A Survey on Blockchain Consensus Algorithms

Behzad Abdolmaleki

Supervisor: Dr. Vitaly Skachek
University of Tartu

1 Introduction

A blockchain is a linked data blocks, with each block containing a number of transactions. It provides a decentralised, immutable data store that can be used across a network of users, create assets and act as a shared black book that records all transactions. Each transaction can be easily queried, affording greater transparency and trust to all parties involved [Nak08]. One critical part of any blockchain, required if it is to function properly, is some sort of consensus algorithm that both secures the blockchain and ensures it works efficiently.

A consensus algorithm serves several purposes, two of which are safeguarding from manipulation and ensuring validity of transactions. Currently the most utilized consensus mechanisms are known as Proof of Work (PoW), which is the Bitcoin consensus protocol, and Proof of Stake (PoS), which is used by Ethereum, Peercoin [GKL15, BMZ18].

A primary consideration regarding the operation of blockchain protocols based on PoW—such as bitcoin [Nak08]—is the energy required for their execution. At the time of this writing, generating a single block on the bitcoin blockchain requires a number of hashing operations, which results in striking energy demands [OM14].

A natural alternative mechanism relies on the notion of "proof of stake" [KKR+16]. Rather than miners investing computational resources in order to participate in the leader election process, they instead run a process that randomly selects one of them proportionally to the stake that each possesses according to the current blockchain ledger.

Another alternative to PoW is the concept of proof of space [ABFG14, DFKP15], which has been specifically investigated in the context of blockchain protocols. In a proof of space setting, a "prover" wishes to demonstrate the utilization of space (storage / memory); as in the case of a PoW, this utilizes a physical resource but can be less energy demanding over time [PPA+15].

There are alternative consensus mechanisms being explored and one of these is called Proof of Burn (PoB) [Sli14]. In the proof of burn mechanism users are required to "burn" or make permanently unavailable some mined PoW cryptocurrency [ZXDW16].

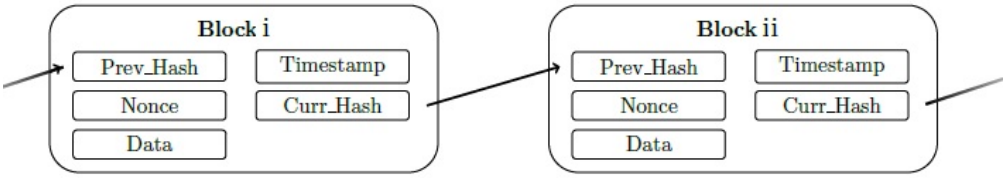

Figure 1: Two subsequent blocks in blockchain with their attributes [Nak08].

Paper overview . We lay out the basic definition of blockchain in Sec. 2. We explain proof of work consensus algorithm in Sec. 3. In short, in Sec. 4, we describe proof of stake and delegated proof of stake algorithm. In Sec. 5, we describe proof of space and its properties. In Sec. 6, we briefly explain how proof of burn works.

2 Blockchain

Blockchain is a decentralized database containing linked data blocks that are resistant to modifications. Blockchain was first introduced for constructing Bitcoin cryptocurrency in [Nak09]. Blockchain protocol contains a number of miners that collect users' data, called transactions, and store that data in discrete chunks, called blocks [Nak08, BMZ18]. Blocks are linked together by hashing to form a blockchain (see Fig. 1).

Every block in a blockchain consists of the same components as follows,

- A block number
- The hash of the previous block (via this means the 'chain' is being formed).
- Nonce, a random number, see below for more information.
- Data: the transactions.
- Timestamp with the time the block is created / found.
- The hash of the current block.

In other words, every block has a block number which, combined with the previous hash and the current hash, determine the order in which the transactions took place. The accompanying timestamp determine the time at which the transactions are recorded to have taken place. The Data contains the transactions, which could consist of nearly everything, not solely valuta. Last, but not least, there is the Nonce. This is used for the proof of work, and will be described in more detail in section 3.

3 Proof of Work

Bitcoin employs a puzzle friendly implementation of a one way hashing algorithm (SHA256) where the initial characters of the to be generated hash is predefined. The key component in Bitcoin protocol is the proof of work (PoW) problem solving. A miner can issue a new block only if it has solved a computationally difficult PoW challenge. In case of Bitcoin, miners need to find a hash of the previous block that has a certain number of zeros in the beginning. The miner that finds it first can issue the next block and her work is rewarded in cryptocurrency.

The nature of one way hashing algorithms is to bear no resemblance to its input. A one way hash function is a deterministic algorithm implying that a given input will always yield its corresponding output. An ideal one way hash function has no collisions. Given that a specific input always returns a specific output, in order to obtain a different output, the input must change.

In bitcoin, the designed puzzle is such that a person must recursively hash until they obtain a hash whose initial digits/characters have already been specified. The difficulty of obtaining a hash that matches the specific pattern increases exponentially as the number of specific digits increases. In the case of Bitcoin, we can observe on a blockexplorer how the hash of each block from genesis block (block 0) lead with a couple of zeros. This pattern of obtaining a series of zeroes to lead a hash gets harder as a longer series is demanded. The series of zeros leading a hash required is often used synonymously with the difficulty level.

This nonce once obtained can be hashed along with the rest of the block's contents to obtain the hash to verify. This verification process wouldn't take much time, but in order to come up with the nonce, one must hash the contents of the block and try various multiple iterations to obtain the nonce. This implies that verification is extremely easy whereas coming up with the nonce in the first place is quite a laborious process. Upon finding the nonce, the block containing the nonce is hence broadcasted to other bitcoin nodes on the network along with the hash to verify [OM14, BMZ18]. The whole process of finding the nonce is described as proof of work.

More precisely, consider Bitcoin as an example of a cryptocurrency system secured with a proof of work algorithm. Each block in Bitcoin consists of two parts:

- block header of key parameters, including block creation time, reference to the previous block and the Merkle tree root of the block of transactions

- block list of transactions.

To reference a specific block, its header is hashed twice with the SHA-256 function [5]; the resulting integer value belongs to the interval $[0, 2^{256} - 1]$. To account for different possible implementations, we will use a generic hashing function hash $H(.)$ with a variable number of arguments and range $[0, K]$. For example, arguments of the function can be treated as binary strings and merged together to form a single argument that can be passed to the SHA-256 hashing function. The block reference is used in the proof of work protocol; in

order for a block to be considered valid, its reference must not exceed a certain threshold:

$$H(B) \leq \frac{K}{D}, \tag{1}$$

where $D \in [1, K]$ is the target difficulty. There is no known way to find B satisfying Eq.1 other than iterating through all possible variables in the block header repeatedly. The higher the value of D, the more iterations are needed to find a valid block; the expected number of operations is exactly D. The time period $T(h)$ for a miner with hardware capable of performing h operations per second to find a valid block is distributed exponentially with the rate $\frac{h}{D}$,

$$Pr[T(h)] \leq tg = 1 - e^{\frac{-ht}{D}}.$$

Consider n Bitcoin miners with hash rates $h_1, h_2, \ldots, h_n$. The period of time to find a block T is equal to the minimum value of random variables $T(h_i)$ assuming that the miner publishes a found block and it reaches other miners immediately. According to the properties of the exponential distribution, T is also distributed exponentially:

$$Pr[T := min(T_1, T_2, \ldots, T_n) \leq t] = 1 - e^{(\frac{t}{D} \sum_{j=1}^{m} h_j)},$$

$$Pr[T = T_i] = \frac{h_i}{\sum_{j=1}^{n} h_j}.$$

The last equation shows that the mining is fair: a miner with a share of mining power p has the same probability p to solve a block before other miners.

Under the assumption that the majority of the PoW mining power follows the Bitcoin protocol, the users can become increasingly confident that the payment transactions that they receive will not be reversed [OM14, KN12, BMZ18]

4 Proof of Stake

Proof of stake is an algorithm to achieve distributed consensus within blockchain networks. It also is one of the main candidates to solve the energy demand problem in the current blockchain protocols such as Bitcoin and Ethereum [KKR+16].

Proof of Stake is intended to replace Proof of work since it is believed that recursive hashing is largely unproductive and unnecessary. Unlike bitcoin where anyone can mine irrespectively, Proof of Stake is structured to prioritize certain miners over other miners. This prioritization can be based on their ownership of ether in the network, however that would cause undesired centralization since the richest miner would then have an unfair permanent advantage in the mining process that happens within the network.

In proof of stake algorithms, equation(1) is modified to depend on the user's ownership of the particular PoS protocol cryptocurrency and not on block properties. Consider a user

with address A and state (balance) $st(A)$. A commonly used proof of stake algorithm uses a condition as,

$$H(H(B_{previous}), A, t) \leq st(A) \cdot \frac{K}{D} \tag{2}$$

where $B_{previous}$ denotes the block the user is building on, and t is the current timestamp.

Unlike equation (1), the only variable that the user can change is the timestamp t in the left part of equation (2). The address balance is locked by the protocol; e.g., the protocol may calculate the balance based on funds that did not move for a day. Alternatively, a PoS cryptocurrency may use unspent transaction outputs as Bitcoin does; in this case, the balance is naturally locked. A proof of stake protocol puts restrictions on possible values of t. For example, if t must not differ from the time on network nodes by more than an hour, then a user can attempt no more than 7200 values of t. Thus, there are no expensive computations involved in proof of stake.

Together with an address A and a timestamp t satisfying equation (2), a user must provide a proof of ownership of the address. To achieve this, the user can sign the newly minted block with his signature; in order to produce a valid signature, one must have a private key corresponding to the address A. The time to find a block for address A is exponentially distributed with rate $st(A) = D$. Consequently, the equation (2) implementation of proof of stake is fair: the probability to generate a valid block is equal to the ratio of user's balance of funds to the total amount of currency in circulation. The time to find a block for the entire network is distributed exponentially with rate $\sum_j st(j) = D$.

Thus, if the monetary supply of the currency $\sum_j st(j)$ is fixed or grows at a predictable rate, the difficulty D should be known in advance:

$$D = \frac{1}{T_b} \sum_j st(j),$$

with T_b denoting the expected time between blocks. In practice, D needs to be adjusted based on recent blocks because not all currency owners participate in block minting [KKR+16, KN12, BMZ18].

4.1 Delegated Proof of Stake

Delegated proof of stake (DPoS) is a generic term describing an evolution of the basic PoS consensus protocols [Lar14]. In this protocol, blocks are minted by a predetermined set of users of the system (delegates), who are rewarded for their duty and are punished for malicious behavior (such as participation in double-spending attacks). In DPoS algorithms, delegates participate in two separate processes:

- building a block of transactions.
- verifying the validity of the generated block by digitally signing it.

While a block is created by a single user, to be considered valid, it typically needs to be signed by more than one delegate. The list of users eligible for signing blocks is changed periodically using certain rules; for example, in Slasher, delegates for each block are selected based on stake and blockchain history. The set of delegates for each block is typically small; a notable exception is Tendermint, for which each block can be signed by any of the users of the system. In some DPoS versions, a delegate needs to show commitment by depositing his funds into a time-locked security account (which is confiscated in case of malicious behavior); this version of DPoS is often referred to as deposit-based proof of stake.

Delegated proof of stake does not use the conditions of the equation (2). Indeed the stake is factored into DPoS with one of the following methods:

- Delegates may be elected based on their stake in the system.
- Delegates may receive votes from all users of the system with voting power depending on a voter's stake.
- Delegates' votes on valid blocks may have power proportional to the size of their security deposit.

Overall, delegated proof of stake is less standardized than basic PoS [Lar14, BMZ18].

5 Proof of Space

Another alternative to PoW is the concept of proof of space [ABFG14, DFKP15], which has been specificall investigated in the context of blockchain protocols. In a proof of space setting, a prover P wishes to demonstrate the utilization of space (storage / memory); as in the case of a PoW, this utilizes a physical resource but can be less energy demanding over time. PoS are partially motivated by the observation that users often have significant amounts of free disk anyway, and in this case using a PoS is essentially for free. This is in contrast to a PoW, as computing will always require energy consumption even if one contributes only CPU time of processors that would otherwise be idle.

A PoS is a protocol between a prover P and a verifier V which has two distinct phases. After an initialisation phase P is supposed to store some data F of size N, whereas V only holds some small piece of information. At any later time point V can initialise a proof execution phase, and at the end V outputs reject or accept. We require that V is highly efficient in both phases, whereas P is highly efficient in the execution phase providing he stored and has random access to the data F.

As an illustrative application for a PoS, suppose that the verifier V is an organization that offers a free email service. In order to prevent that someone registers a huge number of fake-addresses for spamming, V might require users to dedicate some nontrivial amount of disk space, say 100GB, for every address registered. Occasionally, V will run a PoS to verify that the user really dedicates this space [DFKP15].

5.1 Proof of Space Definition

We denote with $(out_V, out_P) \leftarrow (V(in_V), P(in_P))(in)$ the execution of an interactive protocol between two parties P and V on shared input in, local inputs in_P and in_V, and with local outputs out_V and out_P, respectively. A proof of space is given by a pair of interactive random access machines, a prover P and a verifier V. These parties run the PoS protocol in two phases: a *PoS initialization* and a *PoS execution* as defined below. The protocols are executed with respect to some statement id, given as common input (e.g., an email address). The identifier id is only required to make sure that P cannot reuse the same space to execute PoS for different statements.

- **Initialization** is an interactive protocol with shared inputs an identifier id, storage bound $N \in \mathbb{N}$ and potentially some other parameters, which we denote with $prm = (id, N)$. The execution of the initialization is denoted by $(\phi, S) \leftarrow (V, P)(prm)$, where ϕ is short and S is of size N. V can output the special symbol $\phi = \perp$, which means that it aborts (this can only be the case if V interacts with a cheating prover).

- **Execution** is an interactive protocol during which P and V have access to the values stored during the initialization phase. The prover P has no output, the verifier V either accepts or rejects.

In an honest execution the initialization is done once at the setup of the system, e.g., when the user registers with the email service, while the execution can be repeated very efficiently many times without requiring a large amount of computation.

To formally define a proof of space, we introduce the notion of a (N_0, N_1, T) dishonest prover $\hat{P}$. The dishonest prover storage after the initiation phase is bounded by at most N_0, while during the execution phase its storage is bounded to N_1 and its running time is at most T (here $N_0 \leq N_1$ as the storage during execution contains at least the storage after initialization). Notice that $\hat{P}$'s storage and running time is unbounded during the the initialization phase (but, as just mentioned, only N_0storage is available in-between the initialization and execution phase).

A protocol (P, V) as defined above is a (N_0, N_1, T)-proof of space, if it satisfies the properties of completeness, soundness and efficiency defined below.

- **Completeness.** It requires that for any honest prover P:

$$Pr[(\phi; S) \leftarrow (V, P)(prm); out \leftarrow (V(\phi), P(S))(prm) : out = accept] = 1.$$

  Note that the probability above is exactly 1, and hence the completeness is perfect.

- **Soundness.** For any (N_0, N_1, T)-adversarial prover $\hat{P}$ the probability that V accepts is negligible in some statistical security parameter λ. More precisely, we have,

$$Pr[(\phi; S) \leftarrow (V, \hat{P})(prm); out \leftarrow (V(\phi), \hat{P}(S))(prm) : out = accept] \leq 2^{-\lambda}.$$

  The probability above is taken over the random choice of the public parameters prm and the coins of $\hat{P}$ and V.

Efficiency. We require the verifier V to be efficient, by which (here and below) we mean at most polylogarithmic in N and polynomial in some security parameter λ. Prover P must be efficientduring execution, but can run in time poly(N) during initialization.

6 Proof of Burn

Proof of burn is a solution to the drawbacks of proof of work mining. Proof of burn (PoB) utilizes the idea of burning coins to reduce the need for powerful computational resources when mining [Sli14]. Proof-of-burn solves proof of work's dependency of powerful hardware. To acquire more mining power, instead of waiting days/months on end, all that is needed is for some coins to be burned. This idea has been implemented into the coin generation process of the *Slimcoin* cryptocurrency. Like *PPCoin*'s design, this design attempts to demonstrate the viability of cryptocurrencies that can still be minted once a sufficient amount of coins have been generated, while minimally requiring powerful hardware.

6.1 Burning Coins

The role of burning coins is to be proof when mining by proof of burn. The concept simply involves sending coins to a burn address. A burn address is a special address that is specific and predetermined. No one has ownership of the burn address, meaning once coins are sent to it, they are forever gone and are considered "burnt". It is impossible for burnt coins to be "un-burnt". Burning coins can be parallel to buying hardware for proof of work mining. It costs money (burning *Slimcoins*), but after they are burned, they can be used to generate (mine) blocks.

6.2 Burn Hashes

In principal, when mining proof of work with physical hardware, out of all the hashes calculated over a time span x, only the best hash is the most important. Every other hash can be disregarded. Proof of burn parallels proof of work in that exact sense. The burn hash represents the best hash over a specific time span. In proof of burn, that time span is the amount of time from the creation of one proof of work block to the creation of the next proof of work block. Because of this concept, it can be said that "burnt coins are mining rigs", the main difference being the hardware demands. In proof of work, the best hash is found by calculating thousands/millions of hashes and picking the smallest one out of all of them. In proof of burn, only one calculation is necessary, for the burn hash is the best hash.

$$BurnH = m * intH,$$

where $BurnH$ is called burn hash, m is multiplier and $intH$ is internal Hash. Every burn transaction calculates it own burn hash. The algorithm for calculating a burn hash differs from proof of work's algorithm. Burn hashes are calculated by multiplying a multiplier to an internal hash. The multiplier is different for each burn transaction. It follows an exponential

growth curve where the independent variable is determined by the number of proof of work blocks found since the creation of its corresponding burn transaction. The multiplier is also inversely proportional to the amount of coins burned. The multiplier is what causes burnt coins to "decay". Decayed burnt coins are a representation of how much the multiplier has increased. For example, if the multiplier of a burn transaction of value 60 coins triples over the course of 1000 days, the same multiplier would be achieved if on the 1000th day, 20 coins were burned. Thus, it can be said that 40 coins have decayed from the first burn transaction.

The calculation of the internal hash is done in such a way that, if no new proof of work blocks have been found, it will always produce a same exact hash. That means that burn hashes have to be only calculated when a new proof of work block has been found. That is what makes proof of burn a non intensive task, for a few hashes every minute or so can be done with a low-power computer. When being multiplied by the multiplier, the internal hash is treated as a number (256 bits). Based on how many coins have been burned and how many of them have decayed, a burn hash is calculated [Sli14, ZXDW16].

6.3 Proof of Burn Block Generation

The generation of a proof of burn block requires the burn hash to be less than or equal to the burn hash target of the block. The burn hash target of the block is determined by using the total network's effective (mature and undecayed) burnt coins. That target is stored in the block in the form of "burn bits" and is fixed and equal for every node on the network. Like proof of stake block generation, the hashing operation occurs of over a select search space (one hash per burn transaction per proof of work block). The hardware required to maintain that hash operation is minimal, thus the energy consumption and hardware maintenance can be considered to be near none. The block generation of proof of burn scales linearly, meaning if 200 coins were burned and subsequently generated 10 proof of burn blocks over the course of 1 day, it can be expected that if 40 coins were to be burned, 2 proof of burn blocks would be generated over the course of that same day, assuming the effective burnt coins in the network remained constant.

A proof of burn block must include the coordinates of a burn transaction of which its respective burn hash meets the hash target requirements. These coordinates point to exactly where the burn transaction is found in the blockchain. They contain information regarding which block and where within that block the burn transaction is found. The coordinates and are used to look up the burn transaction during the block's verification process. The signature of the block must match the signature of the in-transaction of the burn transaction pointed to by the block. That prevents the same proof of burn block to be copied, have its signature changed, and used again an attacker. In other words, it prevents an attacker from using someone else's burn transaction to mine proof of burn blocks and then have the coins generated be issued to himself instead of to the proper owner of the burn transaction.

In addition, every block in the blockchain, regardless of which type it is, has a check for the network's effective burnt coins. That value is passed down from block to block and adjusts based on the amount of newly burnt coins (its value increases) and the amount of coins that have decayed in the network (its value decreases). This value must be precise

since it is used to calculate the hash target for the proof of burn blocks, thus it must be checked as part of every block's verification process.

6.4 Duplicate Proof of Burn Protocol

A duplicate proof of burn block detection protocol was created to prevent an attacker from using a valid proof of burn block to generate a multitude of copies which could be used to flood the network as a denial-of-service (DoS) attack. Every proof of burn block contains a value (256 bit integer) that is set to the block's burn hash. Each node collects that value of all proof of burn blocks that have been seen. If that value containing the burn hash differs from the calculated burn hash, that would cause its respective proof of burn block to be rejected (deemed invalid). If a block with the same burn hash is received, it is ignored, unless it is required by a successor block that has been orphaned (shunted away until its depending blocks are received) [Sli14, ZXDW16].

References

[ABFG14] Giuseppe Ateniese, Ilario Bonacina, Antonio Faonio, and Nicola Galesi. Proofs of space: When space is of the essence. In *International Conference on Security and Cryptography for Networks*, pages 538–557. Springer, 2014.

[BMZ18] LM Bach, B Mihaljevic, and M Zagar. Comparative analysis of blockchain consensus algorithms. In *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pages 1545–1550. IEEE, 2018.

[DFKP15] Stefan Dziembowski, Sebastian Faust, Vladimir Kolmogorov, and Krzysztof Pietrzak. Proofs of space. In *Annual Cryptology Conference*, pages 585–605. Springer, 2015.

[GKL15] Juan Garay, Aggelos Kiayias, and Nikos Leonardos. The bitcoin backbone protocol: Analysis and applications. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 281–310. Springer, 2015.

[KKR+16] Aggelos Kiayias, Ioannis Konstantinou, Alexander Russell, Bernardo David, and Roman Oliynykov. A provably secure proof-of-stake blockchain protocol. *IACR Cryptology ePrint Archive*, 2016:889, 2016.

[KN12] Sunny King and Scott Nadal. Ppcoin: Peer-to-peer crypto-currency with proof-of-stake. *self-published paper, August*, 19, 2012.

[Lar14] Daniel Larimer. Delegated proof-of-stake (dpos). *Bitshare whitepaper*, 2014.

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

[OM14] Karl J O'Dwyer and David Malone. Bitcoin mining and its energy footprint. 2014.

[PPA+15] Sunoo Park, Krzysztof Pietrzak, Joël Alwen, Georg Fuchsbauer, and Peter Gazi. Spacecoin: A cryptocurrency based on proofs of space. Technical report, IACR Cryptology ePrint Archive 2015, 2015.

[Sli14] P4Titan. Slimcoin. Spacecoin: A peer-to-peer crypto-currency with proof-of-burn. Technical report, IACR Cryptology ePrint Archive 2014, 2014.

[ZXDW16] Zibin Zheng, Shaoan Xie, Hong-Ning Dai, and Huaimin Wang. Blockchain challenges and opportunities: A survey. *International Journal of Web and Grid Services*, 1:1–25, 2016.