S) - Y(S \setminus \{r\}) \leq \left(1 - \frac{2\delta}{3}\right) f|S| - \left(1 - \frac{\delta}{3}\right) f|S| + 1$$
$$\leq 1 - \frac{\delta}{3} f|S|.$$

By definition of λ, i.e., $\lambda \geq 2/f$, we thus have

$$1 - \frac{\delta}{3} f|S| < \left(1 - \frac{\delta}{3}\right) f\lambda.$$

Thus, across both cases, the maximum value that the expression can take is $(1-\delta/3)f\lambda$. This completes the proof. □

From Lemma VI.4, we can infer the minimum weight value blocks must be assigned to so that the adversary will not possibly be able to suppress them. Specifically, the following theorem gives a lower bound w_0 on the weight assigned to superblocks guaranteeing their unsupressibility. Let us denote by (w_0, ℓ)-block a ℓ-superblock with a weight equal to w_0.

Theorem VI.5 ((w_0, ℓ)-superblocks are unsuppressible)**.** *If a (w_0, ℓ)-superblock is attached a weight w_0 with $w_0 \geq \left(1 - \frac{\delta}{3}\right) f\lambda$, then for any blockchain $\mathcal{C}$ adopted by an honest party such that $\mathcal{C}$ contains a (w_0, ℓ)-superblock b, then with overwhelming probability, the adversary cannot replace $\mathcal{C}$ by another blockchain $\mathcal{C}'$ such that $|\mathcal{C}| = |\mathcal{C}'|$ and $\mathcal{C}'$ does not contain a (w_0, ℓ)-superblock b'.*

Proof. Any block mined by the honest parties has a probability $1/2^\ell$ to be a ℓ-superblock. Let w_0 be the weight attached to any such (w_0, ℓ)-superblock. Consider the case where the adversary tries to produce a blockchain $\mathcal{C}'$ alternative to blockchain $\mathcal{C}$ produced by the honest parties such that $\mathcal{C}$ contains a (w_0, ℓ)-superblock b but $\mathcal{C}'$ does not contain any (w_0, ℓ)-superblocks. The adversary can suppress (w_0, ℓ)-superblock b mined in round r if for any S we have

$$w_0 + Y(S \setminus \{r\}) \leq Z(S)(1 - \frac{1}{2^\ell}). \tag{1}$$

So, if we want to prevent (w_0, ℓ)-block b from being suppressed, we must have that

$$w_0 \geq \max_{\forall\ S} \left(Z(S) - Y(S \setminus \{r\}) \right). \tag{2}$$

From Lemma VI.4, we get

$$w_0 \geq \left(1 - \frac{\delta}{3}\right) f\lambda, \tag{3}$$

which completes the proof of the theorem. □

C. Gems' Block Weight Assignment

The Gems' block weight assignment policy relies on Theorem VI.5. It assigns unitary weights to the large majority of superblocks, and non-unitary ones to a small fraction of ℓ-superblocks that we call ℓ-diamond blocks. We propose the following weight assignment with the system parameter β assumed to be known by all the parties in advance.

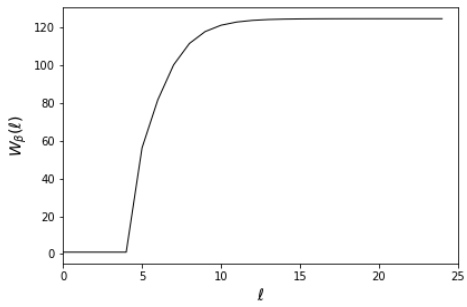


Figure 2: $W_\beta(\ell)$ as a function of ℓ with $\beta = 5$ and parameters $f = 0.03$, $\delta = 0.1$, $\epsilon = 10^{-5}$, and $\lambda = 1,000$.

Definition VI.1. Gems' blocks weight assignment

$$W_\beta(\ell) = \begin{cases} 1 & \ell < \beta \\ (1+\epsilon)\Big(\sum_{i=0}^{\ell-1} \frac{W_\beta(i)}{2^i}\Big)\big(1 - \frac{\delta}{3}\big) f\lambda & \ell \geq \beta \end{cases}$$

The weight assignment scheme, parameterized by β, does not grow exponentially if β is set appropriately. Considering a value of $\beta = 5$, we obtain the plot depicted in Figure 2 where parameters f, δ, ϵ, and λ are the ones used by Garay et al [7] in their proof of Bitcoin's security.

Definition VI.2 (ℓ-diamond-block)**.** We define a ℓ-diamond-block as a block that contains a hash whose value is less than $T/(2^\ell)$ where $\ell \geq \beta$, and T is the target value.

VII. Gems' Algorithms

Notations. We borrow the notation and the mathematical framework introduced by Kiayias et al. in [12]. $\mathcal{C}[i]$ denotes a block in the chain $\mathcal{C}$ with zero-based indexing, while $\mathcal{C}[i:j]$ denotes the blocks from the index i (inclusive) to j (exclusive), omitting any of the two implies taking all the blocks that follow. A negative i or j means to take blocks from the end of the chain instead of from the beginning, so $\mathcal{C}[-1]$ is the tip of the chain. If i and j are replaced by blocks A and Z instead of block indices, we write $\mathcal{C}\{A:Z\}$ to designate blocks of $\mathcal{C}$ from block A (inclusive) to block Z (exclusive), and again any end can be omitted. $\mathcal{C} \uparrow^\mu$ refers to only the subsequence of μ-superblocks in the entire chain $\mathcal{C}$. By definition of μ-superblocks, $(\mathcal{C} \uparrow^\mu) \uparrow^{\mu+i} = \mathcal{C} \uparrow^{\mu+i}$.

A. Chain Compression Algorithm

The chain compression algorithm requires each block header to maintain pointers to the last superblock at every level to form an interlinked blockchain. The algorithm works by only keeping sufficiently many diamond blocks at every level and discarding the rest of the blocks. These samples evolve with time, i.e., a diamond block once selected may be discarded later, but vice-versa does not hold. Pseudo-code of the algorithm is presented in Algorithm 2. The chain compression algorithm is parameterized by a security parameter m and the common prefix parameter k (whose value is derived in Lemma VIII.8), and the system parameter β.

The algorithm compresses the blockchain except for the k most recent diamond blocks (see Line 2), which are called *unstable* blocks. The compression works as follows: For the highest levels ℓ that contain less than $2m$ superblocks, keep all these ℓ-superblocks but for each level μ, $\ell > \mu \geq \beta$, the algorithm retains the most recent μ-superblocks: let b be the $2m$-th most recent μ-superblock and b' be the m-th most recent $\mu+1$-superblock. Then all the μ-superblocks that follow the earlier of b and b' are retained. Proof Π is made of the set of superblocks that are retained after compression and of the blocks of the unstable part χ, containing the sequence of the k most recent diamond blocks (see Line 14).

Algorithm 1 Chain compression algorithm: Given a full chain, it compresses it into a logspace state.

1: **function** DISSOLVE $_{m,k,\beta}(\mathcal{C})$
2: $\quad\mathcal{C}^* \leftarrow \mathcal{C}\{: C \uparrow^{\beta} [-k]\}$
3: $\quad\mathcal{D} \leftarrow \emptyset$
4: $\quad$**if** $|\mathcal{C}^*| \geq 2m$ **then**
5: $\quad\quad\ell \leftarrow \max\{\mu : |\mathcal{C}^* \uparrow^{\mu}| \geq 2m\}$
6: $\quad\quad\mathcal{D}[\ell] \leftarrow \mathcal{C}^* \uparrow^{\ell}$
7: $\quad\quad$**for** $\mu \leftarrow \ell - 1$ down to β **do**
8: $\quad\quad\quad b \leftarrow \mathcal{C}^* \uparrow^{\mu+1} [-m]$
9: $\quad\quad\quad\mathcal{D}[\mu] \leftarrow \mathcal{C}^* \uparrow^{\mu} [-2m :] \cup \mathcal{C}^* \uparrow^{\mu} \{b :\}$
10: $\quad\quad$**end for**
11: $\quad$**else**
12: $\quad\quad\mathcal{D}[0] \leftarrow \mathcal{C}^*$
13: $\quad$**end if**
14: $\quad\chi \leftarrow \mathcal{C}\{C \uparrow^{\beta} [-k] :\}$
15: $\quad$**return** $(\mathcal{D}, \ell, \chi)$
16: **end function**
17: **function** COMPRESS $_{m,k,\beta}(\mathcal{C})$
18: $\quad(\mathcal{D}, \ell, \chi) \leftarrow$ Dissolve $_{m,k,\beta}(\mathcal{C})$
19: $\quad\pi \leftarrow \bigcup_{\mu=0}^{\ell} \mathcal{D}[\mu]$
20: $\quad$**return** $\pi\chi$
21: **end function**

B. Compressed Chain Comparison Algorithm

Any new party that seeks to join the network and starts mining new blocks must first synchronize with other parties in the network by identifying the longest chain. Given multiple compressed chains, the compressed chain comparison algorithm identifies the one with the "largest" PoW. Given multiple compressed chains, $\Pi_1, \Pi_2, \ldots, \Pi_n$, pairwise compare each chain to obtain the chain that captures the most proof-of-work. We describe the pseudo-code of our chain selection algorithm in Algorithm **??**. After having performed a syntactic validity check (Lines 1-7) on two compressed chains Π_i and Π_j, first, separate the last k diamond blocks as χ_i and χ_j from the rest of the chains as $\mathcal{D}_i$ and $\mathcal{D}_j$ respectively. Note that the Dissolve function is invoked with compressed chains and not full chains. If $\mathcal{D}_i = \mathcal{D}_j$ then select the chain having greater weight among χ_i or χ_j else compare $\mathcal{D}_i$ and $\mathcal{D}_j$ in the same manner as Kiayias et al.'s proposal. It is important to notice that our modification applies solely to the case where the provided stable portions of the compressed chain $\mathcal{D}$ are identical. In that case we compare the unstable portions of the chains χ and χ' using their weights (Lines 17-21). Our modification is minor however, the security analysis is non-trivial (see Section VIII).

Algorithm 2 Chain compression algorithm: Given a full chain, it compresses it into a logspace state.

1: **function** DISSOLVE $_{m,k,\beta}(\mathcal{C})$
2: $\quad\mathcal{C}^* \leftarrow \mathcal{C}\{: C \uparrow^{\beta} [-k]\}$
3: $\quad\mathcal{D} \leftarrow \emptyset$
4: $\quad$**if** $|\mathcal{C}^*| \geq 2m$ **then**
5: $\quad\quad\ell \leftarrow \max\{\mu : |\mathcal{C}^* \uparrow^{\mu}| \geq 2m\}$
6: $\quad\quad\mathcal{D}[\ell] \leftarrow \mathcal{C}^* \uparrow^{\ell}$
7: $\quad\quad$**for** $\mu \leftarrow \ell - 1$ down to β **do**
8: $\quad\quad\quad b \leftarrow \mathcal{C}^* \uparrow^{\mu+1} [-m]$
9: $\quad\quad\quad\mathcal{D}[\mu] \leftarrow \mathcal{C}^* \uparrow^{\mu} [-2m :] \cup \mathcal{C}^* \uparrow^{\mu} \{b :\}$
10: $\quad\quad$**end for**
11: $\quad$**else**
12: $\quad\quad\mathcal{D}[0] \leftarrow \mathcal{C}^*$
13: $\quad$**end if**
14: $\quad\chi \leftarrow \mathcal{C}\{C \uparrow^{\beta} [-k] :\}$
15: $\quad$**return** $(\mathcal{D}, \ell, \chi)$
16: **end function**
17: **function** COMPRESS $_{m,k,\beta}(\mathcal{C})$
18: $\quad(\mathcal{D}, \ell, \chi) \leftarrow$ Dissolve $_{m,k,\beta}(\mathcal{C})$
19: $\quad\pi \leftarrow \bigcup_{\mu=0}^{\ell} \mathcal{D}[\mu]$
20: $\quad$**return** $\pi\chi$
21: **end function**

VIII. GEMS SECURITY ANALYSIS

A. Preliminary results

Let $\Sigma(S)$ be the function that returns the total weight of all the blocks (see Definition VI.1) appended to the blockchain in a set of rounds S. Similarly, let $\Sigma_Y(S)$ and $\Sigma_Z(S)$ be the functions that respectively represent the weights of the uniquely successful honest blocks and adversarial ones in S rounds. Let $\Sigma_Z^{\ell}(S)$ be the function that returns the total weight of adversarial superblocks at level ℓ and $\Sigma_Z^{\ell^-}(S)$ be the function that returns the weight of adversarial blocks in S rounds considering only blocks with level strictly less than ℓ. $\Sigma_Y^{\ell}(S)$ and $\Sigma_Y^{\ell^-}(S)$ are similarly defined but for uniquely successful honest blocks. By extension, random variable $Y_{\ell}(S)$, for any $\ell \geq 0$, represents the number of uniquely successful ℓ-superblocks mined in S rounds. Lemma VIII.1 uses Chernoff Bounds to provide lower bounds on the number of superblocks in a set of rounds S.

Lemma VIII.1. *For any set S of consecutive rounds such that $|S| \geq \lambda$ and $\epsilon \in (0, 1]$, we have $(1-\epsilon)\frac{Y(S)}{2^{\ell}} \leq Y_{\ell}(S)$ with overwhelming probability.*

Proof. Let x_j be the random variable that is equal to 1 if the j^{th} block appended to the blockchain in the set S of rounds

is a ℓ superblock, and is equal to 0 otherwise. We have,

$$x_j = \begin{cases} 1 & \text{with probability } 1/2^\ell \\ 0 & \text{otherwise with probability } 1 - 1/2^\ell. \end{cases}$$

We have $\mathbb{E}[x_j] = 1/2^\ell$, and $\mu = \mathbb{E}[Y_\ell(S)] = \frac{Y(S)}{2^\ell}$. From Chernoff Bounds,

$$\mathbb{P}\left\{Y_\ell(S) \geq (1-\epsilon)\mu\right\} \geq 1 - e^{-\frac{2\epsilon^2\mu^2}{Y(S)}},$$

which completes the proof. □

Lemma VIII.2. *For any set S of consecutive rounds such that $|S| \geq \lambda$, and $\epsilon \in (0, 1]$, we have $Z_\ell(S) \leq (1+\epsilon)\frac{Z(S)}{2^\ell}$ with overwhelming probability.*

Proof. Let x_j be the random variable that is equal to 1 if the j^{th} block appended to the blockchain in the set S of rounds is a ℓ superblock, and is equal to 0 otherwise. We have,

$$x_j = \begin{cases} 1 & \text{with probability } 1/2^\ell \\ 0 & \text{otherwise with probability } 1 - 1/2^\ell. \end{cases}$$

By definition of x_j, $\mathbb{E}[x_j] = 1/2^\ell$. Let μ be the expectation of random variable $Z_\ell(S)$. We have

$$\mu = \mathbb{E}[Z_\ell(S)] = \frac{Z(S)}{2^\ell}$$

Then from Chernoff Bounds,

$$\mathbb{P}\left\{Z_\ell(S) \leq (1+\epsilon)\mu\right\} \geq 1 - e^{-\frac{2\epsilon^2\mu^2}{Z(S)}},$$

which completes the proof. □

The following lemma from [12] ensures that in a typical execution, the honest parties not only produce a greater number of blocks than what the adversary does, but also a greater number of superblocks. Lemma VIII.4 further proves that the advantage of honest parties also applies to weights.

Lemma VIII.3 ([12] Lemma 1(d))**.** *$Y_\ell(S) > Z_\ell(S)$ for any set S of consecutive rounds such that $|S| \geq \lambda$ and $\ell \in \mathbb{Z}^+$.*

Lemma VIII.4. *For any set S of consecutive rounds such that $|S| \geq \lambda$, we have $\Sigma_Y(S) > \Sigma_Z(S)$.*

Proof. By definition of $\Sigma_Y(S)$ we have,

$$\begin{aligned} \Sigma_Y(S) &= \sum_{i=0}^{\infty} \Sigma_Y^i(S) \\ &= \sum_{i=0}^{\infty} W_\beta(i) Y_i(S) \\ &\geq \sum_{i=0}^{\infty} W_\beta(i) Z_i(S) \qquad \text{(from Lemma VIII.3)} \\ &\geq \Sigma_Z(S), \end{aligned}$$

which completes the proof. □

A block mined by honest parties and belonging to the chain currently adopted by honest parties will be forked by the adversary if and only if the adversary is capable of providing a chain whose total weight is larger than the one adopted by the honest parties. Lemma VIII.5 captures this condition.

Lemma VIII.5. *If r is a uniquely successful round and the corresponding block does not belong to the chain of an honest party at a later round, then there is a set of consecutive rounds S such that $r \in S$ and $\Sigma_Y(S) \leq \Sigma_Z(S)$.*

Proof. Let $\mathcal{C}$ be the chain of the honest party that was uniquely successful at round r and u be the depth of the corresponding block. Let r' be the first round after r in which an honest party has a chain $\mathcal{C}'$, which does not contain the block at depth u. Let r^* be the round in which the last block common to both $\mathcal{C}$ and $\mathcal{C}'$ was mined. Now for the set $S = \{i : r^* < i < r'\}$, we have $\Sigma_Y(S) \leq \Sigma_Z(S)$. Indeed, if this is not the case, one of the honest parties adopted a chain that was not the heaviest chain available which contradicts our assumption that the party was honest. This completes the proof. □

The following lemma provides an upper-bound on the weight of blocks mined by the adversary.

Lemma VIII.6. *for any set of rounds S such that $|S| \leq \lambda$, $\Sigma_Z^{\ell^-}(S) < (1+\epsilon)\Big(\sum_{i=0}^{l-1} \frac{W_\beta(i)}{2^i}\Big)\big(1 - \frac{\delta}{3}\big) f\lambda$.*

Proof. Let x_j be the random variable that is equal to 1 if the j^{th} block appended to the blockchain is a ℓ superblock, which occurs with probability $1/\ell$, and 0 otherwise, which occurs with probability $1 - 1/\ell$. We have $\mathbb{E}[x_j] = 1/2^\ell$. Then,

$$\Sigma_Z^\ell(S) = W_\beta(\ell)\left(\sum_{j=0}^{j=Z(S)} x_j\right),$$

and

$$\mu = \mathbb{E}[\Sigma_Z^\ell(S)] = W_\beta(\ell)\frac{Z(S)}{2^\ell}.$$

Then from Chernoff Bounds,

$$\mathbb{P}\left\{\Sigma_Z^\ell(S) < (1+\epsilon)\mu\right\} \geq 1 - e^{-\frac{2\epsilon^2\mu^2}{Z(S)}}.$$

Now, let $S \subseteq U$ such that $|U| = \lambda$. By definition of function $\Sigma()$, we have $\Sigma_Z^\ell(S) \leq \Sigma_Z^\ell(U)$, thus

$$\mathbb{P}\left\{\Sigma_Z^\ell(U) < W_\beta(\ell)\frac{(3-\delta)\lambda}{3}\frac{1}{2^\ell}\right\} \geq 1 - e^{-\left(\frac{2\epsilon^2(3-\delta)\lambda}{3}\right)\left(\frac{1}{2^\ell}\right)}.$$

Therefore, $\Sigma_Z^\ell(S) \leq \Sigma_Z^\ell(U) < W_\beta(\ell)((3-\delta)\lambda/3)(1/2^\ell)$ with overwhelming probability. Therefore

$$\Sigma_Z^{\ell^-}(S) = \sum_{i=0}^{\ell-1} \Sigma_Z^\ell(S) < (1+\epsilon)\left(\sum_{i=0}^{l-1}\frac{W_\beta(i)}{2^i}\right)\left(1 - \frac{\delta}{3}\right) f\lambda,$$

holds with overwhelming probability. □

Theorem VIII.7 states that the proposed block weight assignment prevents the adversary from suppressing diamond blocks. The proof of the theorem is analogous to the proof of Theorem VI.5.

Theorem VIII.7 (ℓ-diamond blocks unsuppressibility)**.** *If ℓ-diamond blocks are attached a weight $W_\beta(\ell)$, then for any*

chain $\mathcal{C}$ adopted by an honest party such that $\mathcal{C}$ contains a ℓ-diamond block, then with overwhelming probability, the adversary cannot replace $\mathcal{C}$ by another chain $\mathcal{C}'$ such that $|\mathcal{C}| = |\mathcal{C}'|$ and $\mathcal{C}'$ does not contain any ℓ-diamond block.

Proof. Let us assume by contradiction that the adversary has replaced chain $\mathcal{C}$ with chain $\mathcal{C}'$ such that $\mathcal{C}'$ does not contain any diamond blocks. From Lemma VIII.5, it must exist S such that $\Sigma_Y(S) \leq \Sigma_Z^{\ell^-}(S)$. Let us consider two cases:
Case I: $|S| \leq \lambda$.
From Lemma VIII.6, the maximum value $\Sigma_Z^{\ell^-}(S)$ can take is $(1+\epsilon)\Big(\sum_{i=0}^{l-1} \frac{W_\beta(i)}{2^i}\Big)\big(1-\frac{2\delta}{3}\big)f\lambda$ for $|S| \leq \lambda$. By definition, we have $\Sigma_Y(S) \geq W_\beta(\ell) > (1+\epsilon)\Big(\sum_{i=0}^{l-1} \frac{W_\beta(i)}{2^i}\Big)\big(1-\frac{2\delta}{3}\big)f\lambda$. Hence, no such S exists.
Case II: $|S| > \lambda$.
Consider the set of rounds $S \setminus \{r\}$ where r is the round during which b_ℓ is mined. We have

$$\Sigma_Y(S \setminus \{r\}) = \Sigma_Y(S) - W_\beta(\ell),$$

and by definition of $W_\beta(\ell)$, we have

$$\Sigma_Z^{\ell^-}(S \setminus \{r\}) \geq \Sigma_Z^{\ell^-}(S) - (1+\epsilon)\Big(\sum_{i=0}^{\ell-2} \frac{W_\beta(i)}{2^i}\Big)\big(1-\frac{\delta}{3}\big)f\lambda.$$

Now, from case (c) of Lemma VI.1, we get

$$\Sigma_Y(S \setminus \{r\}) \geq \Sigma_Z^{\ell^-}(S \setminus \{r\}) \qquad (\text{since } |S|-1 \geq \lambda)$$

$$\Sigma_Y(S) - W_\beta(\ell) \geq \Sigma_Z^{\ell^-}(S) - (1+\epsilon)\Big(\sum_{i=0}^{\ell-2} \frac{W_\beta(i)}{2^i}\Big)\big(1-\frac{\delta}{3}\big)f\lambda.$$

Replacing $W_\beta(\ell)$ by its value, we get

$$\begin{aligned}\Sigma_Y(S) &\geq \Sigma_Z^{\ell^-}(S) + (1+\epsilon)\Big(\sum_{i=0}^{\ell-1} \frac{W_\beta(i)}{2^i}\Big)\big(1-\frac{\delta}{3}\big)f\lambda \\ &\quad - (1+\epsilon)\Big(\sum_{i=0}^{\ell-2} \frac{W_\beta(i)}{2^i}\Big)\big(1-\frac{\delta}{3}\big)f\lambda \\ &\geq \Sigma_Z^{\ell^-}(S) + \frac{W_\beta(l-1)}{2^{l-1}}(1+\epsilon)\big(1-\frac{\delta}{3}\big)f\lambda.\end{aligned}$$

Thus, we cannot find a set S of rounds such that $\Sigma_Y(S) < \Sigma_Z^{\ell^-}(S)$. This completes the proof of the theorem. □

B. Safety Property

The common prefix property states that once a block b has been inserted more than k blocks deep into the blockchain, every honest blockchain will contain block b in its chain.

Lemma VIII.8 (ℓ-diamond block common-prefix)**.** *Assume $t < \big(\frac{1}{2}-\delta\big)N$ with $\delta > 3(\epsilon+f)$ and a typical execution. Suppose that at round r of the typical execution an honest party receives two chains, $\mathcal{C}$ and $\mathcal{C}'$ where $\mathcal{C}' \setminus (\mathcal{C} \cap \mathcal{C}')$ has at least $k = \lambda f$ blocks at level $\ell \geq \beta$, then $\mathcal{C}$ has more ℓ-diamond blocks than $\mathcal{C}'$ has.*

Proof. By Theorem VIII.7, if the honest blockchain $\mathcal{C}$ contains a ℓ-diamond block, the adversary cannot fork the ℓ-diamond block with a chain $\mathcal{C}'$ that does not contain a ℓ-diamond block in a typical execution. Let us assume by contradiction that C' has more ℓ-diamond blocks than $\mathcal{C}$ has. This implies that $Y_\ell(S) > Z_\ell(S)$ for some set of rounds S, which is not possible in a typical execution by Lemma VIII.3. This contradicts the assumption and completes the proof. □

We can now devise the transaction acceptance rule as follows:

Rule VIII.9 (Transaction Acceptance Rule)**.** *In Gems, one can safely accept a transaction once it has been validated by k ℓ-diamond blocks, with $k \geq \lambda f$.*

Soundness of the transaction acceptance rule is given by Theorem VIII.10, whose proof appears in the full version of the paper [8].

Theorem VIII.10. *Consider an arbitrary 1/2-bounded PPT adversary in a typical execution. Let Π be a proof generated by an honest party at round r using Algorithm 2 with chain $\mathcal{C}$ as parameter. Let Π' be an arbitrary proof generated by the adversary at round r. Let Π^* be the proof accepted by an honest party using Algorithm* **??**. *Then $|\Pi^* \{(\Pi^* \cap \mathcal{C})[-1]:\}| \geq |\mathcal{C}\{(\Pi^* \cap \mathcal{C})[-1]:\}|$ with overwhelming probability.*

C. Liveness property

The Liveness property states that it does not take more than u rounds to insert transactions of honest parties in the blockchain.

Theorem VIII.11. *If all honest parties try to insert a transaction in a blockchain for u consecutive rounds, the transaction shall be accepted by any honest party by the end of the last round of the set of rounds u with probability at least $1 - e^{-\Omega(\beta u)}$.*

Proof. The theorem can be easily proved using the Chain Growth Lemma from [7] which states that the longest chain will have at least $(1-\epsilon)f|S|$ blocks in $|S|$ rounds and using Chernoff bounds to bound the probability of obtaining ℓ-diamond blocks. Once a ℓ-diamond block is mined in the u rounds, the transaction is accepted by every honest party since with a probability at least $1-2^{-\beta}$ either it is inserted in the ℓ-diamond block itself or a block preceding the ℓ-diamond block. Therefore, the probability of the transaction being accepted by all honest parties is at least $\left(1-\left(\frac{f}{2^\beta}\right)^u\right)(1-2^{-\beta}) = 1+\left(\frac{f}{2^{2\beta}}\right)^u - 2^{-\beta} - \left(\frac{f}{2^\beta}\right)^u$. For decently large β (i.e., $\beta \geq 10$), $2^{-\beta} \to 0$ and $\left(\frac{f}{2^{2\cdot\beta}}\right)^u \to 0$ as u increases and the desired probability $\in 1 - e^{-\Omega(\beta u)}$ □

IX. Limits of Blockchain Compression

We present a lower bound on the storage and communication complexity for any PoW-based blockchain protocol. Unlike the proofs presented in the previous sections, this proof is based on information theory and operates independently of the previously described communication and adversary

model. Furthermore, to derive a result for any general PoW-based blockchain protocol, we consider generalised versions of blockchains, that is *block-DAG*, in which blocks are arranged in the form of a Directed Acyclic Graph (DAG). Let $\mathcal{C}$ a block-DAG. $\mathcal{C}$ is said to be valid if $\forall b \in \mathcal{C}$, predecessors of b also lie in $\mathcal{C}$ except for the genesis block that has no predecessors. Additionally, there is at least one Byzantine adversary in the system along with at least two honest parties. We provide the parties with the following abstract functionalities:

- *Topological Sort* $\mathsf{S}(\mathcal{C})$. The topological sort functionality takes in a valid block-DAG $\mathcal{C}$ and returns a list L of blocks in $\mathcal{C}$ such that $\forall i \in [n]$, $\mathcal{C} \setminus L[1:i]$ is also a valid block-DAG, where n is the number of blocks in $\mathcal{C}$.
- *PoPoW* $\mathsf{P}(\mathcal{C})$. The Proof-of-Proof-of-Work functionality takes in a valid block-DAG $\mathcal{C}$ and returns a PoPoW $\mathsf{P}(\mathcal{C})$ such that $\mathsf{P}(\mathcal{C}) \neq \mathsf{P}(\mathcal{C}')$ for all $\mathcal{C}' \subsetneq \mathcal{C}$.
- *Compress* $\mathsf{C}(\mathcal{C})$. The Compress functionality takes in a block-DAG $\mathcal{C}$ and returns an output of at most ℓ bits.
- *Equivalent PoPoW* $\mathsf{U}(\mathsf{C}(\mathcal{C}))$. This functionality produces a PoPoW from a compressed block-DAG such that $\mathsf{U}(\mathsf{C}(\mathcal{C})) = \mathsf{P}(\mathcal{C})$.

A. Communication Complexity

The communication complexity refers to the number of bits required to transmit a PoPoW to an uninitiated party. We start by showing a property of topological sort that claims that subsets of the block-DAG obtained by removing the blocks from the end can be nested like a russian doll.

Lemma IX.1. $\mathcal{C} \setminus L[1:i] \subsetneq \mathcal{C} \setminus L[1:j]\ \forall j < i$

Proof. $(\mathcal{C} \setminus L[1:j]) \setminus (\mathcal{C} \setminus L[1:i]) = L[j:i]$ □

Based on this preliminary lemma, theorem IX.2 proves the lower bound on the communication complexity.

Theorem IX.2. *The PoPoW* $\mathsf{P}(\mathcal{C})$ *must contain at least* $\lfloor \log_2(n) \rfloor$ *bits where* n *is the number of blocks in* $\mathcal{B}$.

Proof. Let us assume by contradiction that there exists a PoPoW P that produces an output of at most l bits, where $l < \lfloor \log_2(n) \rfloor$. Consider an adversary that obtains an ordering of blocks L via the functionality S and produces n PoPoWs $P_1, P_2, \ldots, P_n$ corresponding to $\mathcal{C} \setminus L[1:i]\ \forall i \in [n]$. From Lemma IX.1,

$$(\mathcal{C} \setminus L[1:n]) \subsetneq (\mathcal{C} \setminus L[1:n-1]) \subsetneq \ldots \subsetneq (\mathcal{C} \setminus L[1:1])$$

By definition, $P_i \neq P_j\ \forall i \neq j$, therefore each PoPoW P_i needs to be mapped to a distinct sequence of l bits. However, there are only $2^l < n$ sequences possible. Therefore, by the Pigeonhole Principle at least two P_i, P_j must be mapped to the same sequence of bits. A contradiction. Thus our assumption that there exists a P that produces an output of at most l bits, where $l < \lfloor \log_2(n) \rfloor$ is incorrect. □

B. Storage Complexity

The storage complexity refers to number of bits required to store a block-DAG $\mathcal{C}$ such that a party can transmit a valid PoPoW $P = \mathsf{P}(\mathcal{C})$ to another party.

Lemma IX.3. *For any two block-DAG* $\mathcal{C}_i$ *and* $\mathcal{C}_j$ *such that* $\mathcal{C}_i \subsetneq \mathcal{C}_j$*, then* $\mathsf{C}(\mathcal{C}_i) \neq \mathsf{C}(\mathcal{C}_j)$.

Proof. Let us assume by contradiction that there are two parties that maintain $\mathcal{C}_i$ and $\mathcal{C}_j$ such that $\mathcal{C}_i \subsetneq \mathcal{C}_j$ and $\mathsf{C}(\mathcal{C}_i) = \mathsf{C}(\mathcal{C}_j) = s$. By definition of PoPoW,

$$\mathsf{P}(\mathcal{C}_i) = \mathsf{U}(s) = \mathsf{U}(s) = \mathsf{P}(\mathcal{C}_j).$$

We reach our contradiction, which completes the proof. □

Theorem IX.4. *A party that can transmit a valid PoPoW for a block-DAG* $\mathcal{C}$ *must store at least* $\lfloor \log_2(n) \rfloor$ *bits where the* n *is the number of blocks in* $\mathcal{C}$.

Proof. Let us assume by contradiction that the Compress functionality allows a party to store the block-DAG $\mathcal{C}$ with solely ℓ bits, with $\ell < \lfloor \log_2(n) \rfloor$. Consider the list L produced with the topological sort functionality. From Lemma IX.1, $(\mathcal{C} \setminus L[1:n]) \subsetneq (\mathcal{C} \setminus L[1:n-1]) \subsetneq \ldots \subsetneq (\mathcal{C} \setminus L[1:1])$. From Lemma IX.3, we know that $\forall \mathcal{C}' \subsetneq \mathcal{C}$, $\mathsf{C}(\mathcal{C}') \neq \mathsf{C}(\mathcal{C})$. There are at least n subsets of $\mathcal{C}$ that must be mapped to a unique sequence of l-bits. However, there are only $2^l < n$ sequences in the codomain of the compression functionality C. By the Pigeonhole Principle, at least two subsets must be mapped to the same output. A contradiction that completes the proof. □

X. Discussion and Future Work

Bünz et al. [5] present a bribing attack against superblocks used in NIPoPoWs in a model with the rational majority. We leave it to future work to determine an appropriate reward scheme to make the blockchain protocol incentive compatible.

Superblock-based NIPoPoWs have only been shown to work in a setting with constant difficulty. We leave it to future work to propose a new design of NIPoPoWs that works in a setting with variable difficulty. The challenge is to properly handle decreasing difficulties as an adversary might propose a slightly more difficult adversarial proof than what should be the honest one. A possible defense might require to increase the footprint of the proof during such periods of decreasing difficulty. We forecast that new blockchains will adopt our construction, while existing ones, such as Bitcoin, can also benefit our construction to allow miners to operate in $O(\mathrm{poly}\log(n))$ storage and communication complexity with theoretically proven security guarantees. In the latter case this would require as suggested by Kiayias et al. [12] a velvet fork, i.e., an upgrade that does not require any modification to the consensus layer, but only a different interpretation of the data [21]. For instance, the interlink data structure could appear in the coinbase data of newly created blocks.

Finally, our $\lceil \log(n) \rceil$ lower bound on storage and communication complexities show that no optimal blockchain compression scheme has been discovered yet. We leave it for future work.

REFERENCES

[1] I. Abraham and D. Malkhi. The blockchain consensus layer and bft. *Bulletin of the European Association for Theoretical Computer Science*, (123), 2017.

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

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

[4] J. Bonneau, I. Meckler, V. Rao, and E. Shapiro. Coda: Decentralized cryptocurrency at scale. https://eprint.iacr.org/2020/352.pdf, 2020.

[5] B. Bünz, L. Kiffer, L. Luu, and M. Zamani. Flyclient: Super-light clients for cryptocurrencies. In *IEEE Symposium on Security and Privacy (S&P)*, 2020.

[6] A. Chepurnoy, S. Srinivasan, and Y. Zhang. EDRAX: A cryptocurrency with stateless transaction validation. https://eprint.iacr.org/2018/968.pdf, 2018.

[7] J. Garay, A. Kiayias, and N. Leonardos. The bitcoin backbone protocol: Analysis and applications. In E. Oswald and M. Fischlin, editors, *Advances in Cryptology - EUROCRYPT 2015*, pages 281–310, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.

[8] A. Jain, E. Anceaume, and S. Gujar. Extending The Boundaries and Exploring The Limits Of Blockchain Compression. https://cnrs.hal.science/hal-03866741, 2023. Full version.

[9] K. Karantias, A. Kiayias, and D. Zindros. Compact storage of superblocks for nipopow applications. In *Mathematical Research for Blockchain Economy*, pages 77–91. Springer, 2020.

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

[11] A. Kiayias, N. Lamprou, and A.-P. Stouka. Proofs of proofs of work with sublinear complexity. In J. Clark, S. Meiklejohn, Peter Y.A. R., D. Wallach, M. Brenner, and K. Rohloff, editors, *Financial Cryptography and Data Security*. Springer Berlin Heidelberg, 2016.

[12] A. Kiayias, N. Leonardos, and D. Zindros. Mining in logarithmic space. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security (CCS)*, page 3487–3501, New York, NY, USA, 2021. Association for Computing Machinery.

[13] A. Kiayias and O. S. Thyfronitis Litos. A composable security treatment of the lightning network. In *IEEE Computer Security Foundations Symposium (CSF)*, 2020.

[14] A. Kiayias, A. Miller, and D. Zindros. Non-interactive proofs of proof-of-work. In J. Bonneau and N. Heninger, editors, *Financial Cryptography and Data Security*, pages 505–522, Cham, 2020. Springer International Publishing.

[15] R. Matzutt, B. Kalde, J. Pennekamp, A. Drichel, M. Henze, and K. Wehrle. Shrinking bitcoin's blockchain retrospectively. *IEEE Transactions on Network and Service Management*, 18(3), 2021.

[16] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system. http://www.bitcoin.org/bitcoin.pdf, 2009.

[17] J. Nick, A. Poelstra, and G. Sanders. Liquid: A bitcoin sidechain. https://blockstream.com/assets/downloads/pdf/liquid-whitepaper.pdf, 2020.

[18] J. Poon and T. Dryja. The bitcoin lightning network: Scalable off-chain instant payments. https://lightning.network/lightning-network-paper.pdf, 2016.

[19] SNAP. Ethereum snapshot protocol. https://github.com/ethereum/devp2p/blob/master/caps/snap.md, 2020.

[20] R. Wattenhofer. *The Science of the Blockchain*. CreateSpace Independent Publishing Platform, North Charleston, SC, USA, 1st edition, 2016.

[21] A. Zamyatin, N. Stifter, A. Judmayer, P. Schindler, E. Weippl, and W.J. Knottenbelt. A wild velvet fork appears! inclusive blockchain protocol changes in practice. In *Financial Cryptography and Data Security*, 2019.