A New Blockchain Proposal Supporting Multi-Stage Proof-of-Work

Palash Sarkar
Applied Statistics Unit
Indian Statistical Institute
203, B.T. Road, Kolkata
India 700108.
email: palash@isical.ac.in
Phone: 91 33 2575 2830

July 13, 2021

Abstract

We introduce a new variant of decentralised, trustless, permissionless blockchain. The main novelty is that the proof-of-work for mining a block is divided into multiple stages. An appropriate linkage structure is defined so that it becomes possible to simultaneously work on various stages of different blocks. The overall effect is an improvement in the transaction processing rate and the time for confirming a transaction. These are achieved without compromising on security. The division of the proof-of-work into several stages also divides the block reward into an equal number of stage rewards. Once a block gets onto the blockchain, the miner which successfully completed a particular stage can claim the reward for that stage. This ensures a more equitable distribution of the reward for successfully mining a block.

Keywords: blockchain, proof-of-work, pipelining, mining, Bitcoin.

1 Introduction

Bitcoin was launched by Satoshi Nakamoto in 2009 [11]. It is a form of currency which operates in a decentralised, trustless and permissionless environment. It is decentralised in the sense that there is no central authority which issues or manages the currency; it is trustless in the sense that users do not require to trust any entity to use the currency; and it is permissionless in the sense that anybody can join the community of users of the currency without requiring any kind of permission. The creation of such a currency is a fascinating technological feat. Starting from its obscure origin a decade ago, Bitcoin is presently an internationally well known invention with a market capitalisation of more than 170 billion US Dollars[1].

Bitcoin is based on the blockchain technology which combines ideas from cryptography and distributed computing. Since cryptography plays an essential role, Bitcoin is called a cryptocurrency. We refer to [8] for an exposition of Bitcoin and the problem it solves.

A blockchain is a method to implement what is called a distributed ledger, i.e., a ledger whose entries are immutable and which is stored in a distributed fashion. The creation of Bitcoin kick-started an immense amount of activity in cryptocurrency in particular and more generally on the blockchain technology and distributed ledger. Over the course of time, several problems have been identified with the performance of Bitcoin. We discuss some of these below.

[1] `https://coinmarketcap.com/`, accessed on 21 February, 2020.

A major issue is that of scalability. The throughput of Bitcoin is less than 10 transactions per second [3]. In Bitcoin, blocks are created at the rate of one block in 10 minutes. Bitcoin suggests that for a transaction to be considered confirmed, it is desirable to wait until five blocks have been added to the blockchain after the block containing the transaction has been mined. Consequently, the time for a transaction to be confirmed is about an hour from the time that the transaction is considered for inclusion in the blockchain. The slow rate of transaction processing and the long time for confirmation of transactions make Bitcoin difficult to use for payment systems which produce transactions at a much faster rate and also require a shorter time for confirmation of an individual transaction.

A second major issue is the distribution of block rewards. The Bitcoin protocol follows a 'winner takes all' approach to allocating block rewards, i.e., the entity which is the first to successfully complete the proof-of-work for a block gets the entire reward for creating the block. Other entities which put in a significant amount of computation towards the proof-of-work, but, were beaten by the winner, get nothing. This approach leads to an intense competition among miners for block mining. A result of such competition has been the formation of mining pools. Miners with moderate computational resources pool their resources together for block mining. Mining pools have been viewed by several authors as a move towards centralisation which cuts at one of the basic premises of Bitcoin.

Our Contributions

This work describes a modification of the proof-of-work blockchain used by Bitcoin. Broadly speaking the idea is the following. The proof-of-work to be done for mining a block is divided into stages. Suppose there are k stages. Consider a block B_i for $i \geq k$. Stage number 0 of the proof-of-work for mining B_i depends upon block B_{i-k}; stage number 1 of the proof-of-work for mining B_i depends upon block B_{i-k+1} and the completion of stage number 0; and so on until stage number $k-1$ of the proof-of-work is taken up after mining of block B_{i-1} and the completion of stage number $k-2$. An effect of the multi-stage structure of the proof-of-work is that the following tasks are independent and can be performed simultaneously: proof-of-work of stage number 0 of block B_{i+k}, proof-of-work of stage number 1 of block B_{i+k-1}, ..., proof-of-work of stage number $(k-1)$ of block B_{i+1}. This leads to a situation where separate groups of miners can simultaneously proceed on the proof-of-work of various stages of different blocks. The block reward is divided into stages. Upon successful completion of the proof-of-work of a particular stage, a miner can release the incremental information. Once the entire proof-of-work is completed, the block is added to the blockchain. At this point, the miners who had completed proof-of-work for the various stages can claim the rewards for the stages they had completed.

We introduce a new notion of mining permit to boost cooperative behaviour among the miners. The top-level idea is the following. Before attempting a proof-of-work for a particular stage, miners have to generate a random number. It is viable for a miner to embark on mining only if the generated random number satisfies a pre-defined condition. Miners who are successful in generating such random numbers start on the proof-of-work of the next stage, while miners who are unsuccessful do not do so. We refer to the pre-defined condition as a mining permit. Once a miner completes the proof-of-work of a particular stage, it is incentivised to release the incremental information, since this miner is not sure that it will be able to obtain a mining permit for the next stage. Such release of incremental information promotes cooperative behaviour among the miners.

The new proposal offers two advantages over the Bitcoin blockchain.

Improved transaction processing: Using a k-stage proof-of-work blockchain and without changing the total time for mining a block, the rate at which transactions are processed increases by a factor of k compared to a usual blockchain. The time required for a transaction to be confirmed also goes down substantially. The improved rate of transaction processing and the lower time for confirmation mitigates the scalability problem to some extent.

More equitable distribution of block rewards: Successful completion of the proof-of-work of a block provides rewards to all the miners who have contributed by successfully completing the proof-of-work of the various stages. So, unlike a 'winner takes all' policy, more entities benefit from the successful mining of a block. Since the blockchain itself provides opportunities for cooperative behaviour, the formation of mining pools becomes redundant. Also, the greater opportunity for receiving rewards will incentivise entities with lesser computational power to participate in block mining thus encouraging a move towards decentralisation.

Previous and Related Works

There have been a number of works which have tried to alleviate the problems of Bitcoin blockchain. We discuss some of these along with the relevance to multi-stage proof-of-work blockchain.

A line of works [13, 9, 14] have proposed the replacement of a linear blockchain by a directed acyclic graph (DAG). This leads to faster processing times for transactions. A basic problem in DAG based schemes is to obtain an ordering of the blocks. This requires a careful analysis. DAG based schemes are quite far from the notion of multi-stage blockchain that we introduce.

A concept called Fruitchain [12] has been proposed to address some of the problems with the Bitcoin blockchain. In a Fruitchain, the blocks contain fruits, where each fruit contains a list of transactions. Fruits refer to a recently mined block. Both fruits and blocks are to be mined. The concept of Fruitchain is complementary to that of a multi-stage proof-of-work blockchain. Later we indicate how the two concepts may be combined.

In a usual blockchain, all the miners simultaneously compete with each other in the mining of the next block. There have been proposals which partition the set of entities such that each group of entities work in parallel on different sets of transactions. Such a strategy has been called sharding and leads to improvement of transaction processing [10, 1, 15]. Later we indicate the possibility of combining sharding strategies with multi-stage proof-of-work blockchain.

Computing proof-of-work can be an energy intensive procedure. There have been several suggestions to replace proof-of-work by proof-of-stake [7, 6, 2]. The idea is that holder of a certain amount of currency has a stake in the currency. Such a holder provides a proof-of-stake and obtains a chance to create a new block. There is no competition for block creation. Some kind of distributed consensus mechanism is followed to determine the entity which will create the new block. A proof-of-stake based currency provides fast transaction processing and avoids wastage of energy required for block mining. On the downside, proof-of-stake schemes tend to favour the richer entities which again moves away from the goal of decentralisation. So, while proof-of-stake is an important concept, we expect proof-of-work based schemes to co-exist in the cryptocurrency space in the forseeable future.

2 Background on Blockchain

In this section, we provide a brief overview of blockchain. Our description is based on the Bitcoin blockchain.

Two basic cryptographic primitives are required to implement a blockchain.

Hash function: A hash function is a map $H : \mathcal{D} \rightarrow \mathcal{R}$ where $\mathcal{D}$ and $\mathcal{R}$ are finite non-empty sets with $\#\mathcal{D} > \#\mathcal{R}$. Typically, $\mathcal{D}$ is the set of all binary strings having some maximum possible length and $\mathcal{R}$

is the set of all binary strings of some fixed length.

- H is said to be collision resistant, if it is computationally difficult to find distinct $x, x' \in \mathcal{D}$ such that $H(x) = H(x')$.
- H is said to be pre-image resistant (or, one-way) if given $y \in \mathcal{R}$, it is computationally difficult to find $x \in \mathcal{D}$ such that $H(x) = y$.
- H is said to be second pre-image resistant if given $x \in \mathcal{D}$, it is computationally difficult to find $x' \in \mathcal{D}$ such that $x \neq x'$ and $H(x) = H(x')$.

Digital signature scheme: A digital signature scheme is a triplet of algorithms: $(\mathsf{setup}, \mathsf{sign}, \mathsf{verify})$.

- An entity executes the algorithm setup to obtain a signing-verification key pair $(\mathsf{sk}, \mathsf{pk})$.
- Algorithm sign is applied to a message M and the signing key sk to produce a signature σ.
- The verification algorithm runs on a message-signature pair (M, σ) and the verification key pk and returns true (indicating that the pair is valid) or false (indicating that the pair is invalid).

For setting up the blockchain, two hash functions H_0 and H are chosen. By n_1 we will denote the length of the outputs of H_0 and by n_2 we will denote the length of the outputs of H. Bitcoin uses SHA-256 (with $n_1 = 256$) to instantiate H_0 and a combination of SHA-256 and RIPEMD-160 (with $n_2 = 160$) to instantiate H. Additionally, an appropriate digital signature scheme $(\mathsf{setup}, \mathsf{sign}, \mathsf{verify})$ is fixed. Bitcoin uses the elliptic curve digital signature algorithm (ECDSA).

Address: Addresses are computed by applying the hash function H to a public key pk, i.e., $\mathfrak{a}$ is an address if $\mathfrak{a} = H(\mathsf{pk})$. Amounts of the cryptocurrency are assigned to addresses. An entity which possesses the signing key sk corresponding to the verification key pk can spend the amount of cryptocurrency associated to the address $\mathfrak{a} = H(\mathsf{pk})$.

Transactions: A record of an interaction between two or more parties is called a transaction. A transaction will be considered valid only if it is signed by the relevant parties. More formally, a transaction T is a tuple $(\mathsf{IL}, \mathsf{OL}, \sigma)$ where

- $\mathsf{IL} = ((\mathsf{pk}_1, \mathfrak{c}_1), \ldots (\mathsf{pk}_s, \mathfrak{c}_s))$, $s \geq 1$. Here $\mathsf{pk}_1, \ldots, \mathsf{pk}_s$ are public keys and for $i = 1, \ldots, s$, $\mathfrak{c}_i$ is the amount of currency associated with $H(\mathsf{pk}_i)$.
- $\mathsf{OL} = ((\mathfrak{a}_1, \mathfrak{d}_1), \ldots (\mathfrak{a}_t, \mathfrak{d}_t))$, $t \geq 1$. Here $\mathfrak{a}_1, \ldots, \mathfrak{a}_t$ are addresses and for $j = 1, \ldots, t$, $\mathfrak{d}_j$ is the amount of currency to be associated to the address $\mathfrak{a}_j$.
- $\sum_{i=1}^{s} \mathfrak{c}_i \geq \sum_{j=1}^{t} \mathfrak{d}_j$. The difference $(\sum_{i=1}^{s} \mathfrak{c}_i) - \left(\sum_{j=1}^{t} \mathfrak{d}_j\right)$ is the transaction fee.
- σ consists of the signature(s) on $(\mathsf{IL}, \mathsf{OL})$ constructed using the signing key(s) corresponding to the public key(s) $\mathsf{pk}_1, \ldots, \mathsf{pk}_s$.

A special kind of transaction, called a coinbase transaction, does not have input public keys, i.e., IL is empty and so, there is no signature component σ in such a transaction. It simply consists of a pair $(\mathfrak{a}, \mathfrak{d})$ indicating that an amount $\mathfrak{d}$ is assigned to the address $\mathfrak{a}$. Successful creation of a block generates new amounts of the cryptocurrency. Such newly created cryptocurrency along with the sum of all the transaction fees in the block is the block reward and is assigned to a particular address using a coinbase transaction.

Root hash tree of a list of transactions: Let $\mathcal{L}$ be a list of transactions. These are hashed together using a tree structure. For example, Bitcoin uses the Merkle hash tree to stitch together the hashes of the transactions in the list. The final output of the hash tree will be called the *root hash* and denoted as $\mathsf{RH}(\mathcal{L})$.

Blockchain: A blockchain is a sequence of blocks $B_0, B_1, \ldots, B_{r-1}$. The first block B_0 is special since it has no previous block. It is sometimes called the genesis block. The block B_{r-1} is called the header of the blockchain.

A block contains a list of transactions, chaining information and metadata. In the following description, we omit the metadata component of a block. The i-th block B_i of the blockchain can be formally considered to be a tuple $B_i = (i, \mathsf{bdigest}_i, \mathcal{L}_i, t_i, \eta_i)$, where

- i is a non-negative integer denoting the block number;
- $\mathcal{L}_i$ is the list of transactions in the block where the first transaction in $\mathcal{L}_i$ is a coinbase transaction;
- t_i is a positive integer which denotes the target value for the block and is specified by the rules for the blockchain;
- η_i is a positive integer providing a proof-of-work done in creating the block;
- $\mathsf{bdigest}_i$ is computed as follows:

$$\mathsf{bdigest}_i = \begin{cases} H_0\left(i, \mathsf{RH}(\mathcal{L}_i), t_i, \eta_i\right) & \text{if } i = 0, \\ H_0\left(\mathsf{bdigest}_{i-1}, i, \mathsf{RH}(\mathcal{L}_i), t_i, \eta_i\right) & \text{if } i > 0. \end{cases}$$

The quantity $\mathsf{bdigest}_i$ is called the digest of the block B_i and has to satisfy the following condition.

$$\mathsf{bdigest}_i < t_i. \tag{1}$$

Ensuring that the condition (1) holds requires performing a computation. While computing $\mathsf{bdigest}_i$, $i > 0$, the input to H_0 is $(\mathsf{bdigest}_{i-1}, i, \mathsf{RH}(\mathcal{L}_i), t_i, \eta_i)$. Of these, the only flexibility is in choosing the value η_i, since the other components are fixed. Since H_0 is pre-image resistant, the only way to ensure that (1) holds, is to repeatedly apply H_0 with various values of η_i until a value is obtained for which (1) holds. On an average, about $2^{n_1}/t_i$ applications of the hash function H_0 will be required to ensure that (1) holds. So, the particular value of η_i provided as part of the block is a proof that a certain amount of work has been done.

Pictorially, a blockchain may be viewed in the following manner.

$$B_0 \leftarrow B_1 \leftarrow B_2 \leftarrow \cdots \leftarrow B_{r-1}.$$

The arrows in the blockchain point in the backward direction indicating that for $i > 0$, B_i is computed from B_{i-1}, i.e., $\mathsf{bdigest}_{i-1}$ is part of the input to H_0 while computing $\mathsf{bdigest}_i$.

The blockchain grows in the forward direction. Suppose the present blockchain is $B_0, \ldots, B_{r-1}$. A list of transactions $\mathcal{L}_r$ is chosen and a suitable value η_r is obtained such that $\mathsf{bdigest}_r$ is equal to $H_0(r, \mathsf{bdigest}_{r-1}, \mathsf{RH}(\mathcal{L}_r), t_r, \eta_r)$ and $\mathsf{bdigest}_r < t_r$. Then the block $B_r = (r, \mathsf{bdigest}_r, \mathcal{L}_r, t_r, \eta_r)$ is appended to the blockchain resulting in the new blockchain $B_0, \ldots, B_{r-1}, B_r$. Creation of the new block B_r by finding the proof-of-work value η_r is called the mining of the block B_r. Entities whose goal is to create new blocks are called miners. A miner which successfully creates a block obtains the block reward for creating the block.

Difficulty: The difficulty parameter of a blockchain determines how difficult it is to mine blocks. The blockchain design specifies how the difficulty parameter is to be determined for a particular block. The difficulty parameter in turn defines the target value t_i of the block B_i. The difficulty of a blockchain is the sum of difficulties of all the blocks in the blockchain. It is a measure of the amount of work that has been put in to create the blockchain.

Block Completion Time: As explained above, creating a new block requires obtaining a proof-of-work. This requires a certain amount of time. An overall goal of the blockchain is to ensure that blocks are created at a constant rate. Let $\mathfrak{T}$ be the time in seconds such that blockchain ensures that on an average a new block is created every $\mathfrak{T}$ seconds. For example, in Bitcoin, blocks are created at a rate of one block in about 10 minutes, i.e., $\mathfrak{T} = 600$ seconds.

The proof-of-work essentially consists of repeatedly applying the hash function H_0. The target value determines the number of times H_0 needs to be applied to obtain a suitable proof-of-work. In a competitive environment, many miners are simultaneously attempting to create the next block. The hash rate of the whole network is the number of times H_0 is applied each second by all the miners. As more miners join and/or employ special purpose hardware for computing H_0, the hash rate increases. As a result, new blocks would be mined faster. Creation of new blocks results in creation of new amounts of cryptocurrency. So, speeding up the block mining time can lead to an inflationary situation. To prevent this, the blockchain specifies rules for adjusting the difficulty and hence the target value, of mining a block. If the block mining time tends to decrease below $\mathfrak{T}$, then the difficulty for the next blocks is increased. Conversely, if for some reason, the block mining time tends to increase above $\mathfrak{T}$, then the difficulty for the next blocks is decreased. The difficulty adjustment procedure ensures that on an average the block mining time equals $\mathfrak{T}$.

Peer-to-peer network: The blockchain considered in this work operates in a trustless, permissionless and decentralised environment. Entities are nodes of a peer-to-peer network. Transactions are generated by the interaction of these entities among themselves. The generated transactions are broadcast over the network using a gossip protocol. The goal is to include such transactions in a block and append to the blockchain. The work of creating new blocks is done by the miners. Each miner maintains a list of transactions which have not yet been included in the blockchain. For creating a new block, a miner chooses a list of transactions from amongst those that are not part of the blockchain and attempts to mine a new block. If it is successful, it propagates the newly created block to the network using the gossip protocol.

Before relaying any information, a node in the network will perform validation checks on the information it relays. This will include verification of formatting, signature, proof-of-work and other consensus rules of the network. Such rules include verification of timestamps and the rules for updating the targets.

Each entity participating in the network maintains a copy of the blockchain. Whenever, an entity creates a new block or receives one from one of its peers, it appends the block to the copy of the blockchain that it maintains. Assuming that all entites have the same copy of the blockchain and all of them receive the same new block, the new copy of the blockchain maintained by all the entities will also be the same. It is, however, possible that two (or, more) miners almost simultaneously create a new block and propagate to the network. This may result in a particular entity receiving two blocks both of which are compatible with the copy of the blockchain that it maintains. More generally, the following situation may arise. Suppose that the blockchain is $B_0, \ldots, B_{i-1}, B_i, \ldots, B_{r-1}$. An entity receives blocks $B'_i, \ldots, B'_{r'}$ such that $B_0, \ldots, B_{i-1}, B'_i, \ldots, B'_{r'}$ is a valid blockchain. The blockchain rules specify a conflict resolution mechanism to determine which blockchain to retain. Basically, the rule is to retain the blockchain having higher difficulty among the two possible chains. Following this rule may lead the entity to discarding the blocks $B_i, \ldots, B_r$ and keeping $B_0, \ldots, B_{i-1}, B'_i, \ldots, B'_{r'}$ as its copy of the blockchain. Such discarding of blocks is a transient effect and eventually all entities in the network have the same copy of the blockchain.

Confirmation of a transaction: Once a transaction is part of a block which is included in the blockchain, it may be considered to be partially confirmed. For full confirmation, due to the transitory effect discussed above, it may be a good idea to wait until a few more blocks have been appended to the blockchain. Once the block containing the transaction gets buried sufficiently deep in the blockchain, it is extremely unlikely that this block will be discarded later. Suppose that ν further blocks are required to achieve full confirmation. Then the average time for confirmation is about $\mathfrak{T}(\nu + 1)$ seconds.

Transaction Processing Rate: The maximum size of a block is fixed. Depending on their complexities, transactions on the other hand, can have variable sizes. Suppose that on an average, the number of transactions that can be accommodated in a block is $\mathfrak{m}$. The block completion time is $\mathfrak{T}$ seconds. So, on an average, about $\mathfrak{m}/\mathfrak{T}$ transactions are added to the blockchain in one second. The quantity $\mathfrak{m}/\mathfrak{T}$ is the transaction processing rate of the blockchain.

51% Attack: Suppose an adversary has a hash rate of ρ_a, i.e., it can perform ρ_a applications of H_0 per second. Further, let ρ be the hash rate of all the other honest miners in the network. Assume that the target for a block is t so that about $2^{n_1}/t$ applications of H_0 are required to mine the block. So, the expected time for the adversary to mine the block is $2^{n_1}/(t\rho_a)$ and the expected time for all the honest miners together to mine the block is $2^{n_1}/(t\rho)$. The expected mining time for the adversary is less than the expected mining time of the honest miners if $2^{n_1}/(t\rho_a) < 2^{n_1}/(t\rho)$ which holds if and only if $\rho < \rho_a$, i.e., the hash rate of the adversary is higher than the hash rate of the honest miners. Let $\mathfrak{f}_a = \rho_a/(\rho + \rho_a)$ denote the fraction of the total hash rate of the network which is controlled by the adversary. The condition $\rho < \rho_a$ is equivalent to $\mathfrak{f}_a > 1/2$. So, if the adversary controls more than half the hash rate of the entire network, then with high probability it can mine blocks faster than the all the honest miners in the network. This provides the adversary with a great degree of control over the blockchain with a good chance of erasing a previously added block from the blockchain. An adversary's ability to control more than half the hash rate is called the 51% attack.

Selfish mining: Suppose an adversary is able to mine a block ahead of the public blockchain. It can then withold the block and keep on privately mining on top of it while it is still ahead of the public blockchain. At a later stage, it releases the blocks it has mined. At this point, the honest miners must discard the blocks they have mined and replace them with the blocks mined by the adversary. This strategy has been called selfish mining and analysed in details [5].

3 A New Blockchain Proposal

The envisaged blockchain will operate in a decentralised, trustless, permissionless environment which is same as the one described in Section 2. Users of the blockchain will be nodes in a peer-to-peer network and will communicate using the gossip protocol of the network. Each node will maintain its own local copy of the blockchain. Interactions between the users will create transactions. These transactions will be relayed to the whole network. Miners will perform the task of creating new blocks. The new blocks will also be relayed to the whole network. Nodes will append new blocks to the local copies of the blockchain that they maintain. This may lead to conflicts as explained in Section 2. The conflict resolution mechanism will be similar to that mentioned in Section 2, namely to retain the blocks which required a greater amount of work to be created and in case of ties to retain the blocks which were received earlier.

3.1 Structure of the Proposed Blockchain

For setting up a multi-stage blockchain, hash functions $H_0, \ldots, H_{k-1}$ and H are chosen. The parameter k denotes the number of stages in the proof-of-work. Also, a digital signature scheme $(\mathsf{setup}, \mathsf{sign}, \mathsf{verify})$ is selected. The digital signature scheme will be used for signing transactions in the manner described in Section 2. The hash functions $H_0, \ldots, H_{k-1}$ will be used for proof-of-work computations while the hash function H will be used for address computation by applying H to public keys as described in Section 2. The structure of transactions are the same as in Section 2 and the definition of the root hash of a list of transactions also remains unchanged. In a multi-stage blockchain, there will be no coinbase transactions.

The blockchain will chain together blocks. A general block of the blockchain contains the following information.

$$\boxed{\begin{array}{l} \mathsf{bn}, \\ \mathsf{bdigest}, \\ \mathcal{L}, \\ t_0,\ \eta_0,\ \tau_0,\ \mathfrak{a}_0,\ \mathfrak{c}_0 \\ t_1,\ \eta_1,\ \tau_1,\ \mathfrak{a}_1,\ \mathfrak{c}_1 \\ \vdots \\ t_{k-1},\ \eta_{k-1},\ \tau_{k-1},\ \mathfrak{a}_{k-1},\ \mathfrak{c}_{k-1} \end{array}}$$

Here:

- bn is the block number.
- $\mathsf{bdigest}$ is the block digest. We later explain the computation of the block digest.
- $\mathcal{L}$ is the (possibly empty) list of transactions in the block.
- For $j = 0, \ldots, k-1$,
  - t_j is the target for stage j;
  - η_j is the nonce corresponding to the proof-of-work for stage j;
  - τ_j is the timestamp for the completion of stage j;
  - $\mathfrak{a}_j$ is the address to which the reward for completing stage j is to be assigned;
  - $\mathfrak{c}_j$ is the reward for completing stage j; the block assigns $\mathfrak{c}_j$ coins to the address $\mathfrak{a}_j$.

The block reward consists of the new coins that is created on successful mining of the block and the sum of all the transactions fees in the block. We denote the block reward of a block B by $\mathsf{BR}(B)$. We do not specify any particular monetory policy for creating new coins. All that we require is that given a block number, it should be possible to determine the number of coins that is created on the successful mining of that block.

There has to be a policy for distributing $\mathsf{BR}(B)$ to the k addresses corresponding to the successful completion of the k stages. The simplest would be $\mathfrak{c}_0 = \cdots = \mathfrak{c}_{k-1} = \mathsf{BR}(B)/k$. We will assume this distribution.

The blockchain is a sequence of blocks:

$$B_0, B_1, \ldots, B_{k-1}, B_k, B_{k+1} \ldots$$

Blocks $B_0, B_1, \ldots, B_{k-1}$ are start-up (or genesis) blocks while later blocks, i.e., $B_k, B_{k+1}, \ldots$ are general blocks. The descriptions of general and start-up blocks are separately given below.

General blocks: For $i \geq 0$,

$$B_{i+k} \quad = \quad \boxed{\begin{array}{l} i+k, \\ \mathsf{bdigest}_{i+k}, \\ \mathcal{L}_{i+k}, \\ t_{i+k,0},\ \eta_{i+k,0},\ \tau_{i+k,0},\ \mathfrak{a}_{i+k,0},\ \mathfrak{c}_{i+k,0} \\ t_{i+k,1},\ \eta_{i+k,1},\ \tau_{i+k,1},\ \mathfrak{a}_{i+k,1},\ \mathfrak{c}_{i+k,1} \\ \vdots \\ t_{i+k,k-1},\ \eta_{i+k,k-1},\ \tau_{i+k,k-1},\ \mathfrak{a}_{i+k,k-1},\ \mathfrak{c}_{i+k,k-1} \end{array}}$$

The proof-of-work of the various stages and the final block digest $\mathsf{bdigest}_{i+k}$ of block B_{i+k} are defined as follows.

$$\left.\begin{array}{rcl} g_{i+k,0} & = & H_0\left(\mathsf{bdigest}_i, i+k, \mathsf{RH}(\mathcal{L}_{i+k}), t_{i+k,0}, \mathfrak{a}_{i+k,0}, \mathfrak{c}_{i+k,0}, \tau_{i+k,0}, \eta_{i+k,0}\right); \\ g_{i+k,1} & = & H_1\left(\mathsf{bdigest}_{i+1}, g_{i+k,0}, t_{i+k,1}, \mathfrak{a}_{i+k,1}, \mathfrak{c}_{i+k,1}, \tau_{i+k,1}, \eta_{i+k,1}\right); \\ \cdots & \cdot & \cdots \\ g_{i+k,k-1} & = & H_{k-1}\left(\mathsf{bdigest}_{i+k-1}, g_{i+k,k-2}, t_{i+k,k-1}, \mathfrak{a}_{i+k,k-1}, \mathfrak{c}_{i+k,k-1}, \tau_{i+k,k-1}, \eta_{i+k,k-1}\right). \end{array}\right\} \quad (2)$$

Finally, $\mathsf{bdigest}_{i+k}$ is set to be equal to $g_{i+k,k-1}$.

Block B_{i+k} depends upon k previous blocks, namely blocks $B_i, \ldots, B_{i+k-1}$. The proof-of-work required to create (or, mine) block B_{i+k} comes in k stages. Essentially, the values $\eta_{i+k,0}, \ldots, \eta_{i+k,k-1}$ are the proof-of-work of the individual stages. For $j = 0, \ldots, k-1$, the nonce $\eta_{i+k,j}$ is used to obtain $g_{i+k,j}$ such that $g_{i+k,j} < t_{i+k,j}$.

A schematic diagram explaining the multi-stage nature of the proof-of-work is shown in Figure 1 for $k = 3$. It shows the dependence structure of block B_{i+3} on the three previous blocks, namely, B_i, B_{i+1} and B_{i+2}. The proof-of-work has three stages shown as Stage-0, Stage-1 and Stage-2 in the figure. The arrow from Stage-0 to B_i indicates that $\mathsf{bdigest}_i$ is provided as part of the input of the hash function H_0 while computing $g_{i+3,0}$ for Stage-0. From Stage-1, there are two arrows, one to Stage-0 and the other to B_{i+1} indicating that $g_{i+3,0}$ and $\mathsf{bdigest}_{i+1}$ are provided as part of the input of the hash function H_1 while computing $g_{i+3,1}$ for Stage-1. Similarly, from Stage-2, there are two arrows, one to Stage-1 and the other to B_{i+2} indicating that $g_{i+3,1}$ and $\mathsf{bdigest}_{i+2}$ are provided as part of the input of the hash function H_2 while computing $g_{i+3,2}$ for Stage-2.

Verification conditions for the different stages of block B_{i+k} are the following.

Verification of mining permits:

$$\left.\begin{array}{rcl} H_0(\mathsf{bdigest}_i, \mathfrak{a}_{i+k,0}) \bmod 2^p & = & 0, \\ H_1(g_{i+k,0}, \mathfrak{a}_{i+k,1}) \bmod 2^p & = & 0, \\ \cdots & \cdot & \cdots \\ H_{k-1}(g_{i+k,k-2}, \mathfrak{a}_{i+k,k-1}) \bmod 2^p & = & 0. \end{array}\right\} \quad (3)$$

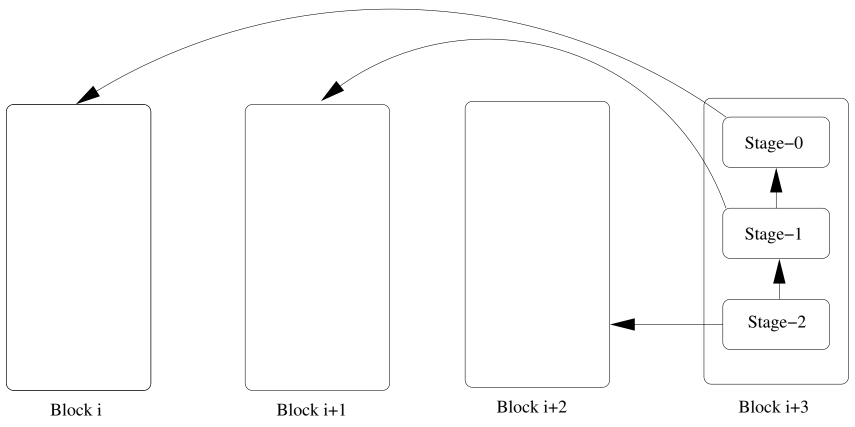

Figure 1: Schematic diagram of a 3-stage proof-of-work blockchain.

Here the parameter p determines the condition for mining permit. We discuss this in more details below.

Verification of proof-of-work:

$$\left.\begin{array}{rcl} g_{i+k,0} & < & t_{i+k,0}, \\ g_{i+k,1} & < & t_{i+k,1}, \\ \cdots & . & \cdots \\ g_{i+k,k-1} & < & t_{i+k,k-1}. \end{array}\right\} \tag{4}$$

A miner working on stage number j of block B_{i+k} ($j = 0, \ldots, k-1$, $i \geq 0$), chooses an address $\mathfrak{a}_{i+k,j}$ for which it knows the corresponding signing key. It starts the proof-of-work on stage number j by first assigning $\mathfrak{c}_{i+k,j}$ coins to the address $\mathfrak{a}_{i+k,j}$. If the miner is successful in completing the proof-of-work of stage number j of block B_{i+k}, then once the block gets on to the blockchain, the miner can use $\mathfrak{c}_{i+k,j}$ coins which is associated to the address $\mathfrak{a}_{i+k,j}$.

Mining permits: Obtaining mining permit for stage number j of block B_{i+k} requires the miner to compute $y_j = H_j(x, \mathfrak{a}_{i+k,j})$, where x is the digest of the previous proof-of-work computation (see (3)). The mining permit condition checks that $y_j \bmod 2^p = 0$. So, a miner proceeds as follows. It first generates an address $\mathfrak{a}$ to which it would like the reward of $\mathfrak{c}_{i+k,j}$ coins to be assigned. Then it computes $y = H_j(x, \mathfrak{a})$ and checks whether $y \bmod 2^p = 0$. If the condition holds, then the miner sets $\mathfrak{a}_{i+k,j}$ to be equal to $\mathfrak{a}$ and tries to complete the proof-of-work of stage number j. Assuming that H_j behaves like a random oracle, the value y is random. So, to obtain a mining permit, a miner generates a random number (based on x and $\mathfrak{a}$) and checks whether the mining permit condition holds. If the condition does not hold, then the miner can try to generate other random numbers with different values of $\mathfrak{a}$, until it obtains a y such that $y \bmod 2^p = 0$.

The value of the permit parameter p determines the chance of the mining permit condition being satisfied. On an average, generating about 2^p random numbers will yield one for which the mining permit condition holds. So, a single miner has to apply H_j about 2^p times to ensure the satisfaction of the mining permit condition. On the other hand, if around $2^{\mathfrak{n}}$ miners are simultaneously trying to satisfy the mining permit condition, then after each miner applies H_j about $2^{p-\mathfrak{n}}$ times, it is expected (under the simplifying assumption that the hash rates of the miners are more or less equal) that some miner will have satisfied the mining permit condition. So, if an individual miner is unable to satisfy the mining permit condition in about $2^{p-\mathfrak{n}}$ trials, it might as well give up, since it is likely that some other miner would have already satisfied this condition.

The parameter p needs to be judiciously chosen. The goal is to ensure that for each stage at least one of the miners is able to satisfy the condition of mining permit within a few trials of the corresponding hash function. So, a good choice is to have 2^p to be equal to the number of miners in the system. The number of miners, though, is not available from the blockchain. Instead, one may use the block number to define p, i.e., one may set $p = \lceil \log_2 \mathsf{bn} \rceil / c$ for some constant $c \geq 1$.

Start-up blocks: For the blockchain to start, the initial k blocks $B_0, \ldots, B_{k-1}$ need to be defined.

For $0 \leq i \leq k-1$,

$$B_i = \boxed{\begin{array}{l} i, \\ \mathsf{bdigest}_i, \\ t_i,\ \eta_i,\ \tau_i,\ \mathfrak{a}_i,\ \mathfrak{c}_i \end{array}}$$

The quantities $\mathsf{bdigest}_0, \ldots, \mathsf{bdigest}_{k-1}$ are defined as follows.

$$\left.\begin{array}{rcl} \mathsf{bdigest}_0 &=& H_0\left(0, t_0, \mathfrak{a}_0, \mathfrak{c}_0, \tau_0, \eta_0\right); \\ \mathsf{bdigest}_i &=& H_i\left(\mathsf{bdigest}_{i-1}, i, t_i, \mathfrak{a}_i, \mathfrak{c}_i, \tau_i, \eta_i\right), \quad 1 \leq i \leq k-1. \end{array}\right\} \tag{5}$$

Verification conditions for the proof-of-work of blocks $B_0, \ldots, B_{k-1}$ are the following.

$$\mathsf{bdigest}_i < t_i. \tag{6}$$

For $i = 0, \ldots, k-1$, block B_i assigns $\mathfrak{c}_i$ coins to address $\mathfrak{a}_i$. Note that we do not define k-stage proof-of-work for the start-up blocks. This can be done, but, does not seem to be useful. The blockchain would become operational only after the start-up blocks have been prepared. The issue of k-stage proof-of-work becomes relevant only after the blockchain becomes operational.

Stage completion time SCT: For $i \geq k$, the proof-of-work of block B_i has k stages. Each stage of the total proof-of-work has an independent target. The value of this target and the hash rate for the hash function used for the particular stage determine the time required to complete the proof-of-work of the stage. The total time to complete a stage is the sum of the times to obtain the mining permit and complete the proof-of-work of the stage. The goal is to ensure that all the stages require about the same time for the two tasks. We denote this as the stage completion time (SCT) which is defined to be T seconds. The value of T is to be defined as part of the design rules for the blockchain.

The parameters of the blockchain are to be chosen so that the main component of T is the time to complete the proof-of-work of the stage. The targets for the various stages are to be updated at regular intervals so that SCT of T seconds is maintained. For updating the target for stage j, $0 \leq j \leq k-1$, the actual times to complete stage j in a fixed number of previous blocks which are already in the blockchain are considered. We do not specify any particular rule for updation of the targets of the various stages. As long as the updation rule is able to ensure that each stage requires about T seconds, it can be fitted into our framework.

For a k-stage proof-of-work blockchain having T as the SCT, the block completion time $\mathfrak{T}$ for general blocks equals kT, i.e., the time to complete all the proof-of-work of the k stages in a general block is kT.

Difficulty: The difficulty of completing the proof-of-work of a stage of a block is a parameter which is computed as a function of the target for that particular stage of the block. We do not specify the exact method of computing difficulty. The only thing important is that given a value of the target for the stage, it should be possible to obtain the difficulty of the stage. The difficulty of a block is the sum of difficulties of the k stages in the block. This is a rough measure of the amount of work that has been put in to mine the block. Further, the difficulty of a contiguous sequence of blocks in the blockchain is the sum of the difficulties of all the blocks in the sequence.

The difficulty parameters for the various stages are dynamic quantities. Increasing or decreasing the difficulty parameter for any particular stage modifies the target value for that stage. Correspondingly, the time to complete the proof-of-work of that stage also changes. The protocol specifies the method of adjusting the difficulty parameter so that the stage completion time remains T seconds. If more miners begin to work on a particular stage, then the time to complete that stage would decrease. The difficulty parameter for that stage would be appropriately increased so that the SCT becomes T. Similarly, the difficulty parameter for a stage would be decreased if the time to complete that stage increases.

We note that the difficulty of the stages of a multi-stage blockchain cannot be directly compared to the difficulty of a single-stage blockchain. Rather, the goal of the difficulty parameters is to control the time to complete the proof-of-work, which is the SCT for a multi-stage blockchain and the time to mine a block for a single-stage blockchain. So, the meaningful comparison is between the time to mine a block for both types of blockchains. For a k-stage blockchain with SCT T seconds, the total time to mine a block is kT. One may compare such a blockchain with a single-stage blockchain whose block completion time is kT seconds.

3.2 Creation of New Blocks

For the blockchain to start operating, the initial k blocks, namely blocks $B_0, \ldots, B_{k-1}$ will have to be created. These start-up blocks can be used for creating some initial amounts of currency so that transactions become possible. The amounts of currency created during the preparation of the start-up blocks can be offered as initial coin offerings to prospective entities.

After the k start-up blocks have been created, general blocks can be created and added to the blockchain. General blocks can accommodate transactions. So, processing of transactions can proceed after the start-up blocks have been mined. Creating a general block requires completing all the proof-of-work of the k stages. The division of the proof-of-work into various stages gives rise to a cooperative way of creating blocks.

The mining of a general block can be completed in an incremental manner. To describe how this can be done, for block B_{i+k}, $i \geq 0$, we define the notion of incremental block information, $\mathsf{IBI}_{i+k,j}$, $j = 0, \ldots, k-2$.

$$\mathsf{IBI}_{i+k,0} = \boxed{\begin{array}{l} (i+k,0), \\ g_{i+k,0}, \\ \mathcal{L}_{i+k}, \\ t_{i+k,0},\ \eta_{i+k,0},\ \tau_{i+k,0},\ \mathfrak{a}_{i+k,0},\ \mathfrak{c}_{i+k,0} \end{array}}$$

$$\mathsf{IBI}_{i+k,1} = \boxed{\begin{array}{l} (i+k,1), \\ g_{i+k,1}, \\ t_{i+k,1},\ \eta_{i+k,1},\ \tau_{i+k,1},\ \mathfrak{a}_{i+k,1},\ \mathfrak{c}_{i+k,1} \end{array}}$$

$$\vdots \quad \vdots \quad \vdots$$

$$\mathsf{IBI}_{i+k,k-2} = \boxed{\begin{array}{l} (i+k,k-2), \\ g_{i+k,k-2}, \\ t_{i+k,k-2},\ \eta_{i+k,k-2},\ \tau_{i+k,k-2},\ \mathfrak{a}_{i+k,k-2},\ \mathfrak{c}_{i+k,k-2} \end{array}}$$

1. Given $\mathsf{bdigest}_i$ (from block B_i), it is possible to create $\mathsf{IBI}_{i+k,0}$.
2. For $j = 0, \ldots, k-2$, given $\mathsf{IBI}_{i+k,j}$ and $\mathsf{bdigest}_{i+j+1}$ (from block B_{i+j+1}), it is possible to create
   - $\mathsf{IBI}_{i+k,j+1}$ if j is in $\{0, \ldots, k-3\}$; and
   - block B_{i+j+2} if $j = k-2$.

For $j = 0, \ldots, k-2$, the verification of the proof-of-work for $\mathsf{IBI}_{i+k,j}$ consists of verifying that $g_{i+k,j}$ has indeed been correctly computed as given by (2) and that the verification of mining permit given in (3) and the verification of proof-of-work given in (4) hold.

The $\mathsf{IBI}_{i+k,j}$'s are created and relayed by miners to the whole network. So, along with transactions and completed blocks, the nodes in the network are also responsible for relaying the IBI's. However, the nodes do not add the $\mathsf{IBI}_{i+k,j}$'s to the local copies of the blockchain. Once a miner completes the proof-of-work of the entire block B_{i+k}, it relays the block to the entire network. Nodes in the network then append the newly found block to the local copies of their blockchain. So, the blockchain consists of only complete blocks and the blockchain does not keep track of incremental block information of different stages.

Let us now consider how miners can cooperate amongst themselves in creating a block. Recall that the SCT is T seconds. Consider the creation of the first general block B_k after the initial blocks $B_0, \ldots, B_{k-1}$ have been created and added to the blockchain. The proof-of-work for the first stage of B_k depends upon B_0 and is completed in about T seconds. After the proof of work of the first stage of B_k is complete, the miner which successfully completes this stage may release the incremental block information $\mathsf{IBI}_{k,0}$ to the network. This miner as well as other miners can then build upon $\mathsf{IBI}_{k,0}$ to try and complete the proof-of-work of the second stage which depends upon B_1. Simultaneously, other miners can start creating the proof-of-work for the first stage of block B_{k+1}, which also depends upon B_1. After T more seconds, both $\mathsf{IBI}_{k,1}$ and $\mathsf{IBI}_{k+1,0}$ will be available. At this point, work on creating the proof-of-work for the third stage of B_k, the second stage of B_{k+1} and the first stage of B_{k+2} can start simultaneously. Continuing in this fashion, the complete proof-of-work for block B_k will be completed in about kT seconds. Since the mining of block B_{k+1} started T seconds after block B_k and the complete proof-of-work for block B_{k+1} also requires about kT seconds, at the end of $(k+1)T$ seconds, the mining of block B_{k+1} will be completed. Similarly, at the end of $(k+2)T$ seconds, the mining of block B_{k+2} will be completed and so on.

From the structure of the k-stage blockchain, we have that for $i \geq 0$, $\mathsf{bdigest}_{i+k}$ is required for completing stage number $k-1$ of block B_{i+k+1}, stage number $k-2$ of block B_{i+k+2}, …, stage number

0 of block B_{i+2k}. So, after the completion of block B_{i+k} (which provides the value of $\mathsf{bdigest}_{i+k}$), the following activities can start simultaneously.

- Creation of proof-of-work for stage number $k-1$ of block B_{i+k+1}; on completion, B_{i+k+1} is added to the blockchain.
- Creation of proof-of-work for stage $k-2$ of block B_{i+k+2}; on completion, $\mathsf{IBI}_{i+k+2,k-2}$ is released to the network.
- ...
- Creation of proof-of-work for stage 0 of block B_{i+2k}; on completion, $\mathsf{IBI}_{i+2k,0}$ is released to the network.

The above description is reminiscent of typical pipelining scenario in hardware architectures. In fact, the goal of the multi-stage design is to translate benefits of pipelining to block mining. For $k = 3$ and $i \geq 3$, the quantities which can be simultaneously computed are shown in Figure 2.

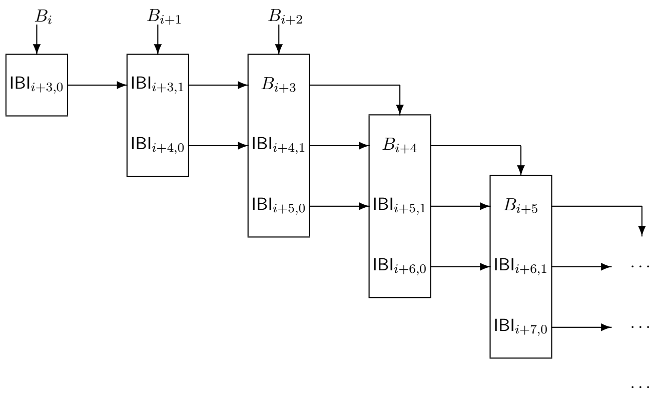

Figure 2: Illustration of quantities which can be computed simulatenously for $k = 3$.

Incentive for incremental mining: Suppose a miner $\mathcal{M}$ is able to complete stage 0 of block $i + k$. It has the option of immediately releasing $\mathsf{IBI}_{i+k,0}$. If it does this, other miners can possibly work on $\mathsf{IBI}_{i+k,0}$ and build the proof-of-work for the subsequent stages and eventually complete the block. As part of $\mathsf{IBI}_{i+k,0}$, miner $\mathcal{M}$ has also provided the address $\mathfrak{a}_{i+k,0}$. This address has been computed by applying the hash function H to a verification key whose signing key is known to $\mathcal{M}$. If the part $\mathsf{IBI}_{i+k,0}$ gets completed to block B_{i+k}, then $\mathfrak{c}_{i+k,0}$ coins will be assigned to the address $\mathfrak{a}_{i+k,0}$. Since $\mathcal{M}$ knows the signing key corresponding to the address $\mathfrak{a}_{i+k,0}$, it will obtain control of $\mathfrak{c}_{i+k,0}$ coins. So, by releasing $\mathsf{IBI}_{i+k,0}$, miner $\mathcal{M}$ has locked in a certain amount of coins which it has a chance of obtaining in the future.

There is nothing special about $\mathsf{IBI}_{i+k,0}$. The above argument applies to $\mathsf{IBI}_{i+k,j}$ for all $i \geq 0$ and $j = 0, \ldots, k-2$. If a miner is able to complete the proof-of-work for stage j of block B_{i+k}, then it has the option of immediately releasing $\mathsf{IBI}_{i+k,j}$. This contains the address $\mathfrak{a}_{i+k,j}$ whose secret key is known to $\mathcal{M}$. If $\mathsf{IBI}_{i+k,j}$ gets completed to a block B_{i+k}, then $\mathfrak{c}_{i+k,j}$ coins gets assigned to $\mathfrak{a}_{i+k,j}$ giving $\mathcal{M}$ control over these coins.

Alternatively, miner $\mathcal{M}$ could withold $\mathsf{IBI}_{i+k,j}$ and try to complete the mining of the entire block B_{i+k} by itself. The use of mining permits disincentivises such behaviour. To complete the proof-of-work of the other stages, $\mathcal{M}$ would need to satisfy the mining permit conditions for each of these stages. For each stage, it would require about 2^p invocations of the hash function of that stage to satisfy the mining permit condition. This puts a significant hurdle upon $\mathcal{M}$ to proceed over all the stages. So, $\mathcal{M}$ runs the risk that the other miners can together mine block B_{i+k} earlier. Then, the entire work done by $\mathcal{M}$ will be wasted and result in no return. It would be better for $\mathcal{M}$ to release stage-wise proof-of-work.

The above combination of incentive and disincentive mechanisms encourage miners to cooperate to mine a complete block. More accurately, the above scenario captures both competitive and cooperative behaviour. Miners compete with each other to complete the mining of the individual stages. By releasing the proof-of-work of the individual stages, the miners cooperate with each other to complete the mining of an entire block.

Cooperative block mining: Assuming that an individual miner will aim to complete the proof-of-work for a single stage, in the steady state, one may expect the miners to get divided into k groups and work simultaneously as follows.

- A group working on obtaining a proof-of-work for stage $k-1$ of block B_i;
- A group working on obtaining a proof-of-work for stage $k-2$ of block B_{i+1};
- ...
- A group working on obtaining a proof-of-work for stage 0 of block B_{i+k-1};

The formation of the groups is not imposed extraneously. The cooperative process of the multi-stage mining will itself incentivise miners to work on individual stages, thus leading to the formation of the groups. For $k = 3$, the above grouping is illustrated in Figure 3. In contrast, for a single stage blockchain, all miners would be competing with each other to mine block B_i.

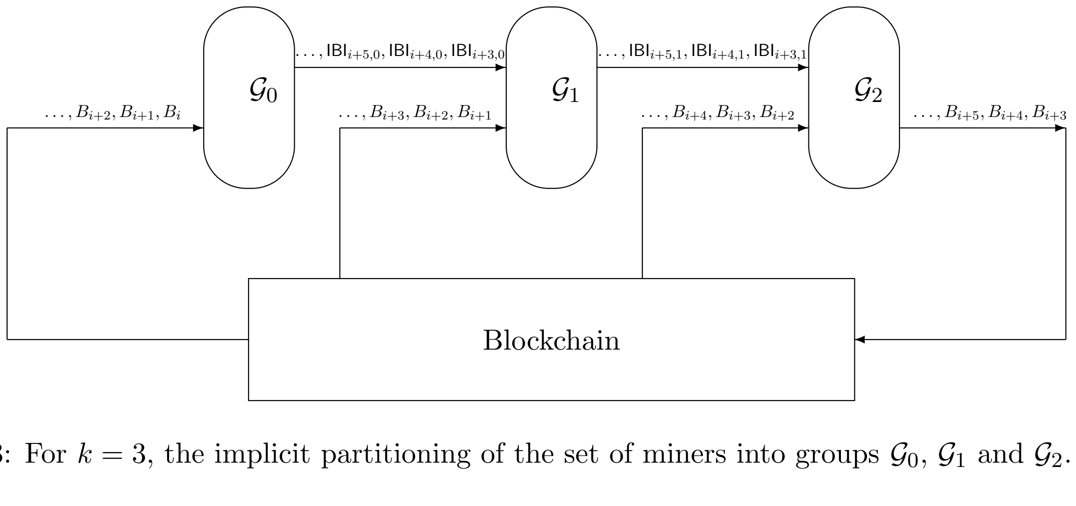
$\mathcal{G}_0$
$\mathcal{G}_1$
$\mathcal{G}_2$
$\ldots, \mathsf{IBI}_{i+5,0}, \mathsf{IBI}_{i+4,0}, \mathsf{IBI}_{i+3,0}$
$\ldots, \mathsf{IBI}_{i+5,1}, \mathsf{IBI}_{i+4,1}, \mathsf{IBI}_{i+3,1}$
$\ldots, B_{i+2}, B_{i+1}, B_i$
$\ldots, B_{i+3}, B_{i+2}, B_{i+1}$
$\ldots, B_{i+4}, B_{i+3}, B_{i+2}$
$\ldots, B_{i+5}, B_{i+4}, B_{i+3}$
Blockchain

Figure 3: For $k = 3$, the implicit partitioning of the set of miners into groups $\mathcal{G}_0$, $\mathcal{G}_1$ and $\mathcal{G}_2$.

3.3 Hardware Incompatible Hash Functions

Hash functions $H_0, \ldots, H_{k-1}$ are to be used for generating the proof-of-work of the k stages. It is possible to choose all of these hash functions to be the same function. Our analysis of improved transaction processing in Section 4 does not require the hash functions to be different. Also, we perform a security analysis of a k-stage blockchain under the assumption that all the hash functions are same. So, benefits of using a multi-stage blockchain can be obtained by using a single hash function for obtaining proof-of-work of all the stages. On the other hand, having access to multiple hash functions can accentuate the cooperative nature of a multi-stage blockchain. Below we explain this further.

We say that two hash functions G and G' are hardware incompatible if it is not possible to easily modify or reconfigure a fast hardware for computing one of the functions to obtain a fast hardware for computing the other function. A set of hash functions is said to be hardware incompatible if the hash functions in the set are pairwise hardware incompatible. Consider the following set of hash functions.

$$\{\text{SHA-2}, \text{BLAKE}, \text{Grøstl}, \text{JH}, \text{Keccak}, \text{Skein}\}.$$

SHA-2 has been standardised by NIST and is used in Bitcoin. Keccak has also been standardised by NIST and named SHA-3. The other hash functions in the above list are the finalists in the NIST hash function competition. There are no known security flaws in any of these functions. Each of the above hash functions follows a different design strategy. This makes their designs fundamentally different. Consequently, a dedicated special purpose hardware for one of the hash functions cannot be easily modified to compute another hash function. So, the above provides a set of hardware incompatible hash functions. If it is desired to extend this list, then one may consider the hash functions that were considered in Round 2 of the NIST hash function competition. Again, there are no known security flaws in the Round 2 hash functions and the designs are also different. So, for a value of k around 10, it is possible to obtain a set containing k hardware incompatible hash functions.

Suppose, a set of μ hardware incompatible hash functions $G_0, \ldots, G_{\mu-1}$ is available and it is desired to have the number of stages k to be greater than μ. Then one may set $H_i = G_{i \bmod \mu}$ for $i = 0, \ldots, k-1$.

In the context of multi-stage blockchain, the advantage of using hardware incompatible hash functions is that it would require a huge investment from an individual miner to obtain very fast special purpose hardware for *all* the hash functions $H_0, \ldots, H_k$. What is more likely is that individual miners will focus on obtaining special purpose hardware for one hash function. This will provide it with a computational leverage over some of the stages, but, not all the stages.

In such a situation, a miner will focus on completing the proof-of-work of the stages for which it has special hardware. Since, other miners have special hardware for other stages, this will incentivise a miner to release proof-of-work of individual stages thus promoting the cooperative behaviour in the mining of a block.

4 Improved Transaction Processing

We state results which show how a blockchain with a k-stage proof-of-work processes transactions faster than a blockchain with a single-stage proof-of-work.

Proposition 1. *A blockchain with a k-stage proof-of-work processes transactions about k times faster than a blockchain with a single-stage proof-of-work under the assumption that the block completion times for both blockchains are equal.*

Proof. First consider the blockchain with a k-stage proof-of-work. We consider a steady state scenario, i.e., the scenario where the blockchain has been initialised with the start-up blocks and further sufficiently many blocks have been added to the blockchain. At such a point of time, proof-of-work for any stage of a block takes about T seconds and the block completion time is $\mathfrak{T} = kT$, where T is the SCT.

For $i \geq 0$, the work on creating block B_{i+k+1} starts about T seconds after the work on creating block B_{i+k} starts. Since completing both blocks require kT seconds, block B_{i+k+1} will be added to the blockchain about T seconds after block B_{i+k} is added to the blockchain. So, blocks will be added to the blockchain at the rate of one block in about T seconds.

Assume that on an average, $\mathfrak{m}$ transactions are accommodated in a block. So, the rate at which transactions get added to the blockchain is $\mathfrak{m}/T$ transactions per second.

Recall that for a single stage blockchain, the rate at which transactions are added to the blockchain is about $\mathfrak{m}/\mathfrak{T} = \mathfrak{m}/(kT)$ seconds. So, the rate at which a k-stage blockchain adds transactions to the blockchain is about k times faster than the rate at which a single-stage blockchain adds transaction to the blockchain. □

Suppose that the block completion time $\mathfrak{T}$ is fixed. A theoretical consequence of Proposition 1 is that the rate of transaction processing can be improved to a great degree by choosing a very high value of k. There is, however, the issue of network delays. Since $\mathfrak{T} = kT$, increasing k decreases the value of T. The blocks and the IBI's which are released by miners require a certain amount of time to reach the entire network. This time is the network latency. The SCT T cannot be lower than the network latency. This puts a restriction on the maximum value of k and hence the extent to which transaction processing can be improved by using a k-stage proof-of-work blockchain.

Proposition 2. *Suppose that a transaction is considered confirmed if ν blocks are added to the blockchain after the block containing the transaction. The confirmation time for a blockchain with a k-stage proof-of-work is a fraction $(k+\nu)/(k(\nu+1))$ of the confirmation time for a blockchain with a single-stage proof-of-work under the assumption that the block completion times for both blockchains are equal.*

Proof. As in the proof of Proposition 1, we consider the steady state scenario where T is the SCT and the block completion time is $\mathfrak{T} = kT$. So, blocks are added to the blockchain at the rate of about one block every T seconds.

Consider the block containing the transaction. Completing the proof-of-work for this block requires about kT seconds. Once this block is added to the blockchain, the next ν blocks are added to the blockchain in about νT seconds. So, the total time for confirmation of the block is about $kT + \nu T$ seconds.

For the blockchain with the single-stage proof-of-work, completing the proof-of-work for the block containing the transaction takes $\mathfrak{T}$ seconds. Each of the ν further blocks require about $\mathfrak{T}$ seconds to be added to the blockchain. So, the total time for confirmation of the transaction is about $(\nu+1)\mathfrak{T} = k(\nu+1)T$.

So, the confirmation time for a transaction on a k-stage blockchain is a fraction $(k+\nu)/(k(\nu+1))$ of the confirmation time for a transaction on a single-stage blockchain. □

Energy consumption: The amount of energy consumption per transaction is an important parameter. Note that the number of transactions that can be accommodated depends upon the size of the block and is independent of the number of stages. Further, the total energy required to mine a block is determined by the block completion time $\mathfrak{T}$ and the hash rate. The block completion time is independent of the number of stages. Under the simplifying assumption that the hash rate does not depend upon the number of stages, the energy consumption per transaction is independent of the number of stages.

By incentivising cooperative behaviour among the miners, a multi-stage blockchain has the potential to reduce the overall energy requirement for mining a block. In this case, the energy consumption per transaction may actually reduce compared to a single-stage blockchain.

5 Security

Suppose that $H_0 = \cdots = H_{k-1}$. The following result shows that the condition for a 51% attack on a multi-stage blockchain is the same as that for a single-stage blockchain.

Proposition 3. *Consider a blockchain with a k-stage proof-of-work where $H_0 = \cdots = H_k$ and the targets for all the stages of a block are equal to t. Suppose there is an adversary with hash rate ρ_a and the hash rate for all the honest miners is ρ. Suppose that block B_r is the current header of the blockchain, where r is sufficiently greater than k so that steady state behaviour of the blockchain may be assumed. Let t_a be the expected time required for the adversary to replace block B_r and t be the expected time required by the honest miners to add the next block to the blockchain. Then $\mathsf{t}_a < \mathsf{t}$ if and only if $\rho_a > \rho$.*

Proof. Since the target for all the stages is t, the number of hash function calls to complete the proof-of-work of any particular stage is about $2^{n_1}/t$, where n_1 is the size of the digest for the hash function H_0. Also, to satisfy the condition for mining permit for each stage, about 2^p hash function calls are required.

The adversary's goal is to replace the block B_r which is the present header of the blockchain. Let the new block be B'_r. To mine B'_r, the adversary will have to satisfy the conditions of mining permit and complete the proof-of-work of all the k stages. Since a single stage requires about $2^p + 2^{n_1}/t$ calls, the k stages of B'_r will require about $k\left(2^p + 2^{n_1}/t\right)$ hash function calls. The hash rate of the adversary is ρ_a and so the adversary can complete the mining of the block B'_r in expected time $k\left(2^p + 2^{n_1}/t\right)/\rho_a$.

While the adversary is trying to mine the block B'_r, the honest miners are trying to add the next block B_{r+1} to the blockchain. Due to the multi-stage nature of the blockchain, at the time block B_r was added to the blockchain, the proof-of-work of the stages numbered 0 to $(k-2)$ of block B_{r+1} have already been completed. So, the honest miners are trying to complete the proof-of-work of stage numbered $(k-1)$ of block B_{r+1}. This requires about 2^p calls to satisfy the mining permit condition and about $2^{n_1}/t$ hash function calls for the proof-of-work.

The hash rate of all the honest miners is ρ. Again due to the multi-stage nature of the blockchain, this hash rate is distributed more or less equally to k tasks, namely, the proof-of-work of the stage number $k-1$ of block B_{r+1}, the proof-of-work of stage number $k-2$ of block B_{r+2}, ..., the proof-of-work of stage numbered 0 of block B_{r+k}. So, the effective hash rate that is employed to complete the proof-of-work of stage numbered $(k-1)$ of block B_{r+1} is ρ/k. With a hash rate of ρ/k, the expected time to complete the $2^{n_1}/t$ hash function calls is $\left(2^p + 2^{n_1}/t\right)/(\rho/k)$.

So, the condition required for the expected time for the adversary to mine block B^r to be less than the expected time for the honest miners to add block B_{r+1} to the blockchain is $k\left(2^p + 2^{n_1}/t\right)/\rho_a < \left(2^p + 2^{n_1}/t\right)/(\rho/k)$ which holds if and only if $\rho_a > \rho$. □

The result and the proof assume that the hash functions of all the stages are the same. If a set of hardware incompatible hash functions is used, then the adversary's task becomes more difficult, since in this case the adversary will have to obtain significant computational capability for all the hash functions. So, using hardware incompatible hash functions can actually lead to security improvement.

5.1 Selfish Mining

Let us consider the possibility of selfish mining in the context of multi-stage blockchain. Suppose a miner successfully completes stage number $(k-1)$ of the present block B_r before other miners. At this point, stage number $(k-2)$ of block B_{r+1} has been mined by the honest users and $\mathsf{IBI}_{r+1,k-2}$ has been made publicly available. To complete the proof-of-work of stage number $(k-1)$ of block B_{r+1}, the block B_r (in particular, $\mathsf{bdigest}_r$) is required. So, the selfish miner can adopt the following strategy. Instead of releasing block B_r, it keeps this block private and attempts to complete stage number $(k-1)$ of block B_{r+1} ahead of the honest miners. If it is successful, then by the time it completes stage number $(k-1)$ of block B_{r+1}, the honest miners have completed the proof-of-work of stage number $(k-2)$ of block B_{r+2} and made $\mathsf{IBI}_{r+2,k-2}$ public. The selfish miner can continue with this strategy. The following result puts a bound on the number of such blocks that the selfish miner can privately mine before being forced to release its private chain.

Proposition 4. *The selfish mining strategy for a k-stage blockchain can continue consecutively for up to k blocks.*

Proof. Assume that selfish mining strategy is employed for i blocks, i.e., a selfish miner completes stage number $(k-1)$ of blocks $B_r, \ldots, B_{r+i-1}$ ahead of the honest miners. The honest miners instead obtain blocks $B'_r, \ldots, B'_{r+i-1}$.

From the definition of k-stage blockchain, the block B_r is required to complete the proof-of-work of stage number 0 of block B_{r+k}. Since B_r is not released by the selfish miner, the honest miners cannot use B_r to complete the proof-of-work of stage number 0 of block B_{r+k}. Instead, the honest miners use block B'_r, for the proof-of-work of stage number 0 of block B_{r+k}. Similarly, the honest miners use block B'_{r+j} for the proof-of-work of stage number j of block B_{r+k}.

If $i > k$, then following the selfish mining strategy, the selfish miner would have to complete the proof-of-work of stage number $(k-1)$ of block B_{r+k}. For this, it has to use $\mathsf{IBI}_{r+k,k-2}$ that has been obtained and made public by the honest miners. Note however, that $\mathsf{IBI}_{r+k,k-2}$ has been prepared using blocks $B'_r, \ldots, B'_{r+k-2}$ which are different from the blocks $B_r, \ldots, B_{r+k-2}$ that the selfish miner has already mined privately. Since the selfish miner plans to release blocks $B_r, \ldots, B_{r+k-2}$, it is useless to it to use the publicly available $\mathsf{IBI}_{r+k,k-2}$. So, it cannot employ the selfish mining strategy to complete the proof-of-work of stage number $(k-1)$ of block B_{r+k}. Consequently, the maximum number of consecutive blocks for which the selfish mining strategy can be employed is k. □

Mining permit as a deterrent for selfish mining: Consider the situation where the selfish miner witholds block B_r and attempts to complete the proof-of-work of stage number $(k-1)$ of block B_{r+1}. To do this, it first needs to satisfy the condition of mining permit for this stage. This requires the selfish miner to expend about 2^p calls of H_{k-1}. During the time the selfish miner spends in these calls, there is a good chance that one of the honest miners completes block B_r and releases it to the network. In that case, the selfish miner can no longer claim the coins for having successfully completed block B_r earlier. More generally, to successfully carry out a selfish mining strategy for i blocks, the selfish miner has to satisfy the mining permit conditions for the last stages of these i blocks. This will require it to expend about $i2^p$ calls of H_{k-1}. Such a requirement provides a deterrent for the selfish mining strategy.

6 Comparison and Discussion

The major goal of multi-stage blockchain is to speed-up transaction processing and the confirmation time for transactions. Propositions 1 and 2 quantify the speed-ups of a multi-stage proof-of-work blockchain over a single-stage proof-of-work blockchain. Let us take some concrete figures to understand the extent of the speed-up.

Example 1: As mentioned earlier, in Bitcoin, the time to mine a block is designed to be about 600 seconds, i.e., $\mathfrak{T} = 600$ seconds. So, blocks are added to the Bitcoin blockchain at the rate of one block per ten minutes. Bitcoin suggests that a transaction may be considered to be confirmed if five further blocks have been added to the blockchain, i.e., $\nu = 5$. This means that Bitcoin requires about one hour for a transaction to be confirmed. Suppose that in a multi-stage setting, the values of $\mathfrak{T}$ and ν are kept unchanged. For a k-stage proof-of-work blockchain and SCT T, we have $kT = \mathfrak{T} = 600$. Suppose, we choose $k = 10$. Then $T = 60$ seconds. So, blocks will be added to the blockchain at the rate of one block per minute. This is ten-fold improvement over the Bitcoin blockchain. With $\nu = 5$, the time for confirming a transaction is $kT + \nu T = 600 + 300 = 900$ seconds, i.e., a transaction will be considered to be confirmed in about 15 minutes. This is a four-fold improvement over the confirmation time for Bitcoin.

Example 2: In Litecoin, the time to mine a block is designed to be about 150 seconds, i.e., $\mathfrak{T} = 150$ seconds. So, blocks are added to Litecoin at the rate of one block per two-and-half minutes. Suppose that a transaction is considered to be confirmed if nine further blocks have been added to the blockchain, i.e., $\nu = 9$. So, a transaction will be confirmed after about $150 \times 10 = 1500$ seconds (25 minutes). Again, suppose that in a multi-stage setting, the values of $\mathfrak{T}$ and ν are kept unchanged. For a k-stage proof-of-work blockchain and SCT T, we have $kT = \mathfrak{T} = 150$. Suppose we choose $k = 5$ so that $T = 30$. Then blocks will be added to the blockchain at the rate of one block per half-minute. This is a five-fold improvement. The time for confirmation of a transaction is $kT + \nu T = 150 + 270 = 420$ seconds (7 minutes). This is about a four-fold improvement in confirmation time.

The parameters $\mathfrak{T}$ and ν determine security. So, using a multi-stage blockchain speeds up both confirmation time for transactions and the overall rate at which transactions are processed by the blockchain without reducing security.

In the above analysis, we have taken $T = 60$ and $T = 30$ for Bitcoin and Litecoin respectively. A lower bound on the value of T is given by the network delay. A comprehensive work on network delay for Bitcoin has been done [4] and the site `http://bitcoinstats.com/network/propagation/` provides statistics of delay in the Bitcoin network. The value of $T = 60$ for Bitcoin respects these network delays. We note that multi-stage blockchain does not provide a method for handling network delays. If it can be ensured that network delay is small, then T can be taken to be much smaller leading to further increase in the rate of transaction processing and lowering of the time for transaction confirmation.

A number of proposals have been put forward for improving various aspects of a proof-of-work blockchain. Some of these ideas are complementary to the idea of using a multi-stage proof-of-work. In particular, we mention two previous ideas and indicate how a multi-stage proof-of-work can be incorporated into these ideas.

Fruitchain: The concept of Fruitchain was suggested in [12]. The main goal is to ensure a fair share of reward to the miners, i.e., a miner possessing a fraction f of the total hash capability, obtains essentially a fraction f of the total reward and so disincentivises selfish mining. Additionally, the protocol ensures that miners get paid more often so that mining pools become redundant.

A multi-stage blockchain also disincentivises selfish mining. Further, by dividing the block reward into several parts a multi-stage blockchain ensures that a greater number of participants receives proceeds from the mining of a block. The cooperative behaviour of the miners in a multi-stage blockchain also makes mining pools unnecessary.

A closer look at the structure of Fruitchain shows that it consists of two kinds of mining activity, namely mining blocks and mining fruits. The fruits are parts of a block and occupy about 8% to 10% of the space of a block. The difficulties of fruit and block minings are independent. It is conceivable that the task of block mining in a Fruitchain can be decomposed into multiple stages. This would lead to a multi-stage proof-of-work Fruitchain. Such a protocol has the potential to combine the benefits of both the Fruitchain structure along with that of multi-stage proof-of-work. Exploring such a combination is a possible future work.

Sharding: The goal of sharding based blockchain is to improve transaction processing. See for example [10, 15]. A sharding protocol provides facility for multiple committees of nodes to process transactions in parallel. The selection of committees take place autonomously and in a verifiable manner. Since transactions are processed in parallel, the overall rate of transaction processing improves.

In the steady state, the miners in a k-stage blockchain get divided (also autonomously) into k groups with one group working on a particular stage. The formation of the groups is an outcome of the cooperative process of a multi-stage blockchain and is helped by the requirement of obtaining mining permits. This can also be considered to be a form of sharding. The sharding is over the various stages required for completing the proof-of-work for a whole block.

The idea of a multi-stage blockchain is complementary to sharding protocols where transactions are processed in parallel. In sharding protocols, blocks are still required to be mined. A combination of sharding protocol and multi-stage blockchain would require the block mining process to be multi-stage while the top level sharding of transaction processing is inherited from the sharding protocol. Potentially such a combination will combine the benefits of both transaction level sharding and multi-stage blockchain. Again, exploring this idea is a possible future work.

7 Concluding Remarks

This paper has introduced a variant of decentralised, trustless, permissionless blockchain where the proof-of-work for mining a block is divided into multiple stages. Several advantages, including improved transaction processing and a more equitable distribution of rewards, have been highlighted.

The discussion in the paper has been theoretical. It is of interest to know how the concept actually performs in practice. This would require actually implementing a cryptocurrency based on a multi-stage proof-of-work blockchain and placing it in the public domain for real world adoption. Such a work would require a major effort by implementors. We hope that the idea of multi-stage proof-of-work blockchain will be found interesting enough by practitioners to motivate them to implement such a cryptocurrency.

We recall an observation from [3] which mentions that major advances in improving transaction processing rate and confirmation time of proof-of-work blockchain require a “basic rethinking of technical approaches”. The idea of multi-stage proof-of-work blockchain may be considered to be one such approach.

Acknowledgement

The author would like to thank Sanjay Bhattacherjee for many fruitful discussions on blockchains in general and cryptocurrency in particular. Thanks to Alfred Menezes for providing helpful comments. Also, we would like to thank the reviewers for their comments.

References

[1] Mustafa Al-Bassam, Alberto Sonnino, Shehar Bano, Dave Hrycyszyn, and George Danezis. Chainspace: A sharded smart contracts platform. In *25th Annual Network and Distributed System Security Symposium, NDSS 2018, San Diego, California, USA, February 18-21, 2018*. The Internet Society, 2018.

[2] Iddo Bentov, Rafael Pass, and Elaine Shi. Snow white: Provably secure proofs of stake. *IACR Cryptology ePrint Archive*, 2016:919, 2016.

[3] Kyle Croman, Christian Decker, Ittay Eyal, Adem Efe Gencer, Ari Juels, Ahmed E. Kosba, Andrew Miller, Prateek Saxena, Elaine Shi, Emin Gün Sirer, Dawn Song, and Roger Wattenhofer. On scaling decentralized blockchains - (A position paper). In Jeremy Clark, Sarah Meiklejohn, Peter Y. A. Ryan, Dan S. Wallach, Michael Brenner, and Kurt Rohloff, editors, *Financial Cryptography and Data Security - FC 2016 International Workshops, BITCOIN, VOTING, and WAHC, Christ Church, Barbados, February 26, 2016, Revised Selected Papers*, volume 9604 of *Lecture Notes in Computer Science*, pages 106–125. Springer, 2016.

[4] Christian Decker and Roger Wattenhofer. Information propagation in the bitcoin network. In *13th IEEE International Conference on Peer-to-Peer Computing, IEEE P2P 2013, Trento, Italy, September 9-11, 2013, Proceedings*, pages 1–10. IEEE, 2013.

[5] Ittay Eyal and Emin Gün Sirer. Majority is not enough: bitcoin mining is vulnerable. *Commun. ACM*, 61(7):95–102, 2018.

[6] Yossi Gilad, Rotem Hemo, Silvio Micali, Georgios Vlachos, and Nickolai Zeldovich. Algorand: Scaling byzantine agreements for cryptocurrencies. In *Proceedings of the 26th Symposium on Operating Systems Principles, Shanghai, China, October 28-31, 2017*, pages 51–68. ACM, 2017.

[7] Aggelos Kiayias, Alexander Russell, Bernardo David, and Roman Oliynykov. Ouroboros: A provably secure proof-of-stake blockchain protocol. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part I*, volume 10401 of *Lecture Notes in Computer Science*, pages 357–388. Springer, 2017.

[8] Neal Koblitz and Alfred J. Menezes. Cryptocash, cryptocurrencies, and cryptocontracts. *Des. Codes Cryptogr.*, 78(1):87–102, 2016.

[9] Yoad Lewenberg, Yonatan Sompolinsky, and Aviv Zohar. Inclusive block chain protocols. In Rainer Böhme and Tatsuaki Okamoto, editors, *Financial Cryptography and Data Security: 19th International Conference, FC 2015, San Juan, Puerto Rico, January 26-30, 2015, Revised Selected Papers*, pages 528–547, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.

[10] Loi Luu, Viswesh Narayanan, Chaodong Zheng, Kunal Baweja, Seth Gilbert, and Prateek Saxena. A secure sharding protocol for open blockchains. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, pages 17–30, New York, NY, USA, 2016. ACM.

[11] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, `https://bitcoin.org/bitcoin.pdf`. 2009.

[12] Rafael Pass and Elaine Shi. Fruitchains: A fair blockchain. In Elad Michael Schiller and Alexander A. Schwarzmann, editors, *Proceedings of the ACM Symposium on Principles of Distributed Computing, PODC 2017, Washington, DC, USA, July 25-27, 2017*, pages 315–324. ACM, 2017.

[13] Yonatan Sompolinsky and Aviv Zohar. Secure high-rate transaction processing in bitcoin. In Rainer Böhme and Tatsuaki Okamoto, editors, *Financial Cryptography and Data Security: 19th International Conference, FC 2015, San Juan, Puerto Rico, January 26-30, 2015, Revised Selected Papers*, pages 507–527, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg.

[14] Yonatan Sompolinsky and Aviv Zohar. Phantom: A scalable blockdag protocol. Cryptology ePrint Archive, Report 2018/104, 2018. `https://eprint.iacr.org/2018/104`.

[15] Mahdi Zamani, Mahnush Movahedi, and Mariana Raykova. Rapidchain: Scaling blockchain via full sharding. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018*, pages 931–948. ACM, 2018.