Multi-Stage Proof-of-Work Blockchain

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

June 3, 2019

Abstract

We introduce a new variant of decentralised, trustless, permissionless proof-of-work blockchain. The main novelty of the new variant is a multi-stage proof of work which is analogous to multi-stage pipelining used in hardware architectures. Among the possible advantages of using a multi-stage proof-of-work blockchain are the following.

- Reduction of the time for confirmation of a transaction and speeding up the overall rate of transactions processing *without* reducing the time for mining a block.
- Encourage cooperative behaviour among the miners so that the reward for mining a block is shared by a number of miners.
- Use of hardware incompatible hash functions for various stages so that it becomes very difficult for a single entity to attain major computational advantage over all the stages of the block mining.
- Improve security by making 51% attacks more difficult to achieve and by providing resilience to selfish mining attacks.

We believe that the new blockchain structure mitigates the problem of scalability without compromising security. By enforcing cooperative behaviour among the miners, reward for mining a block is more equitably distributed. This, in turn, will help in ensuring participation by a greater number of entities in the overall mining activity.
Keywords: blockchain, proof-of-work, pipelining, Bitcoin.

1 Introduction

Bitcoin was launched by Satoshi Nakamoto in 2009 [9]. 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 60 billion US Dollars.

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. 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. We note, however, that not all cryptocurrencies are based on blockchain (an example is XRP of Ripple) and that not all blockchains operate in a trustless and permissionless model (an example is Quorum).

Presently, several problems have been identified with the performance of Bitcoin. A major issue is that of scalability. The design of Bitcoin inherently limits the number of payment transactions that can be processed per second. Also, the confirmation time for a transaction is itself quite long. These two factors limit the applicability of Bitcoin as a currency for regular use. A second issue is that of centralisation. Transactions are confirmed by entities called miners who employ expensive special purpose hardware for so-called block mining. The high cost of block mining has led to the formation of mining pools. As things stand, block mining is done only by a small number of mining pools. This has led to fears of centralisation which cuts at one of the basic premises of Bitcoin. Several proposal have been made in the literature for overcoming these problems. We mention some of these after providing an overview of our main contribution.

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 $i \geq k$. The proof-of-work for mining B_i can start immediately after block B_{i-k} has been mined. Stage numbered 0 of the proof-of-work for mining B_i depends upon block B_{i-k}; stage numbered 1 of the proof-of-work for mining B_i depends upon block B_{i-k+1} and the completion of stage numbered 0; and so on until stage numbered $k-1$ of the proof-of-work is taken up after mining of block B_{i-1} and the completion of stage numbered $k-2$.

The target (and difficulty) of each stage is determined by the network in a manner such that the completion time for all the stages is more or less the same. We denote this parameter as the stage completion time T. So, the entire proof-of-work for a block takes kT seconds while blocks get added to the blockchain at the rate of one block per T seconds. This leads to two advantages. First, the overall transaction processing rate of the blockchain improves. Second, the confirmation time for a transaction also improves. Suppose that a transaction is considered to be confirmed if the block containing it is mined and a further ν blocks are added to the blockchain. Then the transaction confirmation time is $kT + \nu T$ seconds which is an improvement over νkT seconds. The idea underlying the multi-stage blockchain is similar to pipelined computations in hardware architectures.

The rate at which transactions are processed by Bitcoin is limited by several factors – the rate at which blocks are added to the blockchain, the network delay in propagating a block, and the size of a block. A multi-stage proof-of-work blockchain disassociates the time for creating a block from the rate at which blocks are added to the network. Theoretically speaking, the rate of adding blocks to the blockchain in a multi-stage blockchain can be arbitrarily lowered by choosing a suitably high value of the number of stages. This removes the first factor contributing to the delay in transaction processing. The second factor, i.e., network delay, however, remains. In fact, the network delay provides a lower bound on the rate at which blocks are added to the blockchain. The size of a block limits the number of transactions that can be accommodated in a block. One way to improve the transaction processing rate is to increase the block size so that more transactions fit within a single block. This approach is complementary to improving the rate at which blocks are added to the blockchain.

In a multi-stage blockchain, the proof-of-work of an individual stage can be released immediately after they are obtained. Other miners can work on such a proof-of-work to obtain proofs of works of subsequent stages ultimately leading to the mining of a complete block. Once a complete block is mined, the miners who contributed the proofs of works of the individual stages obtain an appropriate portion of the block reward. So, by releasing the proof-of-work of an individual stage, a miner locks in a certain return in case this proof-of-work is completed to mine an entire block. Such a strategy incentivises miners to cooperate in the process of block mining

The proof-of-work of individual stages require the application of hash functions. We put forward the idea that these hash functions should be defined from a set of hardware incompatible hash functions. This will make it very difficult for a single entity to obtain substantial computational advantage for all the stages. This reduces the possibility of 51% and selfish mining attacks.

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 [12, 13, 7, 11, 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 obtaining 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 modification of a blockchain based scheme which is somewhat closer to our notion is fruitchain [10]. 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. A multi-stage blockchain, on the other hand, does not have the two-tiered notion of fruits and blocks. Blocks contain transactions and are mined. It is the mining process itself that goes over several stages and refers to prior blocks.

Computing proofs of works can be an energy intensive procedure. There have been several suggestions to replace proof-of-work by proof-of-stake [6, 5, 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 also expect proof-of-work based schemes to co-exist in the cryptocurrency space.

In a usual blockchain, all the miners simultaneously compete with each other in the mining of the next block. There have been proposals which explicitly partition the set of entities such that each group of entities work on disjoint problems. Such a strategy has been called *sharding* and leads to improvement of transaction processing [8, 1, 15]. A multi-stage blockchain also implicitly introduces a sharding strategy in the sense that in general the miners will be divided into k groups with each group working on the proof-of-work of one particular stage for some block.

2 Description

In this section, we provide the description of the multi-stage blockchain and how it operates.

Parameters:

k	: number of stages;
μ	: number of hardware incompatible hash functions;
T	: stage completion time;
ν	: number of confirmations for a transaction.

Digital signature scheme: A digital signature scheme $(\mathsf{setup}, \mathsf{sign}, \mathsf{verify})$ is required. An entity invokes setup to obtain a signing-verification key pair $(\mathsf{sk}, \mathsf{pk})$. Algorithm sign is used on 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).

Hash functions: Let $H_0, \ldots, H_{k-1}$ be hash functions. The size of the digest for all of these hash functions is n_1 bits. Let H be a hash function which produces digests of size n_2 bits. It is possible to choose H to be one of $H_0, \ldots, H_{k-1}$ and in this case $n_1 = n_2 = n$.

Hardware incompatible hash functions: 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. For example, SHA2 and SHA3 are hardware incompatible hash functions.

A set of hash functions is said to be hardware incompatible if the hash functions in the set are pairwise hardware incompatible. For example, the finalists in the NIST Hash Competition form a set of hardware incompatible hash functions.

It is not easy to find a large set of secure hardware incompatible hash functions. Suppose $\{G_0, \ldots, G_\mu\}$ is a set of hardware incompatible hash functions. The required hash functions $H_0, \ldots, H_{k-1}$ can be defined from this set by the following rule.

$$H_i \quad = \quad G_{i \bmod \mu}, \quad i = 0, \ldots, k-1. \tag{1}$$

Address: Addresses are computed by applying the hash function H to a public key pk, i.e., $\mathfrak{a}$ is an address if $\mathfrak{a} = H(\mathsf{pk})$.

Transaction: A transaction T is a pair $(\mathsf{IL}, \mathsf{OL}, \sigma)$ where

1. $\mathsf{IL} = ((\mathsf{pk}_1, \mathfrak{c}_1), \ldots (\mathsf{pk}_s, \mathfrak{c}_s))$, $s \geq 1$. 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 $H(\mathsf{pk}_i)$.

2. $\mathsf{OL} = ((\mathfrak{a}_1, \mathfrak{d}_1), \ldots (\mathfrak{a}_t, \mathfrak{d}_t))$, $t \geq 1$. 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 $\mathfrak{a}_j$.

3. $\sum_{i=1}^{s} \mathfrak{c}_i \geq \sum_{j=1}^{t} \mathfrak{d}_j$. The difference $(\sum_{i=1}^{s} \mathfrak{c}_i) - \left(\sum_{j=1}^{t} \mathfrak{d}_j\right)$ is the transaction fee.

4. σ consists of the signature(s) on $(\mathsf{IL}, \mathsf{OL})$ constructed using the signing key(s) corresponding to the public key(s) $\mathsf{pk}_1, \ldots, \mathsf{pk}_s$.

We require each transaction to have at least one input address and one output address. Bitcoin has coinbase transactions which do not have an input address and has only one output address. These are used to assign block rewards. Our definition of transaction does not cover coinbase transactions. Instead, we will explicitly mention the addresses used for assigning block rewards as part of a block.

Root hash tree of a list of transactions: Let $\mathcal{L}$ be a list of transactions. These are to be hashed together using some kind of tree structure. Bitcoin uses the Merkle hash tree to stitch together the hashes of the transactions in the list. This uses a hash function to construct the tree on the list of transactions in $\mathcal{L}$ and the final output is a single digest. Ethereum uses a more complicated version of the Merkle hash tree.

For our purposes, the exact structure of the hash tree used to process the transactions in $\mathcal{L}$ is not important. We only specify that the hash function H_0 is used to build a hash tree from $\mathcal{L}$ and the final output of the hash tree will be called the *root hash* and denoted as $\mathsf{RH}(\mathcal{L})$.

Block: A general block B consists of 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. We note, on the other hand, that it is also possible to specify more complicated way of distributing the block reward $\mathsf{BR}(B)$ to the different stage addresses.

Blockchain: The blockchain is of the following form:

$$B_0 \leftarrow B_1 \leftarrow \cdots \leftarrow B_{k-1} \leftarrow B_k \leftarrow B_{k+1} \leftarrow \cdots$$

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.

General blocks: For $i \geq 0$,

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

The proofs of works 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 $g_{i+k,k-1}$.

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

$$\left.\begin{array}{rcl} g_{i+k,0} &<& t_{i+k,0} \\ g_{i+k,1} &<& t_{i+k,1} \\ \cdots & \cdot & \cdots \\ g_{i+k,k-1} &<& t_{i+k,k-1} \end{array}\right\} \quad (3)$$

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 proofs of works 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 $g_{i+k,j} < t_{i+k,j}$. The process of computing the proof of work for the j-th stage, i.e., obtaining $\eta_{i+k,j}$ can start immediately after the completion of the following two tasks.

1. Block B_{i+j} has been created.

2. The proof of work for stage $j-1$ has been completed, i.e., $g_{i+k,j-1}$ has been created.

Incremental block information: A general block has a k-stage proof-of-work. After completing each of the initial $k-1$ stages, a miner may release the proof-of-work for that stage. After completing the k-th stage, a miner releases the entire block. Incremental block information has the following format.

For $i \geq 0$,

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

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

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

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

Given $\mathsf{IBI}_{i+k,j}$ and $\mathsf{bdigest}_{i+j+1}$ from block B_{i+j+1}, it is possible to obtain the following.

1. $\mathsf{IBI}_{i+k,j+1}$ for $j = 0, \ldots, k-3$.

2. Block B_{i+j+1} for $j = k-2$.

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 conditions in (3) hold.

Note that $\mathsf{IBI}_{i+k,j}$ are not added to the blockchain. Only after the proof-of-work of the entire block B_{i+k} is complete that the block is added to the blockchain. So, the blockchain consists of only completely mined blocks and the blockchain does not keep track of incremental block information of different stages.

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.

$$\left.\begin{array}{rcl} \mathsf{bdigest}_0 & = & H_0\left(0, t_0, \mathfrak{a}_0, \mathfrak{c}_0, \tau_0, \eta_0\right); \\ \mathsf{bdigest}_i & = & H_i\left(\mathsf{bdigest}_{i-1}, i, t_i, \mathfrak{a}_i, \mathfrak{c}_i, \tau_i, \eta_i\right), \quad 1 \leq i \leq k-1. \end{array}\right\} \tag{4}$$

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

$$\mathsf{bdigest}_i \quad < \quad t_i. \tag{5}$$

Note that we do not define k-stage proofs of works for the start-up blocks. This can be done, but, does not seem to be necessary. 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.

For $i = 0, \ldots, k-1$, block B_i assigns $\mathfrak{c}_i$ coins to address $\mathfrak{a}_i$. The start-up blocks can be used for initial mining of some coins so that transactions become possible. The coins to be mined in the start-up blocks can be offered as initial coin offerings to prospective entities.

Stage completion time SCT: 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 determines the amount of time required to complete the proof of work of the stage. The goal is to ensure that all the stages require about the same time for the proof of work of the stage to complete. 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.

The targets for the various stages are to be updated at regular intervals so that SCT of T seconds is maintained. The actual rule for updation of target for each stage can be a simple one as used in Bitcoin or, it can be more sophisticated as for example used in Ethereum. We do not specify the actual target updation rule. As long as the updation rule is able to ensure that each stage requires about T seconds it can be fitted into our framework.

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. A simple method such as the one used in Bitcoin may be used, or, alternatively some more sophisticated method may be used. 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.

Block size: There will be an upper bound on the size of a block. We do not specify this bound. Any feasible bound may be used. The block-size parameter is independent of whether the proof-of-work is single-stage or multi-stage. The block-size limits the number of transactions that can be accommodated in a block. Suppose the block-size is fixed so that a maximum of 100 transactions will fit into a single block. This will hold irrespective of whether the proof-of-work is single-stage or multi-stage.

Network: The system is envisaged to operate in a decentralised, permissionless and trustless mode. Any entity would be able to download (or, implement) the software and become a node in the peer-to-peer to network. Such a node will maintain its own local copy of the blockchain. The nodes would communicate among themselves using some gossip protocol. Such a protocol should require nodes to do the following.

1. Relay transactions.

2. Relay incremental block informations.

3. Relay blocks.

Before relaying any information, each node 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 would include verification of timestamps and the rules for updating the targets of the various stages.

Choosing between blocks: Given competing blocks which can be added to the local copy of the blockchain, a node will choose the more difficult one. If the blocks have the same difficulty, then the node will retain the one which it received the earliest. The goal is to ensure that the consensus blockchain represents the maximum amount of work among all possible competing blockchains. This rule for choosing between blocks is essentially the same as that used in Bitcoin.

Forking: A fork is a split in the blockchain leading to two blockchains. These two blockchains share the same history up to a certain block and beyond that there is bifurcation which creates the two separate blockchains. A fork may arise due to the necessity of a rule change, or more, generally a disagreement in the community of users of the blockchain. Note that forking is a block-level phenomenon. Designing the proof-of-work to be multi-stage is a “within block” feature. So, having a multi-stage proof-of-work does not make forking either easier or more difficult to handle.

3 Block Mining

We assume that the k start-up blocks $B_0, \ldots, B_{k-1}$ have been mined. After this, general blocks can be mined and added to the blockchain. General blocks can accommodate transactions. So, processing of transactions can proceed after the start-up blocks have been mined.

The first list of transactions will be included in the block B_k. Completion of the proof of work for block B_k will have k stages. Each stage will require about T seconds to be completed. After the proof of work of the first stage of B_k is complete, the corresponding miner can release the incremental block information $\mathsf{IBI}_{k,0}$. This miner as well as other miners can build upon $\mathsf{IBI}_{k,0}$ to try and complete the proof of work of the second stage. Simultaneously, other miners may group together a second list of transactions and start mining the proof of work for the first stage of block B_{k+1}.

Each stage requires about T seconds to complete. So, the entire 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 entire 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.

More generally, for $i \geq 0$, the following two invariants will hold.

1. The time to complete the mining for the entire proof of work for block B_{i+k} will be about kT seconds.

2. Block B_{i+k+1} will be mined about T seconds after block B_{i+k} is mined.

Consequently, blocks will be added to the blockchain at the rate of one block per T seconds. On the other hand, the mining of each block will require kT seconds. This 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.

Time for confirmation of a transaction: Suppose a transaction is considered to be confirmed, if the block containing the transaction has been mined and at least ν subsequent blocks have been added to the blockchain. The time to mine the block containing the transaction is kT seconds while the time for the addition of ν subsequent blocks is νT. So, the total time for confirming a transaction is $kT + \nu T$.

Let us take some concrete figures to understand the speed-up.

Example 1: In Bitcoin, the time to mine a block is designed to be about 600 seconds. Suppose that a transaction is considered to be confirmed if five further blocks have been added to the blockchain. This means that Bitcoin requires about one hour for a transaction to be confirmed. Suppose that in our setting also, we wish to ensure that a block requires 600 seconds to be mined and is confirmed if five further blocks have been obtained. The time to mine a block in our setting is kT which we set equal to 600. Further, $\nu = 5$. Suppose, we choose $k = 10$. Then $T = 60$ seconds. Thus, the time to mine a block and obtain 5 further confirmations is $kT + \nu T = 600 + 300 = 900$, i.e., a transaction will be considered to be confirmed in about 15 minutes. This is a fourfold improvement over the confirmation time for Bitcoin. Blocks will be added to the blockchain at the rate of one block per minute which is a tenfold improvement over Bitcoin.

Example 2: In Litecoin, the time to mine a block is designed to be about 150 seconds. Suppose that a transaction is considered to be confirmed if 9 further blocks have been added to the blockchain. So, a transaction will be confirmed after about $150 \times 10 = 1500$ seconds (25 minutes). Again, we wish to ensure that the time to mine a block remains 150 seconds and is considered confirmed if 9 subsequent blocks are added to the blockchain, i.e., $kT = 150$ and $\nu = 9$. Suppose we choose $k = 5$ so that $T = 30$. Then the time for confirmation of a transaction is $kT + \nu T = 150 + 270 = 420$ seconds (7 minutes). This is about a fourfold improvement in confirmation time. Blocks will be added to the blockchain at the rate of one block per half-minute which is a fivefold improvement over Litecoin.

The above examples show how to speed up the confirmation time for a transaction without reducing the time to mine an individual block. The time to mine an individual block is a security issue. 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 [3] 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.

Remarks:

1. For any blockchain which is being actually used in the real world, it is reasonable to assume that in the steady state the rate at which new transactions are produced is more than the rate at which transactions are added to the blockchain. So, at any point of time, sufficiently

many new transactions will be available so that a new block is filled to its maximum size. This feature holds irrespective of whether the proof-of-work is single-stage or multi-stage.

2. As explained above, the time to mine a single block remains the same for both single-stage and multi-stage proof-of-work blockchain. Also, in both cases, the size of a block will be the same. Assuming for simplicity that all transactions have the same size, in both cases, a block will contain the same number of transactions.

Incremental mining: The multi-stage framework provides the option for releasing proofs of works of individual stages. We consider the incentive to a miner for doing so. Suppose a miner 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 proofs of works for the subsequent stages and eventually complete the block. As part of $\mathsf{IBI}_{i+k,0}$, miner 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 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 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 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 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 M control over these coins.

Alternatively, miner M could withold $\mathsf{IBI}_{i+k,j}$ and try to complete the mining of the entire block B_{i+k} by itself. By doing this, it runs the risk that the other miners can together mine block B_{i+k} earlier. Then, the entire work done by M in obtaining proofs of works of various stages will be wasted and result in no return. It would be better for the miner M to release stage-wise proof of works.

The above incentive mechanism encourages 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 proofs of works of the individual stages, the miners cooperate with each other to complete the mining of an entire block. In contrast, single stage blockchain only allows for competitive behaviour among the miners.

Simultaneous block mining: By the nature of the multi-stage blockchain, for $i \geq k$, the mining of stage $j+1$ of block B_i can be done simultaneously with the mining of stage j of block B_{i+1}, $j = 0, \ldots, k-2$. In the steady state, the miners will be divided into k groups 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};

This is another way in which the miners cooperate to extend the whole blockchain. In contrast, for a single stage blockchain, all miners would be competing with each other to mine block B_i.

The above mentioned partition of miners to work on disjoint problems is a form of *sharding*. Several proposals have been described which have sharding as an explicit goal. For a multi-stage proof-of-work blockchain, sharding arises implicitly.

Hardware incompatible hash functions: We have specified that the hash functions $H_0, \ldots, H_{k-1}$ are to be instantiated from a set of μ hardware incompatible hash functions as given by (1). It would require a huge investment from an individual miner to obtain very fast special purpose hardware for *all* the hash functions $G_0, \ldots, G_\mu$. 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. Other miners will invest in obtaining special purpose hardware for the other stages providing computational leverages for these 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 proofs of works of individual stages thus promoting the cooperative behaviour in the mining of a block.

4 Security

Suppose the j-th target for a block B is t_j. Then successful completion of the proof-of-work of stage j of the block requires about $2^{n_1}/t_j$ invocations of the hash function H_j. So, the total amount of work required to mine block B is

$$\frac{2^{n_1}}{t_0} C_{H_0} + \frac{2^{n_1}}{t_1} C_{H_1} + \cdots + \frac{2^{n_1}}{t_{k-1}} C_{H_{k-1}} \tag{6}$$

where C_{H_j} is the cost of applying the hash function H_j once.

Let ρ_j be the hash rate of the network for the hash function H_j, i.e., the network is able to compute H_j at the rate of ρ_j invocations per second. Then the time to mine block B is

$$\mathfrak{t} = \frac{2^{n_1}}{\rho_0 t_0} + \frac{2^{n_1}}{\rho_1 t_1} + \cdots + \frac{2^{n_1}}{\rho_{k-1} t_{k-1}}. \tag{7}$$

Let us consider the immutability of the blockchain which is important for ensuring that there is no double spending. Suppose there is an adversary $\mathcal{A}$ having hash rate ρ_j^* for the hash function H_j. Then the time required by $\mathcal{A}$ to mine block B is

$$\mathfrak{t}^* = \frac{2^{n_1}}{\rho_0^* t_0} + \frac{2^{n_1}}{\rho_1^* t_1} + \cdots + \frac{2^{n_1}}{\rho_{k-1}^* t_{k-1}}. \tag{8}$$

The adversary takes shorter time, i.e., $\mathfrak{t}^* < \mathfrak{t}$ if and only if

$$\frac{1}{t_0}\left(\frac{1}{\rho_0} - \frac{1}{\rho_0^*}\right) + \frac{1}{t_1}\left(\frac{1}{\rho_1} - \frac{1}{\rho_1^*}\right) + \cdots + \frac{1}{t_{k-1}}\left(\frac{1}{\rho_{k-1}} - \frac{1}{\rho_{k-1}^*}\right) > 0. \tag{9}$$

The total hash rate for hash function H_j is $\rho_j + \rho_j^*$ of which the fraction $\mathfrak{f}_j = \rho_j/(\rho_j + \rho_j^*)$ is controlled by the honest miners and the fraction $\mathfrak{f}_j^* = \rho_j^*/(\rho_j + \rho_j^*)$ is controlled by the adversary.

The j-th component of (9) is positive if and only if $\mathfrak{f}_j^* > 1/2$. For a usual (single stage) blockchain based on hash function H_j, the condition $\mathfrak{f}_j^* > 1/2$ is called the 51% attack. This signifies that if an adversary is able to control more than half the hash rate, it has a good chance of lauching an attack on the immutability of the blockchain which can lead to double spending.

If $k > 1$, then the $\mathfrak{f}_j^* > 1/2$ for one particular j does not necessarily imply a 51% attack. It is still possible that the other terms of (9) cancel out the advantage that the adversary has in the j-th component. If $\mathfrak{f}_j^* > 1/2$ for all $j = 0, \ldots, k-1$, then the adversary can surely launch an attack on the immutability of the blockchain. However, since the hash functions $H_0, \ldots, H_{k-1}$ are based on a set of hardware incompatible hash functions, it would require a huge investment for an adversary to gain control of more than half the hash rate for all the hash functions.

To summarise, the immutability of a multi-stage blockchain is at least as hard as that of a single-stage blockchain with the added benefit that an adversarial control over more than half the hash rate of any particular stage does not necessarily imply an attack on the immutability of the multi-stage blockchain. So, using a multi-stage blockchain does not decrease security and may actually improve it in some situations.

Selfish mining attack: In a usual (i.e., single-stage) blockchain, 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 results in a wastage of the computational power by the honest miners and a proportionate gain in computational power by the adversary. This strategy has been called selfish mining and analysed in details [4].

In the context of multi-stage blockchain, if an adversary is able to privately mine a whole block ahead of the public blockchain, then the attack applies in essentially the same way as that for a single-stage blockchain. However, in a multi-stage blockchain, proofs of works of individual stages are released as soon as they are completed. So, the honest miners cooperate to obtain the entire proof of work for any particular block. Further, the various stages use hardware incompatible hash functions. As a result of these two factors, it is extremely unlikely that an adversary is able to privately complete the entire proof of work for a single block ahead of all the honest miners. So, block level selfish mining seems to be difficult to mount on a multi-stage blockchain.

An adversary may attempt selfish mining on a particular stage of block mining. Suppose, it obtains a proof of work of one particular stage and does not release the corresponding incremental block information. Instead, it tries to complete the rest of the proof of work. This, however, does not amount to selfish mining. The incremental block information are not added to the blockchain. Only a completely mined block is added to the blockchain. So, no discarding of blocks from the blockchain of the honest miners occur. Also, as mentioned above, it is unlikely that any single entity will be able to complete the proofs of works of all the stages for mining a block.

By making it difficult for any single entity to mine an entire block by itself, multi-stage blockchain provides resilience to selfish mining attacks.

5 Conclusion

We have introduced a variant of decentralised, trustless, permissionless blockchain where the proof of work for mining a block is divided into multiple stages. This reduces the time for confirmation of a transaction and improves the rate at which transactions are processed without affecting the time to mine a block. It also provides several improved security benefits. Further, the process of mining a block becomes a combination of cooperative and competitive efforts between miners as opposed to only a competition for a single-stage blockchain. The improved features of a multi-stage blockchain make the idea an attractive option for future designs and implementations.

Acknowledgement

The author would like to thank Sanjay Bhattacherjee for many fruitful discussions on blockchains in general and cryptocurrency in particular. Thanks also to Alfred Menezes for providing helpful 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] 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.

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

[5] 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.

[6] 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.

[7] 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.

[8] 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.

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

[10] 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.

[11] Yonatan Sompolinsky, Yoad Lewenberg, and Aviv Zohar. Spectre: A fast and scalable cryptocurrency protocol. Cryptology ePrint Archive, Report 2016/1159, 2016. `http://eprint.iacr.org/2016/1159`.

[12] Yonatan Sompolinsky and Aviv Zohar. Accelerating bitcoin's transaction processing. fast money grows on trees, not chains. Cryptology ePrint Archive, Report 2013/881, 2013. `http://eprint.iacr.org/2013/881`.

[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.