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
anurag.jain@research.iiit.ac.in

Emmanuelle Anceaume
CNRS, IRISA, Rennes, France
emmanuelle.anceaume@irisa.fr

Sujit Gujar
International Institute of Information Technology (IIIT), Hyderabad, India
sujit.gujar@iiit.ac.in

***Abstract*—Blockchain technology aims to replace traditional banking systems and manage the world's economic data. However, 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 a blockchain protocol based on superblock Non-Interactive-Proofs-of-Proof-of-Work (NIPoPoWs) as a mechanism to reduce the storage and communication complexity of blockchains to O(polylog(n)). However, their protocol is only resilient to an adversary that may control strictly less than 1/3rd of the total computational power, which is a reduction from the security guaranteed by Bitcoin and other existing blockchain protocols that guarantee security against an adversary that may control strictly less than 1/2 of the total computational power. We present an improvement to the Kiayias et al. proposal termed Gems-scheme, which is resilient against an adversary that may control less than 1/2 of the total computational power while operating in O(polylog(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; Cryptocurrencies; Proof-of-work; NIPoPoWs**

1. 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. Enormous amounts of financial data are being generated continuously due to the fast speed of commerce on a global scale. 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 party to join the system as it would need to download the entire blockchain from the network in order to start mining[1]. For the blockchains to be adopted on a global scale, optimization is crucial to reduce the storage and communication requirements and securely prune the data.

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, i.e., UTXO (Unspent Transaction Output), account data, smart contract states, etc. while the *Consensus Data* 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 party that joins the blockchain system must synchronize with both types of data in order to initiate mining. In a *Proof-of-Work* 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 have been well deployed in practice. For instance, Ethereum uses snapshots that periodically summarize all active data, discarding inactive data that has been overwritten or mutated. However, compressing consensus data is still an area of active research since the consensus data in existing protocols increases linearly with time.

Kiayias et al. [1] propose an elegant scheme to construct a blockchain protocol using *Non-Interactive Proofs-of-Proof-of-Works (NIPoPoWs)* that operates in $O(\mathrm{poly}\log(n))$ storage complexity and $O(\mathrm{poly}\log(n))$ communication

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 only ones that do not need to download the full blockchain are Simplified Payment Verification (SPV) parties, but such parties cannot check the security of the blockchain.

complexity, i.e., 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 [1] 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 blockchain protocols such as Bitcoin and Ethereum. However, the proposal can only guarantee tolerance against a Byzantine adversary that may control strictly less than 1/3rd of the total computational power, reducing security from the original Bitcoin protocol and other blockchain protocols. In this work, we address this issue and present Gems, which improves the Kiayias et al.'s blockchain protocol, that can provably achieve security against a Byzantine adversary that controls strictly less than 1/2 of the total computational power.

Kiayias et al.'s proposal [1] uses NIPoPoWs that rely on the probability distribution of the hash values of blocks in the blockchain and uses this *a priori* to sub-sample the blocks and assume the same distribution of hash values as the original blockchain. However, this creates a security threat where the adversary may skew the sub-sample by selectively "suppressing" superblocks; superblock of level ℓ is a block whose hash is ($2^{-\ell}$·target hash). Kiayias et al. illustrate this weakness in Lemma 6.7 and 6.8 in [1] where the authors determine the optimal attack strategy of the attacker to suppress the superblocks from the sub-sample. We solve this issue by attaching weights to the superblocks at above certain threshold level, which we call *diamond blocks*. The diamond blocks have higher preference in chain selection rule. With the modified chain selection rule, it is improbable for an attacker controlling less than 1/2 of the total hashing power to suppress the superblocks as shown in Section 8. We thereby improve the security of Kiayias et al.'s scheme to tolerate a Byzantine adversary that may control up to 1/2 of the total hashing power.

The only major limitation of both, our solution, Gems and Kiayias et al.'s approach is that they are only proven to operate securely in a setting with constant difficulty. However, we remark that there has not been any work in the literature that tackles the problem of blockchain compression in a setting with variable difficulty. Bünz et al. [2] present a scheme for verifying the blockchain by an SPV client in sublinear space and time in a setting with variable difficulty. Our solution and Kiayias et al.'s solution are the only solutions that achieve mining in logspace.

Our scheme that achieves optimal security while still operating in $O(\text{poly}\log(n))$ storage and communication complexity leaves the open question of whether such a scheme is optimal or if there is further room for improvement. We progress this question by presenting a novel proof using information theory to establish a lower bound on the limit of blockchain compression. Therefore, we righteously claim that a PoW-based blockchain protocol cannot operate in less than $O(\log(n))$ storage and communication complexity.

Contributions of this work In summary, the contributions of this work are as follows:

- We propose Gems, a NIPoPoW protocol derived from Kiayias et al. [1].
- We propose a novel weight assignment scheme to superblocks for improving security guarantees in the original proposal. (Section 7.3)
- We prove that Gems tolerates a Byzantine adversary that may control up to 1/2 of the total hashing power (Section 8)
- For the first time, we provide proofs of boundaries of blockchain compression by showing that it is not possible to compress a PoW blockchain protocol beyond $O(\log(n))$ storage and communication complexity (Theorems 9.2 and 9.4, Section 9).

Organization of the paper Section 2 is dedicated to related work, and Section 3 presents the assumptions on the system in terms of communication and power of the adversary. Section 4 provides some minimal background on permissionless blockchains, and non-interactive proofs-of-proof-of-works (NIPoPoWs). Section 5 presents the main lines of the Kiayias et al. to compress PoW-blockchains with NIPoPoWs. Section 6 describes the main components of Gems, our solution towards improving the security of Kiayias et al.'s proposal. Section 7 presents the weight intuition of Gems. Section 8 analyzes the security of our solution, Gems. In Section 9, we answer the open question raised by Kiayias et al. by presenting a definitive lower bound on the storage space and communication complexities for any PoW-based blockchain protocol. Section 10 concludes and presents some future work.

2. Related Work

The problem of blockchain becoming of considerable size was initially predicted by Satoshi Nakomoto himself in his original paper that introduced Bitcoin [3]. He offered a simple solution of a *Simplified Payment Verification (SPV)* that only requires a client to store the block headers to verify the transactions.

Kiayias et al. [4] introduced and formalized an interactive proof mechanism, *Proofs-of-Proof-of-Work (PoPoW)* based on superblocks that allowed a client to verify a chain in sublinear time and communication complexity. However, the authors later showed the existence of an attack on the scheme and proposed a non-interactive yet polylogarithmic alternative, *Non-Interactive Proofs-of-Proof-of-Work (NIPoPoWs)* [1]. However, the solution did not address the size of the blockchain that needed to be stored by any miner. Kiayias et al. further used *NIPoPoWs* to develop a scheme that also allowed 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 1/3 of the total computation power and limiting itself to operate in an environment with a fixed difficulty [1]. In this work, we build upon their solution to present a scheme with improved security.

All these techniques based on NIPoPoWs are complementary to sharding techniques. Briefly, sharding aims at processing transactions in parallel to scale the system horizontally. Nodes (either correct or Byzantine) are allocated into different shards, and each shard processes transactions concurrently. Nodes in each shard maintain their blockchain containing transactions submitted from users of the sharded blockchain. A transaction may involve a single shard (intra-shard transaction) or multiple shards (cross-shard transaction). To process cross-shard transactions, nodes in different shards may communicate with each other. For the last few years, sharding techniques have received much attention, and among the different existing propositions, we can cite the following ones. Elastico [5] is a PoW permissionless blockchain that combines both network and transaction-sharding to scale transaction rates almost linearly with available computational power. Omniledger [6] and Rapidchain [7] are among the first solutions that improve blockchain performance by implementing state-sharding, while Stakecube [8] introduces the notion of adaptability of the number of shards to the current number of parties in the system. More recently, some deployed solutions such as TON [9] and Elrond [10] propose to support smart contracts, while Monoxide [11] or Brokerchain [12] propose to reduce the number of cross-shard transactions by organizing parties as a function of their transactional relationships. To cope with the hardness of the problem, BrokerChain [12] proposes a state-graph partitioning algorithm, executed by a single shard responsible for the reorganization of the parties within the shards.

3. Model of the System

In this section, we formally describe the model used in the first half of the paper. We discuss our proposal to improve the security of Kiayias et al.'s superblock-based NIPoPoW scheme. Our proof that provides the lower bound on the storage and communication complexity of a PoW-based blockchain protocol (see Section 9) is based on information theory and operates independently of the model described in this section.

3.1. Communication Model

In this work, we consider a static setting where the parties operate in a *synchronous* network. This means that a fixed upper bound Δ on the time it takes for a message to be transmitted between any two parties and a fixed upper bound Φ on the time that elapses between two consecutive steps of a party exist and are known by all the parties of the system. This model allows us to organize the execution of a distributed algorithm in rounds such that in each round, every party can send a message to each of its neighbours, receive the messages sent during the round, and execute a computational step based on the received messages.

Symbol	Description
N	Number of parties in the system
t	Number of parties controlled by the adversary
T	Target for the hash value of a valid block
n	Total number of blocks appended in the execution of a blockchain protocol
k	Common prefix parameter
δ	Size of the application data in a block
a	Size of the application data in a blockchain
ℓ	Level of a superblock
χ	Unstable portion of the blockchain
Π	NIPoPoW / Compressed Chain
f	Probability that at least one honest party succeeds in finding a PoW in one round
λ	Length of the smallest execution considered to be typical
m	Security parameter of the compression scheme
$\mathcal{D}$	Set of superblocks in the compressed chain
$W(\cdot)$	Weight assignment for blocks
β	Parameter associated with the weight assignment
$\Sigma(\cdot)$	Returns the sum of weights of blocks
$Y(S)$	Random variable denoting the number of blocks appended by the honest parties in a set of rounds S
$Z(S)$	Random variable denoting the number of blocks obtained by the adversary in a set of rounds S

Table 1: List of symbols used in the paper

3.2. Adversary Model

We assume the presence of a Byzantine or malicious adversary that may control strictly less than 1/2 of the total amount of computational power currently available in the system. This model, named the "Computational Threshold Adversary" [13], is an alternative to the Common Threshold Adversary Model, which bounds the total number of parties the adversary controls relative to the total population of the system. In this work, we limit the adversary to a probabilistic polynomial-time Turing machine that behaves arbitrarily, i.e., at any time, it may follow or not follow the prescribed protocol. In particular, the adversary can broadcast arbitrary messages to parties and may send different messages to different parties. However, the adversary 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.

4. Background

A *blockchain* is a data structure that enables coordination or consensus in a distributed system. In this paper, we consider *Proof-of-Work* based blockchains which achieve consensus without relying on a trusted party by requesting the parties to contribute a limited resource such as hashing power. We use the term "consensus" to refer to a weaker form of consensus that is achieved by a PoW blockchain wherein the network may reach consensus eventually, i.e., if no new updates are issued, the system reaches a quiescent state where the shared state is consistent [14].

4.1. Proof-of-Work

The Proof-of-Work or PoW scheme requires each party to generate a "proof" of investing a limited resource such as hashing power that takes time to generate but can be quickly verified by other parties. PoW enables consensus in a *permissionless* system in which any party can join or exit at any time without requiring permission from any other party. This form of permissionless consensus is provably secure against a Byzantine probabilistic polynomial-time adversary that may control strictly less than 1/2 of the total hashing power in a synchronous system [15].

In PoW-based blockchain, every party that wants to insert a block into blockchain is required to provide a *nonce* along with the contents of the block, hashes to a value below a given target. The hash function $\mathcal{H}$ is modelled as a random oracle that produces constant length output. Since the distribution of hash values is stochastic, some blocks end up with hash values significantly below the target. In particular, blocks that hash to a value less than $T/(2^\ell)$, where T is the target value, are called ℓ-*superblocks* [16], [4], [17], [1]. Note that every ℓ-superblock is also a ℓ'-superblock for any $\ell' \leq \ell$ and the genesis block is considered to have a hash value of $0\text{x}00\ldots0$ and hence, is a superblock of the highest level.

In a PoW-based blockchain protocol like Bitcoin, every honest party tries to extend the longest chain. These parties are rewarded only when their blocks are accepted into the longest chain, providing the necessary incentives to extend the longest chain.

4.2. Bitcoin

Bitcoin builds a *robust* blockchain using the PoW scheme that enables storing transactions required for the activity of a secure cryptocurrency. A robust blockchain protocol must ensure that the following properties with overwhelming probability [15]:

1) *Safety* with parameter $k \in \mathbb{N}$: If a valid transaction block appears in a block that is at least k blocks away from the end of the blockchain of an honest party, then this transaction will appear at the same position at all honest parties' blockchain.
2) *Liveness* with parameters $k \in \mathbb{N}$: if a valid transaction is received by all honest parties, then this transaction will appear in a block at least k blocks away from the end of the blockchain of all honest parties.

4.3. Application and Consensus Data

In a blockchain system, each party must independently maintain a copy of the blockchain in order to mine new blocks and verify transactions. The data stored by a party can be divided into two types:

Definition 4.1 (Application Data)**.** The application data consists of all information required to verify the validity of a transaction, i.e., the unspent transaction outputs (UTXOs), account data, smart contract states, etc.

Definition 4.2 (Consensus Data)**.** The consensus data consists of all information required apart from the application data for the complete participation of a party. In a PoW-based blockchain, this involves all the block headers in the chain.

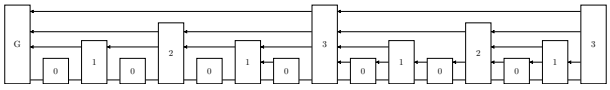
A new honest party that joins the blockchain system must synchronise with both types of data to initiate the mining process. In a PoW-based blockchain protocol, the consensus data grows linearly with every new appended block. This causes issues with both storing the consensus data and communicating the full blockchain when bootstrapping a new party. To address these issues, *Non-Interactive Proofs-of-Proof-of-Works* (NIPoPoW) allows us to store and communicate the PoW in a compressed form.

4.4. Superblock NIPoPoWs

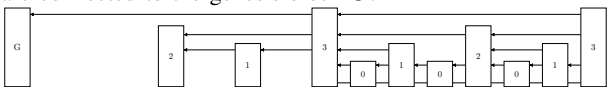
Superblock Non-Interactive Proofs-of-Proof-of-Works (NIPoPoWs) compress the PoW by subsampling the block headers [4]. The working principle behind this compression lies in the assumption that a sub-sample of the block headers can be sufficient to estimate the size of the original distribution of block headers. NIPoPoWs rely on designating some of the blocks in a blockchain as *"superblocks"* [16], [17], [1]. 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. In more detail, in a typical execution of the PoW blockchain, on average, $1/2^\ell$ of the blocks are ℓ-superblocks. NIPoPoW samples the ℓ-superblocks to prove that the original blockchain contained 2^ℓ blocks. In order to convince honest parties, the NIPoPoW contains a constant number of superblocks at each level. The idea behind NIPoPoW is that the security properties associated with the entire blockchain are also associated with superblocks at each level. Hence, the longest chain can be proven with only the superblocks from the blockchain.

The scheme requires every block header to store pointers to the last superblock at every level in order to ensure that the subsampled blocks also form a valid chain. A chain of n blocks will contain superblocks at $O(\log(n))$ levels, as illustrated in Figure 1. Hence, the space and communication complexity of Kiayias et al.'s proposal [4] is $O(\text{poly}\log(n))$. In Section 9, we show that any such compression needs at least $O(\log(n))$ blocks.

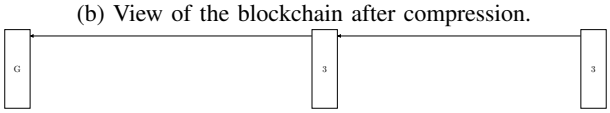
The proposal by Kiayias et al. [1] offers the best-known compression of PoW blockchains so far. It uses NIPoPoWs to present a blockchain protocol that achieves $O(\text{poly}\log(n)c + k\delta + a)$ storage and communication costs while allowing parties to mine new blocks based on this compressed blockchain, where k is the common prefix parameter, δ is the size of application data per block, and a is the size of application data in the blockchain. However, their solution reduces the security of the protocol by guaranteeing resilience to only a 1/3$^{\text{rd}}$ Byzantine adversary. Improving



(a) The probabilistic hierarchical blockchain. Higher levels have achieved a higher difficulty during mining. All blocks are connected to the genesis block G.



(b) View of the blockchain after compression.



(c) View of the same section of blockchain after a period of time upon compression.

Figure 1: Illustration of Kiayias et al.'s [1] compression scheme.

these security guarantees in NIPoPow (specifically Kiayias et al.'s one) is the primary focus of the work. We first briefly discuss Kiayias et al.'s approach.

5. Kiayias et al.'s Scheme

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. Both algorithms rely on the notion of a superblock that has been defined as follows.

Definition 5.1 (ℓ-superblock ([1]))**.** A block that hashes to a value less than $T \cdot 2^{-\ell}$ is said to be a ℓ-superblock.

5.1. 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 sufficient superblocks at every level and discarding the rest of them. These samples evolve with time, i.e., a superblock once selected may be discarded later, but vice-versa does not hold.

The Kiayias et al.'s chain compression algorithm (from [1], Algorithm 1) is parameterized by a security parameter m and the common prefix parameter k. The algorithm compresses the blockchain except for the k most recent unstable blocks. The compression works as follows: For the highest level ℓ that contains more than $2m$ blocks, keep all the blocks but for every level μ below ℓ, only keep the last $2m$ blocks and all the blocks after the m^{th} block at the $\mu + 1$ level. Π is used to represent an instance of NIPoPoW proof.

5.2. 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. Let $\Pi_1, \Pi_2, \ldots, \Pi_n$ be the different compressed blockchains that a new party receives. The party first applies the compression algorithm to every compressed blockchain to make the comparison fair. To compare any two compressed blockchains Π and Π', the compression algorithm selects the minimum level μ that contains a block present in both Π and Π'. If no such block is found, it necessarily implies that the greatest level (compression level ℓ) in the two compressed blockchains is not the same, and thus simply, the algorithm selects the one with the greatest level. If block b is found in both Π and Π' at the same level μ, then the blockchain with the greatest number of blocks after b wins the comparison.

5.3. Succinctness

An informal argument for the succinctness of the protocol follows from the fact that for a blockchain containing n blocks, the number of levels will be in $O(\log(n))$. At each level, the compressed blockchain may contain at most $4m$ blocks with overwhelming probability. Therefore, with overwhelming probability, the number of blocks in the compressed blockchain will be in $O(4m \log(n))$. A detailed formal proof is presented in Section 8.

5.4. Security

The high-level intuition behind the security of Kiayias et al.'s proposal [1] is that the common prefix property which states that any block sufficiently deep in the blockchain is guaranteed to remain in the blockchain held by the honest parties, holds true not only for the whole compressed chain but also at every level of the compressed chain. So the adversary cannot possibly produce a blockchain that forks superblocks at a higher level. So we may safely 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 [1] only holds true if the adversary cannot possibly produce a blockchain 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 would be to "suppress" these superblocks. Lemma 6.7 of [1] shows that an adversary with at least 1/3$^{\text{rd}}$ of the total hashing power could possibly suppress all the superblocks. Hence, their protocol can be proven to be secure only if the adversary is allowed to control less than 1/3$^{\text{rd}}$ of the total hashing power, which is a reduction from the security guaranteed by Bitcoin [15]. In the following section, we describe how we fix this reduction in security to achieve the same security guarantees as the vanilla Bitcoin protocol.

6. Gems Blockchain: Main components

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 Bitcoin blockchain has not been adversarially biased in the compressed chain. Kiayias' proposal uses the default chain selection rule that is oblivious to the presence of superblocks, and thus it is possible for an adversary to "suppress" superblocks by forking the chain, biasing the superblock distribution accordingly. Thus Kiayias et al.'s proposal security heavily depends on the superblocks that the adversary can suppress: if the adversary owns at least 1/3rd of the hashing power, it can possibly suppress all the superblocks mined by honest parties. With Gems we improve the security of their proposal by hindering the suppression of superblocks. We assign different weights to the blocks as per their superblock level so that they are given preference in the chain selection rule. By adding greater preference for superblocks, we improve the security by making it difficult for the adversary to selectively fork the superblocks.

Concretely, and as will be detailed in Section 7.3, we designate a small fraction of the superblocks with level $\ell \geq \beta$ as $\ell-$*diamond blocks* and assign a weight $W(\ell)$ to them such that the probability that an adversary controlling up to $1/2$ of the total hashing power can suppress a ℓ-diamond block becomes negligible. The system parameter β quantifies the trade-off between storage and communication costs and increasing weights. The intuition comes from the fact that an adversary that does not control the majority of the total hashing power cannot produce a blockchain that has more weight than the blockchain containing a ℓ-diamond block without including another ℓ-diamond block. Figure 2 illustrates such a scenario where the adversary cannot impose its own blockchain unless it mines a ℓ-diamond block that counts for a weight equal to $W(\ell)$.

Notation In this work, we borrow the notation and the mathematical framework introduced by Kiayias et al. in [1]. $\mathcal{C}[i]$ 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, $(\mathcal{C}\uparrow^{\mu})\uparrow^{\mu+i} = \mathcal{C}\uparrow^{\mu+i}$.

6.1. Chain Compression Algorithm

We modify the chain compression algorithm of Kiayias et al. [1] to exploit the presence of diamond blocks. Pseudo-code of the algorithm is presented in Algorithm 1. $\mathcal{D}$ refers to the set of superblocks that are retained after compression, indexed by their superblock level. The unstable part of the compressed chain is modified so as to take advantage of diamond blocks (see Line 14). Our modification is minor; however, the security analysis of the same is non-trivial.

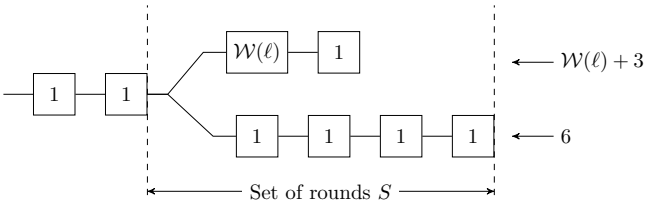


Figure 2: Adversary trying to suppress a superblock by forking the chain

Algorithm 1 Chain compression algorithm for transitioning a full miner to a logspace miner. Given a full chain, it compresses it into logspace state.

1: **function** DISSOLVE $_{m,k}(\mathcal{C})$
2: $\quad\mathcal{C}^* \leftarrow \mathcal{C}[:-k]$
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 0 **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 $_{m,k}(\mathcal{C})$
18: $\quad(\mathcal{D},\ell,\chi) \leftarrow$ Dissolve $_{m,k}(\mathcal{C})$
19: $\quad\pi \leftarrow \bigcup_{\mu=0}^{\ell}\mathcal{D}[\mu]$
20: $\quad$**return** $\pi\chi$
21: **end function**

6.2. Compressed Chain Comparison Algorithm

We formally describe our compressed chain comparison rule: 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. After having performed a syntactic validity check (Lines 1-7) on two compressed chains Π_i and Π_j, first, separate the last k 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. We describe the pseudo-code of our chain selection algorithm in Algorithm 2. 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).

Algorithm 2 The compressed chain comparison algorithm.

```
1:  function maxvalid_{m,k}(Π, Π')
2:      if Π is not valid then
3:          return Π'
4:      end if
5:      if Π' is not valid then
6:          return Π
7:      end if
8:      (χ, ℓ, D) ← Dissolve_{m,k}(Π)
9:      (χ', ℓ', D') ← Dissolve_{m,k}(Π')
10:     M ← {μ ∈ ℕ : D[μ] ∩ D'[μ] ≠ ∅}
11:     if M = ∅ then
12:         if ℓ' > ℓ then
13:             return Π'
14:         end if
15:         return Π
16:     end if
17:     μ ← max(min M, β)
18:     b ← (D[μ] ∩ D'[μ])[−1]
19:     if |D'[μ]{b :}| > |D[μ]{b :}| then
20:         return Π'
21:     else if |D'[μ]{b :}| < |D[μ]{b :}| then
22:         return Π
23:     else if Σ(χ') > Σ(χ) then
24:         return Π'
25:     else
26:         return Π
27:     end if
28: end function
```

7. Gems: Weight intuition

In this section, we lay down the pre-requisites required for analysing the safety and liveness guarantees of Gems. Prior to presenting our proofs, we briefly describe the Bitcoin Backbone Protocol [15] that provides the framework for both Kiayias et al.'s analysis [1] and Gems' one.

7.1. The Bitcoin Backbone Protocol Analysis

The Bitcoin Backbone Protocol analysis by Garay et al. [15] provides bounds for the Safety and Liveness Properties of the Bitcoin Protocol (see Section 4.2) by modelling it as a synchronous system along with introducing useful mathematical tools to develop additional proofs for the blockchain.

Garay et al. [15] propose a model in which each party is allowed to make q queries to a cryptographic hash function in every round and the parties can communicate via broadcast messages. 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.

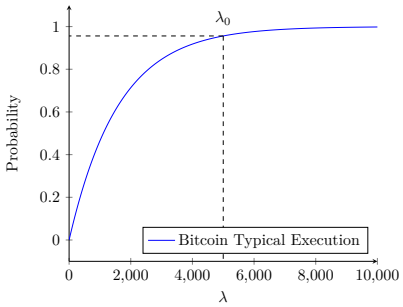


Figure 3: Probability of typical execution with respect to λ

7.1.1. 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 for a short period. However, since in Bitcoin, the adversary controls less than 1/2 of the hashing power, in the long run, the honest parties should be able to mine more blocks than the adversary. So they consider executions with at least λ consecutive rounds so that during such long enough executions, the honest parties mine more blocks than the adversary with a probability of at least $1 - e^{-\Omega(\epsilon^2 \lambda f)}$, where λ satisfies $\lambda \geq 2/f$, f being the probability that at least one honest party succeeds in finding a PoW in one round [15], and $\epsilon \in (0, 1)$. Figure 3 shows the value $\lambda_0 = 5000$ where the probability of typical execution is strictly larger than 0.95. For any set S of rounds, let $Y(S)$ and $Z(S)$ be two random variables defined as follows:

- $Y(S)$ represents the number of rounds in S in which the honest parties produced exactly one block.
- $Z(S)$ represents the number of blocks produced by the adversary in the set of rounds S.

Additionally, δ is the advantage of honest parties, and we have $\delta \leq 1 - t/(N - t)$, where N represents the number of mining parties, out of which t are controlled by the adversary. Both δ and f relate as $3(f + \epsilon) < \delta < 1$.

Lemma 7.1 ([15], Lemma 11)**.** *The following holds for any set S of at least λ consecutive rounds in a typical execution.*

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

Theorem 7.2 ([15], Theorem 10)**.** *A typical execution guarantees that for any S such that $|S| \geq \lambda$, $Z(S) < \left(1 - \frac{2\delta}{3}\right)f|S|$.*

7.2. Warm-up: Assigning common weight to superblocks

This section shows 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 specific to each level of superblocks.

Lemma 7.3 provides an upper-bound on the number of blocks mined by the adversary in a non-typical execution.

Lemma 7.3. *For any set S of consecutive rounds such that $|S| \leq \lambda$, $Z(S) < \big(1 - \frac{2\delta}{3}\big) f\lambda$.*

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 $Z(S) \geq \big(1 - \frac{2\delta}{3}\big) f\lambda$.

By Lemma 7.1, 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 7.1 we have for any set of consecutive rounds S' with $|S'| \geq \lambda$, $Z(S') < \big(1 - \frac{2\delta}{3}\big) f|S'|$. Hence, execution U cannot be a typical execution, which contradicts the assumption, and completes the proof of the lemma. □

Lemma 7.4. *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 $Y(\{r\}) \leq 1$. □

Lemma 7.5 provides an upper bound on the number of blocks the adversary can successfully mined in addition to the ones mined by the honest parties.

Lemma 7.5. *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. Let us consider two cases:
Case I: $|S| < \lambda$.
From Lemma 7.3, $Z(S) < \big(1 - \frac{2\delta}{3}\big) f\lambda$. By definition, $Y(S) \geq 0$. Hence, $\big(Z(S) - Y(S \setminus \{r\})\big) < \big(1 - \frac{\delta}{3}\big) f\lambda$.
Case II: $|S| \geq \lambda$.
From the conditions of typical executions we know that for $|S| \geq \lambda$, $Z(S) < \big(1 - \frac{\delta}{3}\big) f|S|$ and by Lemmas 7.4 and 7.1, we have $Y(S \setminus \{r\}) \geq \big(1 - \frac{\delta}{3}\big) f|S| - 1$. Thus,

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

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 $\big(1 - \frac{\delta}{3}\big) f\lambda$. This completes the proof. □

From Lemma 7.5, we can infer the minimum weight value blocks must be assigned to so that the adversary will not possibly be able to suppress them. Our security measure uses the upper-bound so obtained to assign weight to the super-blocks that exceeds the upper-bound making it improbable for the adversary to suppress these blocks. Furthermore, by adopting a similar approach as the one of Kiayias et al. [1], we generalize our results by using the notion of block property satisfying a predicate Q on its hash output $h \in \{0,1\}^k$. A Q-block is a block whose hash h satisfies predicate Q, i.e., $Q(h)$ = true. Probability ξ_Q quantifies the probability that a block satisfies predicate Q, i.e., $\xi_Q = \mathbb{P}\{Q(h) | h \leq T\}$, where T is the mining difficulty.

Theorem 7.6 (Q-blocks are unsuppressible)**.** *If a Q-block is attached a weight w with $w \geq \big(1 - \frac{2\delta}{3}\big) f\lambda$, then for any blockchain $\mathcal{C}$ adopted by an honest party such that $\mathcal{C}$ contains a Q-block b satisfying predicate Q, 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 Q-block b'.*

Proof. Any block mined by the honest parties has a probability ξ_Q to satisfy predicate Q. Let w be the weight attached to any such Q-blocks. 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 Q-block b but $\mathcal{C}'$ does not contain any Q-blocks. The adversary can suppress Q-block b mined in round r if for any S the following holds.

$$w + Y(S \setminus \{r\}) \leq Z(S)(1 - \xi_Q). \tag{1}$$

So, if we want to prevent Q-block b from being suppressed, we must have that

$$w \geq \max_{\forall\ S} \Big(Z(S) - Y(S \setminus \{r\})\Big). \tag{2}$$

From Lemma 7.5, we get

$$w \geq \big(1 - \frac{2\delta}{3}\big) f\lambda, \tag{3}$$

which completes the proof of the theorem. □

Notice that assigning all the Q-blocks with a unique weight w allows us to secure a given set of blocks. In Section 7.3 we propose a *block weight assignment* for each ℓ-superblock level, $\forall\ \ell \geq 0$.

7.3. Gems' Block Weight Assignment

The superblock-based NIPoPoW scheme involves different levels of superblocks capturing their underlying distribution. The block weight assignment policy assigns unitary weights to the large majority of superblocks, and non-unitary ones to a small fraction of superblocks that we termed ℓ-diamond blocks. We propose the following weight assignment with the system parameter β assumed to be known by all the parties in advance.

Definition 7.1. Gems' blocks weight assignment

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

Definition 7.2 (ℓ-diamond-block)**.** We define a ℓ-diamond-block as a block that contains a hash whose value is less than $T \cdot 2^{-\ell}$ where $\ell \geq \beta$.

In the following section, we show that our proposed block weight assignment ensures security and liveness against a Byzantine adversary that controls up to $1/2$ of the total hashing power.

8. Analysis

In this section, we show that our approach provides security against an adversary that controls up to $1/2$ of the total hashing power which is an improvement over the Kiayias et al.'s proposal that can only guarantee security against an adversary that controls at most 1/3rd of the total hashing power.

8.1. Notation

Let $\Sigma(S)$ be the function that returns the total weight of all the blocks 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. The following lemma uses Chernoff Bounds to provide lower bounds on the number of superblocks in a set of rounds S.

Lemma 8.1. *$(1-\epsilon)\frac{Y(S)}{2^\ell} \leq Y_\ell(S)$ for any set S of consecutive rounds such that $|S| \geq \lambda$, $\epsilon \in (0,1]$, 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 of rounds S is a ℓ superblock, and is equal to 0 otherwise. We have,

$$x_j = \left\{ \begin{array}{ll} 1, & \text{with probability } 1/2^\ell \\ 0, & \text{otherwise with probability } 1-1/2^\ell. \end{array} \right\}$$

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

$$\mu = \mathbb{E}[Y_\ell(S)] = \frac{Y(S)}{2^\ell}.$$

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 proves that $(1-\epsilon)\frac{Y(S)}{2^\ell} \leq Y_\ell(S)$ with overwhelming probability. □

Lemma 8.2. *$Z_\ell(S) \leq (1+\epsilon)\frac{Z(S)}{2^\ell}$ for any set S of consecutive rounds such that $|S| \geq \lambda$, $\epsilon \in (0,1]$ with overwhelming probability.*

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

$$x_j = \left\{ \begin{array}{ll} 1, & \text{with probability } 1/2^\ell \\ 0, & \text{otherwise with probability } 1-1/2^\ell. \end{array} \right\}$$

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 proves that $Z_\ell(S) \leq (1+\epsilon)\frac{Z(S)}{2^\ell}$ with overwhelming probability. □

The following lemma from [1] 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 8.4 further proves that the advantage of honest parties also applies to weights.

Lemma 8.3 ([1] 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 8.4. *$\Sigma_Y(S) > \Sigma_Z(S)$ for any set S of consecutive rounds such that $|S| \geq \lambda$.*

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

$$\begin{aligned}\Sigma_Y(S) &= \sum_{i=0}^{\infty} \Sigma_Y^i(S) \\ &= \sum_{i=0}^{\infty} W_\beta(i) \cdot Y_i(S) \\ &\geq \sum_{i=0}^{\infty} W_\beta(i) \cdot Z_i(S) \qquad \text{(from Lemma 8.3)} \\ &\geq \Sigma_Z(S),\end{aligned}$$

□

A block mined by honest parties and belonging to the chain currently adopted by honest parties will be forked by the adversary iff the adversary is capable of providing a chain whose total weight is larger than the one adopted by the honest parties. Lemma 8.5 captures this condition.

Lemma 8.5. *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 is analogous to Lemma 7.3 and provides an upper-bound on the weight of blocks mined by the adversary.

Lemma 8.6. $\Sigma_Z^{\ell^-}(S) < (1+\epsilon)\Big(\sum_{i=0}^{l-1} \frac{W_\beta(i)}{2^i}\Big)\big(1-\frac{\delta}{3}\big) f\lambda$ *for any set of rounds S such that $|S| \leq \lambda$.*

Proof. Let x_j be the random variable that denotes whether the j^{th} block is a ℓ superblock. Then,

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

where $x_j = \left\{ \begin{array}{ll} 1, & \text{with probability } 1/2^\ell \\ 0, & \text{otherwise with probability } 1-1/2^\ell. \end{array} \right\}$

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

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

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

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

which completes the proof. □

Theorem 8.7 states that the proposed block weight assignment from Section 7.3 prevents the adversary from suppressing diamond blocks. The proof of the theorem is analogous to the proof of Theorem 7.6.

Theorem 8.7 (ℓ-diamond blocks unsuppressibility)**.** *If ℓ-diamond blocksare 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 8.5, 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 8.6, 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 can exist.
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 7.1, 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

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

Thus, we cannot find a set of rounds S such that $\Sigma_Y(S) \leq \Sigma_Z^{\ell^-}(S)$. Our assumption that the adversary can replace the chain C with a chain C' that does not contain a block b_ℓ is wrong. □

8.2. Safety Property

We now show that our ℓ-diamond -based approach guarantees safety against an adversary that controls up to $1/2$ of the total hashing power.

8.2.1. The Common Prefix 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. This property ensures safety since once a transaction is in a block that becomes a part of the common prefix, it is guaranteed to remain in the blockchain of all honest parties.

Lemma 8.8 (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 a 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 λf blocks at a level $\ell \geq \beta$, then $\mathcal{C}$ has more ℓ-diamond blocks than $\mathcal{C}'$.*

Proof. 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 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 cannot be possible in a typical execution as shown in Lemma 8.3. Therefore, our assumption was incorrect and C will have more ℓ-diamond blocks than C' has. □

Theorem 8.9. *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 1 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 2. Then $|\Pi^* \{(\Pi^* \cap \mathcal{C})[-1]:\}| \geq |\mathcal{C}\{(\Pi^* \cap \mathcal{C})[-1]:\}|$ with overwhelming probability.*

Proof. Let $\mathcal{C}' = \Pi'$. We need to show that, either Π will be the proof accepted by the verifier, or Π^* is a proof extending the honest chain that is longer at level 0, as mandated by the theorem statement.

Let us consider first the case where the lowest level μ containing a common ℓ-diamond block at the same position of Algorithm 2 exists. When $\mu = 0$, Algorithm 2 determines the chain with the greater weight and correctly accepts the corresponding proof. That is, Algorithm 2 will either choose $\Pi^* = \Pi$ or $\Pi^* = \Pi'$. If $\Pi^* = \Pi'$, the algorithm will choose the adversarial proof Π' which contains an unstable portion χ' that extends the honest chain's unstable portion χ at level 0 (up to k blocks long) with a longer alternative. This is the only case in which Π' can win. For the other cases, we will now argue that the adversary cannot win.

Let us now focus on the case $0 < \mu \leq \ell$. Note that, since $\mathcal{D}[\mu-1] \cap \mathcal{D}'[\mu-1] = \emptyset$ (by the minimality of μ), both superchains must have at least m blocks after their common block b. The diamond-block common-prefix Lemma (Lemma 8.8) implies that Π is accepted.

Next, consider the case where no such μ exists. Clearly, $\ell \neq \ell'$ (otherwise $\mathcal{D}[\ell] \cap \mathcal{D}'[\ell']$ would contain the genesis block) and we need to argue that $\ell > \ell'$. Assume by contradiction that $\ell < \ell'$ and consider the statement of the diamond-block common-prefix Lemma (Lemma 8.8). Together with $\ell < \ell'$, it implies that C' has fewer than m ℓ-diamond blocks after the common block with C (since C has fewer ℓ-diamond blocks than C' in total, it must also have fewer on its fork; and they must necessary share a common block, since both must begin with the genesis block). But then, both C and C' have fewer than m ℓ-diamond blocks after their common block. Since $\mathcal{D}[\ell] \cap \mathcal{D}'[\ell] = \emptyset$ by assumption, this cannot be the case, which completes the proof of the lemma. □

Based on the safety theorem, we devise a transaction acceptance rule that can provide assurance of safety to any client that wishes to accept a transaction.

Transaction Acceptance Rule In Gems, one can safely accept a transaction once it has been validated by $k \geq \lambda f$ blocks at any level $\ell \geq \beta$ based on Lemma 8.8. However, this discrete rule can be extended to have a conditional acceptance rule based on the probability of double spending.

8.3. Liveness

We now state the liveness property of our scheme. The following theorem proves that our scheme achieves the same liveness guarantee as that of the original Bitcoin protocol as shown by Garay et al. [15].

Theorem 8.10. *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 $1 - e^{-\Omega(\beta u)}$*

The theorem can be easily proved using the Chain Growth Lemma from [15] which states that the longest chain will have atleast $(1-\epsilon)f|S|$ blocks in $|S|$ rounds and using Chernoff bounds to bound the probability of obtaining ℓ-diamond blocks.

9. Limits of Blockchain Compression

Kiayias et al. present a novel scheme for compressing a PoW-based blockchain using NIPoPoWs and manage to compress the blockchain in $O(\text{poly}\log(n))$ storage and communication complexity. Their proposal leaves an open question on whether Kiayias et al.'s technique is optimal or there may exist even better techniques to compress blockchains in lesser storage space or communication requirements. 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.

9.1. Model

In order to derive a result for any general PoW-based blockchain protocol, we need to work with a model that can accomodate blockchain protocols that have very different design schemes than the Bitcoin protocol. A generalised version of a blockchain is known as a *block-DAG* in which blocks are arranged in the form of a *Directed Acyclic Graph (DAG)*. Let us assume we have a blockchain $\mathcal{B}$ in which the blocks are organised as a *Directed Acyclic Graph*. $\mathcal{B}$ is said to be valid if $\forall b \in \mathcal{B}$, predecessors of b also lie in $\mathcal{B}$ 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* $\mathsf{S}(\mathcal{B})$. The topological sort functionality takes in a valid block-DAG $\mathcal{B}$ and returns a list L of blocks in $\mathcal{B}$ such that $\forall i \in [n]$, $\mathcal{B} \setminus L[1:i]$ is also a valid block-DAG, where n is the number of blocks in $\mathcal{B}$.
- *PoPoW* $\mathsf{P}(\mathcal{B})$. The *Proof-of-Proof-of-Work* functionality takes in a block-DAG $\mathcal{B}$ and returns a PoPoW $\mathsf{P}(\mathcal{B})$ such that $\mathsf{P}(\mathcal{B}) \neq \mathsf{P}(\mathcal{B}') \; \forall \; \mathcal{B}' \subsetneq \mathcal{B}$. We assume the functionality is guaranteed to provide a PoPoW for any valid blockchain.
- *Compress* $\mathsf{C}(\mathcal{B})$. The Compress functionality takes in a block-DAG $\mathcal{B}$ and returns an output of at most ℓ bits.
- *Equivalent PoPoW* $\mathsf{U}(\mathsf{C}(\mathcal{B}))$. This functionality produces a PoPoW from a compressed block-DAG such that
$$\mathsf{U}(\mathsf{C}(\mathcal{B})) = \mathsf{P}(\mathcal{B}).$$

9.2. 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 the subsets of the block-DAG by removing the blocks from the end can be nested like a russian doll in the following lemma.

Lemma 9.1. $\mathcal{B} \setminus L[1:i] \subsetneq \mathcal{B} \setminus L[1:j] \; \forall j < i$

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

Now we come to the key theorem proving the lower limit of the communication complexity.

Theorem 9.2. *The PoPoW* $\mathsf{P}(\mathcal{B})$ *must contain at least* $\lfloor \log_2(n) \rfloor$ *bits where* n *is the number of blocks in* $\mathcal{B}$.

Proof. Let us assume 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 $B \setminus L[1:i] \; \forall i \in [n]$.

From Lemma 9.1,
$$(\mathcal{B} \setminus L[1:n]) \subsetneq (\mathcal{B} \setminus L[1:n-1]) \subsetneq \ldots \subsetneq (\mathcal{B} \setminus L[1:1])$$
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.

Since, we reach a contradiction, 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. □

9.3. Storage Complexity

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

Lemma 9.3. *For two blockchains* $\mathcal{B}_i$ *and* $\mathcal{B}_j$, $\mathcal{B}_i \subsetneq \mathcal{B}_j \implies \mathsf{C}(\mathcal{B}_i) \neq \mathsf{C}(\mathcal{B}_j)$.

Proof. Let us assume that there are two parties that have $\mathcal{B}_i$ and $\mathcal{B}_j$ such that $B_i \subsetneq B_j$ and $\mathsf{C}(\mathcal{B}_i) = \mathsf{C}(\mathcal{B}_j) = s$. By definition of PoPoW,
$$\mathsf{P}(\mathcal{B}_i) = \mathsf{U}(s) = \mathsf{U}(s) = \mathsf{P}(\mathcal{B}_j).$$
Hence, we reach our contradiction. □

Theorem 9.4. *A party that can transmit a valid PoPoW for a blockchain* $\mathcal{B}$ *must store at least* $\lfloor \log_2(n) \rfloor$ *bits where the* n *is the number of blocks in* $\mathcal{B}$.

Proof. Let us assume that the Compress functionality allow a party that can store the blockchain $\mathcal{B}$ in l bits, where l is less than $\lfloor \log_2(n) \rfloor$. Consider the list L produced via topological sort,

From Lemma 9.1,
$$(\mathcal{B} \setminus L[1:n]) \subsetneq (\mathcal{B} \setminus L[1:n-1]) \subsetneq \ldots \subsetneq (\mathcal{B} \setminus L[1:1])$$

From Lemma 9.3, we know that $\forall \mathcal{B}' \subsetneq \mathcal{B}$, $\mathsf{C}(\mathcal{B}') \neq \mathsf{C}(\mathcal{B})$.

There are at least n subsets of $\mathcal{B}$ that must be mapped to a unique sequence of l-bits. However, there are only $2^l < n$ sequences in the codomain of C. Hence, by the Pigeonhole Principle, at least two subsets must be mapped to the same output.

Hence, we reach a contradiction. □

10. Discussion and Future Work

In this paper, we have presented an improved scheme to compress PoW-based blockchains based on Kiayias et al.'s proposal that achieves optimal security in a Byzantine adversary, i.e., an adversary that controls strictly less than 1/2 of the total computational power. We have proved the security of our scheme using the same mathematical framework introduced by Kiayias et al. [1]. We also have established a lower bound on the storage and communication complexity for the blockchain protocol.

Future Work

Bünz et al. [2] 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. Secondly, superblock-based NIPoPoWs have only been shown to work in a setting with constant difficulty, and therefore, we leave it to future work to determine a scheme that works in a setting with variable difficulty. To the best of our knowledge, the lower bound presented in this paper is lower than what is achieved by our solution and any other solution to compress blockchains. This indicates that no optimal blockchain compression scheme has been discovered yet. We leave it for future work to either find an optimal blockchain compression scheme or improve the lower bound to $O(\mathrm{poly}\log(n))$.

References

[1] 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)*. New York, NY, USA: Association for Computing Machinery, 2021, p. 3487–3501. [Online]. Available: https://doi.org/10.1145/3460120.3484784

[2] B. Bünz, L. Kiffer, L. Luu, and M. Zamani, "Flyclient: Super-light clients for cryptocurrencies," in *2020 IEEE Symposium on Security and Privacy (SP)*, 2020, pp. 928–946.

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

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

[5] L. Luu, V. Narayanan, C. Zheng, K. Baweja, S. Gilbert, and P. Saxena, "A secure sharding protocol for open blockchains," in *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2016.

[6] E. Kokoris-Kogias, P. Jovanovic, L. Gasser, N. Gailly, E. Syta, and B. Ford, "Omniledger: A secure, scale-out, decentralized ledger via sharding," in *Proceedings of the IEEE Symposium on Security and Privacy (S&P)*, 2018.

[7] M. Zamani, M. Movahedi, and M. Raykova, "Rapidchain: Scaling blockchain via full sharding," in *Proceedings of the ACM SIGSAC Conference on Computer and Communications Security (CCS)*, 2018.

[8] A. Durand, E. Anceaume, and R. Ludinard, "Stakecube: Combining sharding and proof-of-stake to build fork-free secure permissionless distributed ledgers," in *Proceedings of the International conference on networked systems (NETYS)*, 2019.

[9] N. Durov, "Telegram Open Network," Tech. Rep., 2019. [Online]. Available: https://ton.org/

[10] T. E. Team, "Elrond - A Highly Scalable Public Blockchain via Adaptive State Sharding and Secure Proof of Stake," Tech. Rep., 2019. [Online]. Available: https://https://elrond.com/assets/files/elrond-whitepaper.pdf

[11] J. W. and H. W., "Monoxide: Scale out blockchains with asynchronous consensus zones," in *Proceedings of the USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, 2019.

[12] H. Huawei, P. Xiaowen, Z. Jianzhou, Z. Shenyang, L. Yue, Z. Zibin, and G. Song, "Brokerchain: A cross-shard blockchain protocol for account/balance-based state sharding," in *Proceedings of the IEEE Conference on Computer Communications (Infocom)*, 2022.

[13] I. Abraham and D. Malkhi, "The blockchain consensus layer and bft," *Bulletin of the European Association for Theoretical Computer Science*, no. 123, 2017.

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

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

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

[17] A. Kiayias, A. Miller, and D. Zindros, "Non-interactive proofs of proof-of-work," in *Financial Cryptography and Data Security*, J. Bonneau and N. Heninger, Eds. Cham: Springer International Publishing, 2020, pp. 505–522.