BLOCKASTICS

Stochastic models for blockchain analysis

Pierre-O Goffard

April 6, 2022

Chapter 1

Introduction

A blockchain is a distributed ledger made of a sequence of blocks maintained by achieving consensus among a number of nodes in a Peer-to-Peer network. The blockchain technology has attracted a lot of interest after the advent of the bitcoin cryptocurrency in 2008, see Nakamoto [2008]. Since then, the blockchain concept has been used to develop decentralized systems to store and maintain the integrity of time-stamped transaction data across peer-to-peer networks. Besides the creation of a digital currency, blockchain applications include the sharing of IT resources, the registration of authentication certificate or the implementation of smart contracts.

A blockchain is

- Decentralized as it is maintained by a network. Nodes can be light or full nodes. Light nodes are blockchain users that broadcast transactions, full nodes are in charge of verifying and recording the transactions, see Figure 1.1.

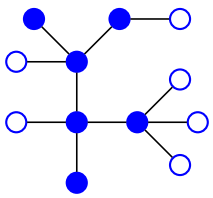


Figure 1.1: A network made of full nodes (blue) and light nodes (white)

  - A local copy is stored by each full node which grants security
  - The governance is not handled by a central authority

- Public or private. In public blockchain anyone can access the data, in private blockchain reading access is restricted.
- permissionned or permissionless. In permissionless blockchain, anyone can join the network as a full node.

- Immutable. Altering the information written in the blockchain is made difficult if not impossible.
- Incentive compatible. The process of reaching consensus is costly to the full nodes who must be compensated for their hard work.

The consensus protocols, at the core of the blockchain technologies, are the focus of these lecture notes. The goal is to evaluate consensus protocol according to three dimensions

1. Efficiency: The amount of data being processed per time unit
2. Decentralization: The fairness of the distribution of the decision power among the nodes
3. Security: The likelihood of a successful attack on the blockchain

Because consensus protocols involve random components, stochastic modelling is required to assess a blockchain system within the Efficiency/Decentralization/Security trilemma in Figure 1.2. As it is hard to improve one dimension without negatively impacting the other two, trade-offs

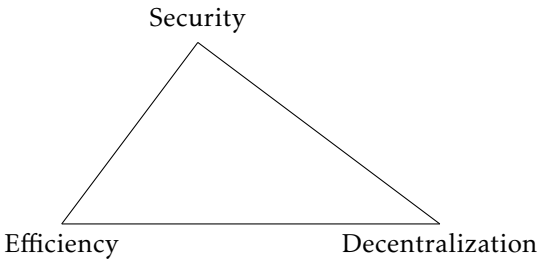


Figure 1.2: The blockchain trilemma

must be made. We will see how to use classical models of applied probability, including urn, epidemic, graph, queue and risk models, to provide numerically tractable indicators to quantify the efficiency, decentralization and security of blockchain systems. These indicators will then allow us to carry out sensitivity analysis with respect to the model parameters to optimize and improve blockchain implementations.

The main application of blockchain systems today is undoubtedly cryptocurrencies, the most well known of which being the bitcoin introduced by Nakamoto [2008]. Public and permissionless blockchain, like the bitcoin one, must be associated to a cryptocurrency. Indeed, to add a block to the bitcoin blockchains the full nodes compete to solve a cryptrographic puzzle using brute force search algorithm. The first node (referred to as a miner) who finds a solution, appends the next block and collects a reward expressed in cryptocurrency. Assuming this reward is worth something, it offsets the operational cost which is essentially the electricity consumed to run the computers 24/7. A cryptocurrency must be equipped with following features

1. No central authority (Decentralized network)

2. Ledger to record all the transactions and coin ownership (the blockchain)
3. A coin generation process (block finding reward)

   ↪ It creates an incentive compatible system to the full nodes
4. Ownership can be proved cryptographically, a wallet is secured with a public/private key system
5. Transactions can be issued by an entity proving ownership of the cryptographic unit through the private key
6. The system cannot process more than one transaction associated to the same cryptographic unit. It must be robust to double spending attack in which a fraudster is issuing two conflicting transactions to recover the funds she already spent

This charcaterization is given by Lansky [2018]. Cryptocurrencies draw their fundamental value from the fact that they

- provide transaction anonymity
- provide a reliable currency in certain regions of the world
- permit money transfer worldwide at low fare
- do not require a thrusted third party

An important implication of this architecture is disintermediation, it creates an environment where multiple parties can interact directly and transparently. Blockchain is therefore immediately relevant to banks and financial institutions which incur huge middlemen costs in settlements and other back office operations. Decentralized finance (DeFi) offers a new financial architecture that is non-custodial, permissionless, openly auditable, pseudo-anonymous and with potential new capital efficiencies. It extends the promise of the original bitcoin whitepaper Nakamoto [2008] of non-custodial transaction to more complex financial operations, see the SoK of Werner et al. [2021].

Blockchain is a research topic of interest to many communities. Computing science distributed ledger technologies (synonymous with blockchains) rely on distributed algorithms and enable cooperation within a peer-to-peer network. Linking blocks and checking the authenticity of data uses cryptographic functions which is another field of computer science. The establishment of an incentive system within a network of individuals adopting a strategic behavior naturally leads to problems of game theory similar to those solved by economists. The discussion on the nature of new financial assets such as crypto-currencies, utility tokens and non-fungible tokens, is also at the center of the concerns of researchers in finance and monetary economics.

We focus here on the use of mathematics to optimize blockchain systems which makes our problems very close to those encountered in operations research. These notes are organized as follows. Chapter 2 presents the various consensus algorithms. Chapter 3 focuses on the security aspects. In Chapter 3, we take a look at decentralization in Chapter 4. We close on efficiency with Chapter 5.

Chapter 2

Consensus protocol

Transactions flow through the network of full nodes. After reviewing them, the full nodes must agree on the transaction that will be recorded in the next block. to do this, an algorithm must be designed so that consensus is reached. A consensus protocol must be based on one of the scarce resources available to the network peers which include

- bandwidth
- computational power
- storage

The first solution that comes to mind for reaching consensus is a majority vote based on a message exchange system. This solution has been proposed by Lamport et al. [1982] within the famous "Byzantine general problem". A voting system inside a large network involves a colossal number of messages exchanged leading to the consumption of all the bandwidth, the failure of some nodes by denial of service and delays in the synchronization of the network. Practical solution like the celebrated Practical Byzantine Fault Tolerance (PBFT) presented in Castro and Liskov [1999] have been implemented in some blockchain systems. Despite these advances, a change in methods was needed to accommodate a network that could grow indefinitely.

Nakamoto [2008] solved this scaling problem by proposing a system based on the election of a leader. The Proof-of-Work (PoW) protocol appoints a leader based on its computing resources. Each node competes to solve a puzzle with a brute force search algorithm. The first node who is able to propose a solution append the next block. The search for a solution, referred to as mining, is associated with an operational cost borne by the nodes which is compensated by a reward expressed in the native blockchain cryptocurrency. The surge in cryptocurrency prices has led to a rush in block mining, leading to a major spike in the electricity consumption and electronic waste generation of blockchain networks. The blockchain network consumes as much electricity as countries the size of Thailand at the time of the writing. The need for a more environmentally friendly consensus protocol therefore becomes pivotal. Protocol such

as *Proof-of-Capacity* and *Proof-of-Spacetime* use storage. Using storage is seen as a fairer and greener alternative by blockchain enthusiasts due to the general purpose nature of storage and the lower energy cost required by storage. The fact that most storage resources are owned by companies offering cloud storage solution poses a threat to the decentralized nature of the distributed ledger. The Proof-of-Interaction (PoI) protocol, proposed by Abegg et al. [2021], takes as leader the first node that is able to contact and obtain a response from a random sequence of nodes. This is a bandwidth-based alternative that is more scalable than majority voting. Along with bandwidth, computing power, and storage, a new resource has emerged with the advent of cryptocurrencies as a medium of exchange. The Proof-of-Stake protocol, described by Saleh [2020], selects a node with a probability proportional to the number of cryptocoins it holds.

The consensus protocol are applied so that a blocks are appended sequentially and not at the same time. Usually the consensus process is divided into time slots, also called rounds. The block generation time must be higher than the propagation delay in the network. If two blocks are created at the same time then a fork will occur. Two branches of the blockchain co-exists. A fork situation then resolves by applying the *Longest Chain Rule* (LCR).

Definition 1. *The Longest Chain Rule states that if there exist several branches of the blockchain tehn the longest should be trusted.*

This definition implies that a threshold must be chosen in order to decide when shorter branches of the blockchain should be discarded. For instance, a branch can be considered legitimate if it is $k \in \mathbb{N}$ blocks ahead of its pursuers. For the consensus protocol to be viable, nodes must be incentivized to follow the LCR.

This chapter is organized as follows. Section 2.1 gives a brief description of the voting based ways to get consensus by reviewing the "generals" problem. Section 2.2 goes through the leader based consensus protocols, including PoW in Section 2.2.1, PoSp in Section 2.2.2, PoI in Section 2.2.3, and PoS in Section 2.2.4. For an exhaustive list of the existing protocols the reader is referred to https://tokens-economy.gitbook.io/consensus/.

2.1 Voting system

The problem of reaching consensus in a peer-to-peer network via a majority vote has been abstractedly compared to generals who must agree on a common battle plan. We start from the simple two general case before moving on the the situation of interest with several ones.

2.1.1 Two generals problem

Two generals wish to attack a city but they must agree on a timing to attack a city. They communicate via a messenger who must cross enemy territory at the risk of being intercepted.

The first general G_1 sends a message to the second one G_2 saying

```
"I will attack tomorrow at dawn"
```

For the attack to succeed, both generals must attack at the same time. Because their communication medium is unreliable, then G_1 must await confirmation from G_2 in order to attack. If G_1 does not receive confirmation then she will not attack. G_2 is aware of that and respond

```
"I will follow your lead"
```

G_2 does not know whether the message went through and must wait for confirmation. This creates an infinite loop of messages and response, as on Figure 2.1. The two general problem is

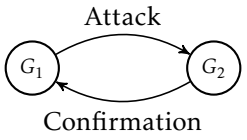


Figure 2.1: Message and confirmation loop

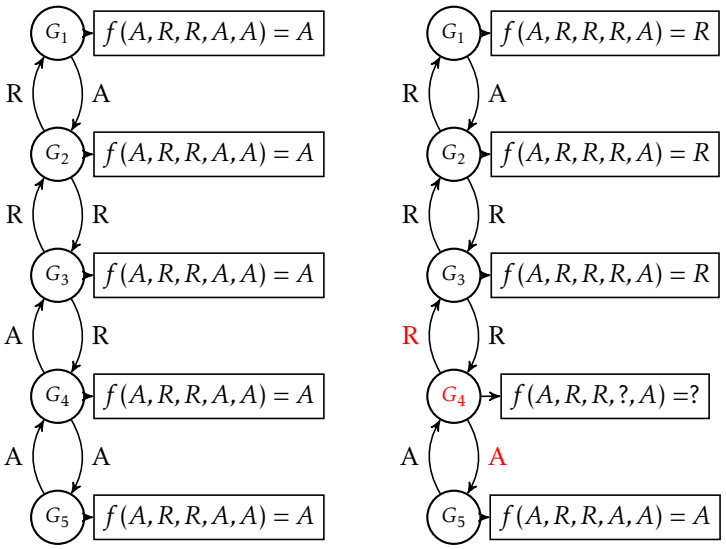
deemed unsolvable from a theoretical point of view and corresponds to a situation where two nodes communicate through an unreliable link. A practical solution for generals is to send many messengers hoping that at least one of them will succeed. This is only a thought experiment leading to the several general problem.

2.1.2 Byzantine General problem

The blockchain network contains more than two nodes, these nodes must agree on the transactions to confirm. In a permissionless blockchain the nodes do not trust each other. The problem of the previous section generalizes to more than two generals, assuming that some generals are traitors which corresponds to faulty nodes in the network. This problem is referred to as The "Byzantine general problem" and was coined by Lamport et al. [1982]. Assume that $n > 2$ generals must agree on a common battle plan for instance "Attack" (A) or "Retreat" (R) and that they can only communicate by two party messages. Denote by $m(i, j)$ the message sent by general i to general j. Each general j receives $n-1$ messages and applies a function f to determine the course of action, for instance

$$f(\{m(i,j);\ i = 1,\dots,n\}) = \begin{cases} A, & \text{if } \sum_{i=1}^{n} \mathbb{I}_{m(i,j)=A} > n/2, \\ R, & \text{else.} \end{cases}$$

If there are no traitors, each general is communicating the same value to all the peers and consensus is reached as in Figure 2.2a. If one general is traitor, then he might not communicate the same value to all the generals and no consensus can be reached. It is the case for G_4 in Figure 2.2b. To handle such a situation, roles are given to the general. One of them become the leader and the other are the lieutenants. We aim at finding an algorithm such that



(a) No traitor (b) One traitor

Figure 2.2: Majority vote with or without a traitor

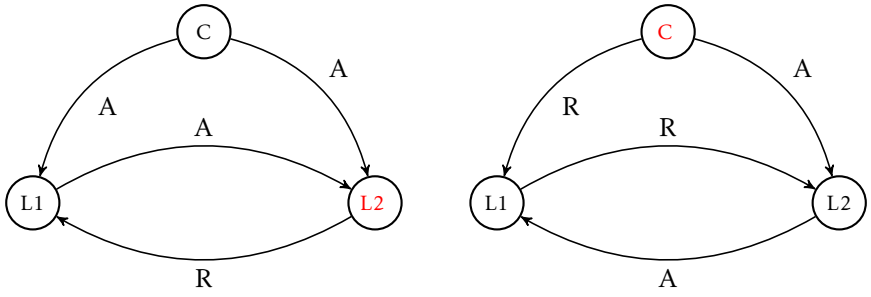
C1 All the loyal lieutenants obey the same order

C2 If the commanding general is loyal, then every loyal lieutenants obey the order he sends

A first result from Lamport et al. [1982] is the following

Theorem 1. *There are no solution to the Byzantine General problem for $n < 3m + 1$ generals where m is the number of traitors.*

Proof. Consider the situation where $n = 3$ and $m = 1$. The traitor is either the commander or one of the lieutenants as shown in Unfortunately for Lieutenant 2, there is no way for her to tell



(a) Commander is loyal (b) Commander is a traitor

Figure 2.3: Majority vote with or without a traitor

apart the situation pictured in Figure 2.3a and Figure 2.3b and therefore no way to ensure both C1 and C2. We prove the result for $n > 3$ by contradiction. Assume that there is a way to verify both C1 and C2 with $3 < n < 3m + 1$. We then construct a solution with generals by having one general simulate the commander plus at most $m - 1$ generals, and the other two simulating at

most m generals. One of the generals gather all the traitors and is therefore a traitor. The other two are loyal generals as they only simulate loyals general. We have built a solution with three generals that we know is impossible. □

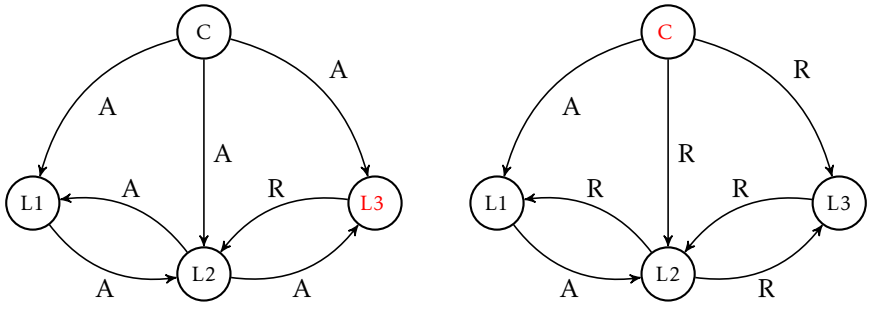
Now we need an algorithm that allows $n > 3m+1$ generals to deal with m traitors. The 'Oral Message' algorithm denoted by OM(m) and summarized in Algorithm 1 can handle m traitors if the number of generals verifies $n > 3m+1$. Before looking into the theoretical justification of

Algorithm 1 The Oral message algorithm OM(m)

```
1:  if m = 0 then;
2:      for i = 1 → n − 1 do
3:          Commander sends v_i = v to lieutenant i
4:          Lieutenant i set their value to v
5:      end for
6:  end if
7:  if m > 0 then;
8:      for i = 1 → n − 1 do
9:          Commander sends v_i to lieutenant i
10:         Lieutenant i uses OM(m-1) to communicate v_i to the n − 2 lieutenants
11:     end for
12:     for i = 1 → n − 1 do
13:         Lieutenant i set their value to f(v_1, ..., v_{n−1})
14:     end for
15: end if
```

OM(m), let us illustrate the algorithm with an example.

Example 1. *Consider the situation where $n = 4$ and $m = 1$ shown in Figure 2.4. If the commander is*



(a) Commander is loyal

(b) Commander is a traitor

Figure 2.4: Illustration of the OM(m) algorithm in the case where $n = 4$ and $m = 1$.

loyal then one of the lieutenant is a traitor, see Figure 2.4a. The commander gives the order to attack to all the lieutenant 3 *tells the other that she heard retreat from the commander. The loyal lieutenants*

then apply the map f to agree on their value

$$f(A, A, R) = A,$$

which coresponds to the order the commander sent, hence IC1 and IC2 are satisfied. If the commander is a traitor as in Figure 2.4b, then he sends conflicting order to the lieutenant but after communicating the value they received to each other finally agree on the following value

$$f(A, R, R) = R,$$

hence IC1 is satisfied and IC2 can be ignored since the commander is a traitor.

Theorem 2. *Algorithm OM(m) satisfies conditions IC1 and IC2 if $n > 3m + 1$.*

Proof. The proof follows from simple induction.

First assume that the commander is loyal. For $m = 0$, the commanders simply sends the value v to all the lieutenants and IC2 holds. Assume that OM($m-1$) works when the commader is loyal. The commander sends v to all the lieutenants. The lieutenants then applies $OM(m-1)$. Because $n-1 > 2k+m-1$, then it follows from the induction hypothesis that each loyal lieutenants get the value v for each of the loyal lieutenants j. The loyal lieutenants $n-1-m > 2k-1 > m$ outnumber the traitorous lieutenants and therefore set their value to

$$f(v_1, \ldots, v_{n-1}) = v,$$

and both IC1 and IC follow.

Let us assume that the commander is a traitor, we only have to worry about IC1 in that case. There are at most m traitors and the commander is one of them. We therefore have $m-1$ traitors among the lieutenants. Since the total number of lieutenants exceeds three times the number of traitors $n-1 > 3m > 3(m-1)$ then by applying OM($m-1$) all the loyal lieutenants receive the same vector of values $v_1, \ldots, v_{n-1}$, agree on the same value

$$f(v_1, \ldots, v_{n-1}) = v,$$

which leads to the verification of IC1. □

The main problem associated to this Oral message algorithm is the number of messages is n^{m+1} which is prohibitive for large values of n and m. A celebrated algorithm, called Practical Byzantine Fault Tolerance (PBFT) has been developped later on by Castro and Liskov [1999] but still not fast enough to enable the infinite growth of the network associated to public and permissionless blockchains.

2.2 Leader system

The scalability issue can be solved by opting for a leader based mechanism instead of a majority vote mechanism. The protocols presented in this section use computational power, storage and bandwidth to elect a leader each time a new block must be appended to the blockchain.

2.2.1 Proof-of-Work

The bitcoin blockchain relies on a consensus protocol based on computational power called Proof-of-Work (PoW), presented in Nakamoto [2008]. A block consists of

- a header
- a list of "transactions" that represents the information recorded through the blockchain.

The header usually includes

- the date and time of creation of the block,
- the block height which is the index inside the blockchain,
- the hash of the block
- the hash of the previous block.

The hash of a block is obtained by concatenating the header and the transactions in a large character string thus forming a "message" denoted by m, to which a hash function h is applied.

Definition 2. *A hash function is a function that can map data of arbitrary size to fixed-sized values,*

$$h : \{0,1\}^* \mapsto \{0,1\}^d$$

The hash functions used in blockchain applications must be cryptographic, i.e.

- quick to compute
- one way
- deterministic

Remark 2.2.1. *It must be nearly infeasible to generate a message with a given hash value or to find two messages with the same hash value. A small change in the message should change dramatically the hash value so that the new hash value appears to be uncorrelated to the previous hash,*

$$\text{if } m_1 \approx m_2 \text{ then } h(m_1) \neq h(m_2).$$

We will not expand on how to build such a cryptographic hash function, we refer the interested reader to the work of Al-Kuwari et al. [2011].

In the bitcoin blockchain as well as in many other applications, the standard is the SHA-256 function which converts any message into a hash value of 256 bits. The latter is usually translated into a hexadecimal digest, for instance the hash value of the title of the present manuscript reads as

98b1146926548f6b57c4347457713ff2f035beda9c93f12fbc9b202e9c512e80.

Mining a block means finding a block hash value lower than some target which can only be achieved by brute force search thanks to the properties of cryptographic hash functions. In practice, the search for an appropriate hash value, referred to as a solution, is done by appending a nonce to the block message before applying the hash function. A nonce is a 32 bits number, drawn at random by miners until a nonce resulting in a proper block hash value is found. For illustration, consider the block in Figure 2.5.

```
Block Hash: 1fc23a429aa5aaf04d17e9057e03371f59ac8823b1441798940837fa2e318aaa
Block Height: 0
Time:2022-02-25 12:42:04.560217
Nonce:0
Block data: [{'sender': 'Coinbase', 'recipient': 'Satoshi', 'amount': 100, 'fee': 0}, {'sen
der': 'Satoshi', 'recipient': 'Pierre-O', 'amount': 5, 'fee': 2}]
Previous block hash: 0
Mined: False
--------------
```

Figure 2.5: A block that has not been mined yet.

The hash value in decimal notation is $1.43e^{76}$ while the maximum value for a 256 bits number is $2^{256} - 1 \approx 1.16e^{77}$. We refer to the latter as the maximal target and denote it by $T_{\max}$. The Proof-of-Work protocol sets a target $T < T_{\max}$ and ask miners to find a nonce such that the hash value of the block is smaller than T. Practitioners would rather talk about the *difficulty* which is defined as $D = T_{\max}/T$. If the difficulty is one, any hash value is acceptable. Increasing the difficulty reduces the set of allowable hash values, making the problem harder to solve. A hash value is then called *acceptable* if its hexadecimal digest starts with a given number of zeros. If we set the difficulty to 2^4, then the hexadecimal digest of the hash of the block must start with at least 1 leading zero, making the hash value of the block in Figure 2.5 not acceptable. After completing the nonce search we get the block in Figure 2.6. Note that it took 5 attempts to

```
Block Hash: 0869032ad6b3e5b86a53f9dded5f7b09ab93b24cd5a79c1d8c81b0b3e748d226
Block Height: 0
Time:2022-02-25 13:41:48.039980
Nonce:2931734429
Block data: [{'sender': 'Coinbase', 'recipient': 'Satoshi', 'amount': 100, 'fee': 0}, {'sen
der': 'Satoshi', 'recipient': 'Pierre-O', 'amount': 5, 'fee': 2}]
Previous block hash: 0
Mined: True
--------------
```

Figure 2.6: A mined block with a hash value having on leading zero.

find this nonce. The number of needed trials is geometrically distributed with parameter $1/D$,

which means that with a difficulty of $D = 2^4$ it takes on average 16 trials. The protocol adjusts the difficulty automatically every 2,016 block discoveries so as to (globally) maintain one block discovery every 10 minutes on average. The time between two block discoveries depends on the number of hash values computed by the network at a given instant. At the time of writing, the network computes 182.58 Exahashes per second and the difficulty is 27,967,152,532,434.[1] For an exhaustive overview of the mining process in the bitcoin blockchain, we refer the reader to the book of Antonopoulos [2017, Chapter 10]. As each trial (of the system) for mining a block is independent of the others and leads to a success with very small probability, the overall number of successes is binomially distributed and will be very well approximated by a Poisson random variable. This justifies the Poisson process assumption made in the sequel to model the block arrival and the reward collecting processes. Empirical studies of the block inter-arrival times data tend to confirm this hypothesis, see the work of Bowden et al. Bowden et al. [2020]. The information recorded in a public blockchain may be retrieved by anyone and can be accessed through a blockchain explorer such as blockchain.com, the content of the block of height #724724 may be viewed through the following link block content.

The PoW protocol implies that the nodes are running computations 24/7 therefore consuming humungous quantity of electricity. Bitcoin mining originally started by running computations using the Central Processing Unit (CPU). It turns out that certain kinds of computation are more efficient on Graphics Processing Unit (GPU) than on CPUs. CPU is designed to complete a wide variety of task while computing hashes is very specific. GPU are tailored to run thousands of computation of the same type. Miners then turned to GPUs leading to a shortage of graphics card at the expense of PC gamers around the world! Eventually GPU got replaced by Application Specific Integrated Circuits (ASICs) that are designed to complete very specific task compared to graphic cards. ASICs consumes 10 times more power than graphic cards but compute 10,000 more hashes than a graphic card per time unit. Miners then decided to equip themselves with ASIC chips leading to harmful consequences

- Increase of the network electricity consumption
- Increase in the e-waste generation. ASICs are single purpose and it cannot be repurpose for any other task. When ASICs become obsolete with the arrival of a new generation of chips, they are thrown in the trash.
- The main manufacturer of ASICs is a company called BITMAIN which equips major mining pool such as Antpool and BTC.com. A threat on centralization exists since a company like BITMAIN could take control of the network by owning more than half of the overall hashpower.

A pro-ASIC argument is that it would be impossible for anyone (apart from BITMAIN) to suddenly acquire enough of these chips to have more than half of the world's hash power.

[1] Source: bitcoinblockhalf.com

2.2.2 Proof-of-SpaceTime and Proof-of-Capacity

Consensus protocol based on storage capacities are seen by many as a fairer and greener alternative to PoW. We describe below two such protocols *Proof-of-Capacity* and *Proof-of-Spacetime*.

Proof-of-Capacity

In the *Proof-of-Capacity*, miners compute hashes and cache the result on their hard disk space. Mining then only requires to search through the cache for an admissible solution.

Proof-of-Spacetime

In the *Proof-of-Spacetime*, the nodes store data and produce proofs to show that the data has been stored for a given time period. The probability of a node being chosen is proportional to the amount of data stored. This protocol has been designed for a specific application allowing nodes to provide storage to clients through the Filecoin project.

To some extent the *Proof-of-Capacity* protocol is similar to PoW while the *Proof-of-Spacetime* shares similarities with the *Proof-of-Stake* protocol which is discussed below.

Such protocols do not generate ewaste because disk space can always be used for some other purpose. Storing data is less energy consuming than computing hashes. The problem of hiring external storage capacities from provider remains.

2.2.3 Proof-of-Interaction

The *Proof-of-Interaction* protocol, introduced by Abegg et al. [2021], asks each validating node to get in touch with a sequence of nodes. The number of nodes and the nodes to be contacted are drawn randomly so that the time to complete the task is also varying from one node to another. The block reward is shared by the contacting and responding nodes to create an incentive compatible environment. If we assume that the time required to complete the task is exponentially distributed then the time to generate a new block is the minimum of exponential random variables which is again exponentially distributed. PoI is still in the developping phase and many interesting work must be done to assess the security and viability of such protocol. Some nodes may indeed collude to send replies faster or not to send replies to some node. It is necessary to evaluate the probability and the opportunity for the nodes to collude.

2.2.4 Proof-of-Stake

Besides bandwidth, computing power and storage, one ressource that appears with the advent of cryptocurrencies as medium-of-exchange and store-of-value asset are the cryptocoins. Each time a block must be appended to the blockchain, a coin is drawn at random. The owner of that

coin appends a new block and collect the reward.

Let the network be of size N. We denote by π_i^t the proportion of coins owned by node $i \in \{1,\dots,n\}$ at time $t \in \mathbb{N}$. Note that π_i^t is exactly the probability of node i being elected as leader at time t, we have

$$\begin{cases} \pi_i^t > 0, \\ \sum_{i=1}^N \pi_i^t = 1, \end{cases} \quad \text{for } t > 0.$$

Denote by S_t the total number of coins in circulation at time t and by R_t the size of the reward for appending a new block at time t. Let A_i^t be the event of node i appending a block at time t, the share of coins then evolves as

$$\pi_i^t = \frac{S \cdot \pi_i^t + \sum_{s=1}^{t-1} \mathbb{I}_{A_i^t} R_s}{S + \sum_{s=1}^{t-1} R_s},$$

where

$$\mathbb{I}_A = \begin{cases} 1 & \text{if A occurs,} \\ 0 & \text{otherwise.} \end{cases}$$

Two potential issues needs to be studied

- The Nothing-at-Stake (NaS) problem: If a a fork is ongoing then each branch will elect a leader who will append a block, collect the reward and perpetuate the disabreement.
- The rich get richer problem: When a node is chosen, it becomes richer which increase its likelihood to be chosen in future rounds.

The "rich get richer" problem will be extensively studied in Chapter 4. Regarding the NaS problem, the nodes when chosen by a branch decide whether they want to add a block, it is an option. The cryptocoin value comes from its use as a medium of exchange. A long lasting disagreement results in a useless cryptocoin with no value.

Let τ be the duration of the fork and let $\delta \in (0,1)$ be a discount rate, then the present value of a coin at τ is $1/(1+\delta)^\tau$. If $\mathbb{P}(\tau = \infty) > 0$ then the coin value is zero when taking the expectation.

The nodes are therefore incentivized to follow the Longest chain rule in order to resolve the fork situation as soon as possible. This is essentially the rationale in Saleh [2020] to show that

- The coin value reaches a maximum if all the nodes follow the longest chain rule
- There exists an equilibrium in which the nodes follow the LCR if

  $$\min \pi_i^0 \cdot S \geq \frac{R}{\delta(1-\delta)^2},$$

  which corresponds to a minimum stake condition.

- If $\sum_t R_t < \infty$ then there exist no equilibrium for which

$$\mathbb{P}(\tau = \infty) > 0.$$

A modest reward schedule precludes the posssibility of an ever lasting fork.

A practical implementation of the PoS protocol to create a cryptocurrency is PeerCoin, see the white paper by King and Nadal [2012]. The notion of coin age is introduced, the stake is actually defined by the number of coins times the number of time period during which the coins was hold. When a peer finds a block, a *coinstake* transaction is made that transfers the node its own coin to reset the coin age to zero.

Chapter 3

Security of blockchain systems

The security evaluation of blockchain systems consists in calculating the probability of a successful attack on the blockchain. We will focus, in Section 3.1, on the double spending attack which is concern for PoW powererd cryptocurrency like teh bitcoin one. Security is also at risk when the node have an incentive to deviate from the prescribed protocol. Section 3.2 discusses the opportunity for miner of PoW equipped blockchain to resort to blockwithholding strategy to optimize their revenue.

3.1 Double-spending in PoW

A double spending attack aims at generating a concurrent blockchain to replace the main one. Consider the following scenario

1. Marie sends to John BTC10
2. The transaction from Marie to John is recorded in the blockchain
3. John is advised for α confirmation, that is for $k-1$ block to be appended after the block where the Marie to John transaction is recorded
4. Once α confirmations have been sent, John ships the good
5. Meanwhile, Marie has started working on her own blockchain version where the Marie to John transaction is replaced by a Marie to Marie transaction
6. At the shipment date the main blockchain is ahead by z blocks
7. Marie's goal is then to work on her blockchain branch to catch up with the main branch. If she manages to to that then her branch will replace the public branch and she recovers her bitcoin. She can therefore spend these bitcoins again hence the name double spending.

The race between the two competing branches of the blockchain is summarized on Figure 3.1.

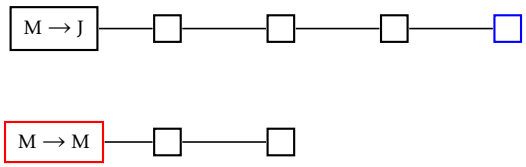


Figure 3.1: Double spending race illustrated

3.1.1 Random walk model

We define a discrete time stochastic process $(R_n)_{n\geq 0}$ equal to the difference in length between the public and the private branch of the blockchain. At each time step a block is found, it belongs to the main branch with probability p to the other branch with probability $q = 1-p$. The parameter p represents the proportion of hashpower owned by the honest miners, while q is that of the attacker. We have

$$R_0 = z, \text{ and } R_n = z + Y_1 + \ldots + Y_n.$$

The Y_i's are i.i.d. random variables such that

$$\mathbb{P}(Y = 1) = p \in (0,1), \text{ and } \mathbb{P}(Y = -1) = 1 - p = q,$$

$(R_n)_{n\geq 0}$ is therefore a random walk on $\mathbb{Z}$. We assume that $p > q$ so that the attacker does not hold more than half of the total hashpower. Define the double spending time as

$$\tau_0 = \inf\{n > 0 \, ; \, R_n = 0\}.$$

Our goal is to study the distribution of this stopping time with respect to the filtration

$$\mathcal{F}_n = \sigma(Y_1, \ldots, Y_n), \; n \geq 1.$$

An illustration of this first-hitting time problem is provided in Figure 3.2. Let us denote by

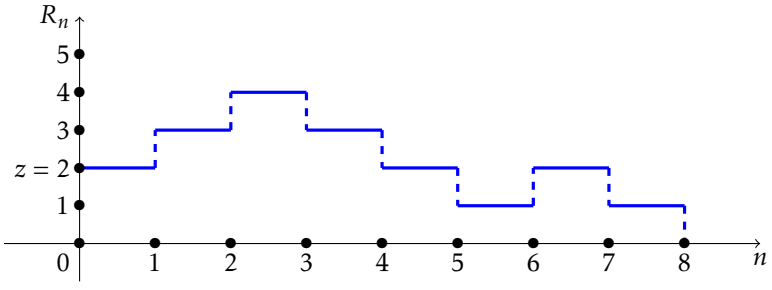


Figure 3.2: Illustration of the first-hitting time problem of a double spending attack.

$$\mathbb{P}_z(\cdot) = \mathbb{P}(\cdot | R_0 = z) \text{ and } \mathbb{E}_z(\cdot) = \mathbb{E}(\cdot | R_0 = z)$$

We are interested for now in the cvonditional distribution of τ_0 provided that $R_0 = z$.

Double spending probability

The double spending probability is defined as

$$\phi(z) = \mathbb{P}_z(\tau_0 < \infty),$$

and given in the following result

Theorem 3. *If $p > q$ then*

$$\phi(z) = \left(\frac{q}{p}\right)^z.$$

We give two proofs for this result, the first one uses simple first step analysis exploiting the Markov property of the random walk. The second one uses Martingale and the optional stopping theorem.

Proof 1:

Using a first step analysis, we have

$$\phi(z) = p\phi(z+1) + (1-p)\phi(z-1),\ z \geq 1. \tag{3.1}$$

We also have the boundary conditions

$$\phi(0) = 1 \text{ and } \lim_{z\to+\infty} \phi(z) = 0 \tag{3.2}$$

Equation (3.1) is a linear difference equation of order 2 associated to the following characteristic equation

$$px^2 - x + 1 - p = 0$$

which has two roots on the real line with

$$r_1 = 1, \text{ and } r_2 = \frac{1-p}{p}.$$

The solution of (3.1) is given by

$$\phi(z) = A + B\left(\frac{1-p}{p}\right)^z,$$

where A and B are constant. Using the boudary conditions (3.2), we deduce that

$$\phi(z) = \left(\frac{1-p}{p}\right)^z$$

as announced.

For the second proof we need the notion of martingale

Definition 3. *A stochastic process $(X_n)_{n\geq 0}$, is called a martingale with respect to a filtration $\mathcal{F}_n$, if*

(i) X_n *is* $\mathcal{F}_n$*-adapted*

(ii) $\mathbb{E}(X_n) < \infty$ *for* $n \geq\geq 0$

(iii) $\mathbb{E}(X_n|\mathcal{F}_{n-1}) = X_{n-1}$

and the optional stopping theorem.

Theorem 4. *Let T be a stopping time for the Martingale $(X_n)_{n\geq 0}$ then it holds that*

$$\mathbb{E}(X_T) = \mathbb{E}(X_0)$$

in each of the following situations

(i) T is bounded almost surely

(ii) There exists $c > 0$ such that $|X_{T\wedge n}| < c$ for every $n > 0$.

(iii) $\mathbb{E}(T) < \infty$, and, for some $K > 0$ we have that

$$|X_n(\omega) - X_{n-1}(\omega)| \leq K, \ \forall (n, \omega).$$

Proof 2:

Define the process

$$X_n = \exp\left[sR_n - n\kappa_Y(s)\right],$$

where $s > 0$, and

$$\kappa_Y(s) = \log\left[\mathbb{E}\left(e^{sY}\right)\right],$$

is the cumulant generating function of Y.

Lemma 1. *$(X_n)_{n\geq 0}$ is a $\mathcal{F}_n$-martingale.*

Proof. Denote by $M_Y(s) = \mathbb{E}(e^{sY})$ the moment generating function of Y, we have that

$$\begin{aligned}
\mathbb{E}(X_n|\mathcal{F}_n) &= \mathbb{E}\left\{\exp\left[sR_n - n\kappa_Y(s)\right]|\mathcal{F}_n\right\} \\
&= \exp\left[sR_{n-1} - n\kappa_Y(s)\right]\mathbb{E}\left[\exp\left(sY_n\right)|\mathcal{F}_n\right] \\
&= \exp\left[sR_{n-1}\right]M_Y(s)^{-n}M_Y(s) \\
&= X_{n-1}.
\end{aligned}$$

□

The equation $\kappa_Y(s) = 0$ is equivalent to

$$pe^s + qe^{-s} = 1$$

which admits $\gamma = \log(q/p)$ as only nonnegative solution. The process $(\mathrm{e}^{\gamma R_n})_{n\geq 0}$ is a $\mathcal{F}_n$-Martingale. Define $\tau_a = \inf\{n \geq 0 \ ; \ R_n = a\}$, for $a > z$. Consider the stopping time $\tau = \tau_0 \wedge \tau_a$, we have that for any $n > 0$,

$$\mathbb{P}(\tau = \infty) \leq \mathbb{P}(\tau > n) < \mathbb{P}(|R_n| \leq a) = 0;$$

We can therefore apply the optional stopping time theorem at τ to get

$$\begin{aligned}
\mathbb{E}(X_\tau) = \mathbb{E}(X_0) \quad &\Leftrightarrow \quad \mathbb{P}(\tau = \tau_0) + [1 - \mathbb{P}(\tau = \tau_0)]\mathrm{e}^{az} = \mathrm{e}^{\gamma z} \\
&\Leftrightarrow \quad \mathbb{P}(\tau = \tau_0)] = \frac{\mathrm{e}^{\gamma z} - \mathrm{e}^{az}}{1 - \mathrm{e}^{az}}.
\end{aligned}$$

We then let $a \to \infty$ in the above equation to conclude that

$$\mathbb{P}(\tau = \tau_0) = \left(\frac{q}{p}\right)^z.$$

Exercise 1. *What happens if*

- $p = q = 1/2$?
- $q > p$?

In practice the number of blocks z is actually random variable

$$Z = (\alpha - M)_+,$$

where M corresponds to the number of blocks the attacker managed to mine while the vendor waits for α confirmations. If we assume that a block mined by the honest miners is a success while a block mined by the attacker is a failure then M actually counts the number of failure before α successes. We have that $M \sim \text{Neg-Bin}(\alpha, p)$ where M has a probability mass function (p.m.f.) given by

$$\mathbb{P}(M = m) = \binom{\alpha + m - 1}{m} p^\alpha q^m.$$

Whenever $Z = 0$ then double spending occur right away as $\phi(0) = 1$. To derive the double spending probability, we condition upon the values of Z via the law of total probability

$$\mathbb{P}(\text{Double Spending}) = \mathbb{P}(M \geq \alpha) + \sum_{m=0}^{\alpha-1} \binom{\alpha + m - 1}{m} q^\alpha p^m.$$

Double spending time

In the block mining world time is money. Every hour spent in computing hashes is costly in terms of energy. It is then very interesting to know whether a double spending attack is meant to last long or not. Intuitively, we can think that if it must occur the it should at a earlier stage because as $p > 1/2$ our random walk $(R_n)_{n \geq 0}$ will eventually drift toward $+\infty$. The following result provides the probability distribution of τ_0 when $R_0 = z$.

Theorem 5. *If $z = 0$ then $\tau_0 = 0$ almost surely. If $z > 0$ then τ_0 admits a p.m.f. given by*

$$\mathbb{P}(\tau_0 = n) = \frac{z}{n} \binom{n - z}{(n - z)/2} p^{(n-z)/2} q^{(n+z)/2} \text{ if } n > z \text{ and } n - z \text{ is even},$$

and 0 otherwise.

Proof. We start by showing the following lemma, sometimes referred to as the Markov hitting time theorem.

Lemma 2.

$$\mathbb{P}(\tau_0 = n) = \frac{z}{n} \mathbb{P}(R_n = 0)$$

Proof. □

□

3.1.2 Counting process model

Double spending probability

Double spending time

3.2 Blockwitholding in PoW

3.2.1 The selfish mining strategy

3.2.2 Ruin and expected profit of a miner

3.2.3 Ruin and expected profit of a selfish miner

3.3 Nothing-at-stake in PoS

Chapter 4

Decentralization of blockchain system

4.1 Decentralization in PoS

Rich get richer? Polya's urn

4.1.1 Average stake own by each peer

4.1.2 Distribution of the stakes

4.2 Decentralization in PoW

4.2.1 Mining pools and reward systems

4.2.2 Mining pool risk analysis

Chapter 5

Efficiency of blockchain systems

5.1 A queueing model with bulk service

5.2 Latency and throughputs computation

Bibliography

Jean-Philippe Abegg, Quentin Bramas, and Thomas Noël. Blockchain using proof-of-interaction. In *Networked Systems*, pages 129–143. Springer International Publishing, 2021. doi: 10.1007/978-3-030-91014-3_9.

Saif Al-Kuwari, James H. Davenport, and Russell J. Bradford. Cryptographic hash functions: Recent design trends and security notions. Cryptology ePrint Archive, Report 2011/565, 2011. https://ia.cr/2011/565.

Andreas Antonopoulos. *Mastering Bitcoin*. O'Reilly UK Ltd., July 2017. ISBN 1491954388. URL https://www.ebook.de/de/product/26463992/andreas_antonopoulos_mastering_bitcoin.html.

R. Bowden, H. P. Keeler, A. E. Krzesinski, and P. G. Taylor. Modeling and analysis of block arrival times in the bitcoin blockchain. *Stochastic Models*, 36(4):602–637, jul 2020. doi: 10.1080/15326349.2020.1786404.

Miguel Castro and Barbara Liskov. Practical byzantine fault tolerance. In *Proceedings of the Third Symposium on Operating Systems Design and Implementation*, OSDI '99, page 173–186, USA, 1999. USENIX Association. ISBN 1880446391. URL https://dl.acm.org/doi/10.5555/296806.296824.

Sunny King and Scott Nadal. Ppcoin: Peer-to-peer crypto-currency with proof-of-stake. *whitepaper*, 2012.

Leslie Lamport, Robert Shostak, and Marshall Pease. The byzantine generals problem. *ACM Transactions on Programming Languages and Systems*, pages 382–401, July 1982. URL https://www.microsoft.com/en-us/research/publication/byzantine-generals-problem/.

Jan Lansky. Possible state approaches to cryptocurrencies. *Journal of Systems Integration*, 9(1): 19–31, jan 2018. doi: 10.20470/jsi.v9i1.335.

S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system. Available at https://bitcoin.org/bitcoin.pdf, 2008. URL https://bitcoin.org/bitcoin.pdf.

Fahad Saleh. Blockchain without waste: Proof-of-stake. *The Review of Financial Studies*, 34(3): 1156–1190, jul 2020. doi: 10.1093/rfs/hhaa075.

Sam M. Werner, Daniel Perez, Lewis Gudgeon, Ariah Klages-Mundt, Dominik Harz, and William J. Knottenbelt. Sok: Decentralized finance (defi), 2021.