TÉCNICO LISBOA

Etherspace: Practical Proof-of-Space for Blockchains

Diogo Peres Castilho

Thesis to obtain the Master of Science Degree in

Information Systems and Software Engineering

Supervisor(s):
Prof. Miguel Ângelo Marques de Matos
Prof. João Pedro Faria Mendonça Barreto

Examination Committee

Chairperson: Prof. Alexandre Paulo Lourenço Francisco
Supervisor: Prof. Miguel Ângelo Marques de Matos
Member of the Committee: Prof. Pedro Miguel dos Santos Alves Madeira Adão

November 2019

I dedicate this work to my father, José Castilho,

and my grandmother, Clarisse Peres,

may them rest in peace.

Acknowledgments

I would like to thank everyone that helped me during this work.

To my advisors Prof. João Pedro Faria Mendonça Barreto and Prof. Miguel Ângelo Marques de Matos for all the help, guidance, motivation, feedback, and support they have given me along the way.

To Paulo Silva for sharing his knowledge about the Ethereum implementation in go, which was essential for this work.

To my mother, Marisa Castilho, my brother and sisters, Hugo Castilho, Barbara Castilho, Susana Castilho, for the good times we shared, for encouraging me and discussing ideas.

To my girlfriend, Ana Novais, for her love, emotional support, patience, for helping me be a better version of myself and never letting me give up.

To my friends, Nuno Ribeiro for distracting me through tough times, Diogo Morais for always knowing what I needed most, Marie Castilho and Monique Castilho for all the company they gave me when I felt alone.

To all my colleagues at Instituto Superior Técnico with a special mention to Pedro Ceyrat, for all the restless nights and fun times we shared.

Last, but not least, I would like to thank Instituto Superior Técnico for the education I received, and the personal growth that came from it, which will be fundamental throughout my life.

This work is partially funded by Fundo Europeu de Desenvolvimento Regional (FEDER) via Programa Operacional Regional de Lisboa and by Fundos Nacionais via FCT - Fundação para a Ciência e a Tecnologia through project Lisboa-01-0145-FEDER-031456 (Angainor) and UID/CEC/50021/2019.

Resumo

A maior parte das *blockchains* são baseadas no modelo da Bitcoin, e portanto, a sua segurança depende de provas de trabalho (*proof-of-work*, ou PoW, em inglês). Para adicionar blocos à *blockchain* é necessário que os utilizadores provem que usaram uma certa quantidade de poder computacional. As provas de trabalho fazem com que os requisitos energéticos das *blockchains* sejam muito elevados consumindo, no caso da Bitcoin, energia equivalente a um país como a Irlanda.

Este trabalho propõe uma nova implementação de *blockchain* que substitui as provas de trabalho pelas provas de espaço com o objectivo de diminuir os requisitos energéticos das *blockchains*. Numa *blockchain* que use provas de espaço, os mineiros têm de provar que estão a dedicar quantidades não triviais de memória ao protocolo. Antes de participarem no protocolo começam por realizar computações cujos resultados serão guardados na memória. Quando adicionarem blocos à *blockchain* têm de demonstrar que estão de facto a guardar corretamente o resultado dessas computações. O uso de provas de espaço em *blockchains* é um tópico recente e a maioria do trabalho nesta área é teórico, não sendo trivial como podem ser contruídas soluções práticas. Este trabalho propõe uma nova implemtentação de *blockchain* que substitui as provas de trabalho pelas provas de espaço, realizada em cima do protocolo do Ethereum, uma das criptomoeadas mais populares.

Palavras-chave: Provas de trabalho, Provas de espaço, Blockchain, Ethereum, Criptomoeda

Abstract

Most blockchains follow Bitcoin's model, and, as a result, their security relies on proof-of-work. In order to add blocks to the chain, users must prove that they used a certain amount of computational power. Proof-of-work is very energy-intensive, with Bitcoin's energy consumption being on par with a country like Ireland.

This thesis aims at proposing a novel proof-of-space alternative to proof-of-work that reduces the energy requirements of blockchain protocols. In a blockchain that uses proof-of-space, miners must prove that they are dedicating non-trivial amounts of memory to the protocol. Before being able to start mining, miners must perform some computations whose results will be stored in the memory. Whenever they add a block to the blockchain, they must prove that they are correctly storing the result of those computations. The usage of proof-of-space in blockchains is a recent topic, and the majority of existing work in this area is only theoretical. This work proposes a concrete implementation of a blockchain that uses proof-of-space, built on top of the Ethereum protocol, one of the most popular blockchains.

Keywords: Proof-of-work, Proof-of-Space Blockchain, Ethereum

Contents

List of Tables

List of Figures

Chapter 1

Introduction

In recent years blockchains have become a popular topic. A blockchain is a data structure maintained by a distributed network of nodes that provides an immutable append-only log of records. The records are grouped into blocks, and each block has the cryptographic hash of the previous one. Thus any change to the order of blocks or the content of any block will invalidate the chain. Nodes must also run a consensus protocol to ensure that all nodes agree on the blocks that are added to the chain.

Blockchains are essential in scenarios where we want to run a decentralized application on an untrusted network with nodes that can have arbitrary behaviors. The most popular type of applications that use blockchains are cryptocurrencies. A cryptocurrency is a digital asset that provides an alternative to conventional currencies. Cryptocurrencies run in a decentralized network where there are no trusted third-parties (in opposition to conventional currencies that rely on central banks). Another appealing feature of blockchains is smart contracts. Smart contracts are pieces of code that run on top of the blockchain. They allow two or more parties to build trusted distributed applications since the underlying blockchain ensures that the code is executed and that all nodes agree on the result of executing that code. Ethereum [7] is a prominent blockchain platform that can be used to develop such smart contracts.

Most blockchains use a variant of the protocol known as Nakamoto Consensus [26], originally introduced in Bitcoin [65]. One can think of Nakamoto Consensus as the combination of 3 things: incentives, a chain selection rule, and proof-of-work. Proof-of-work is a cryptographic puzzle that nodes must solve in order to add blocks to the blockchain that binds the ability to contribute to the protocol with the expenditure of non-counterfeitable resources, namely computational power, that prevents Sybil attacks [43]. Nevertheless, this mechanism is not sufficient to select which blocks should be added to the chain. In a big network where all nodes are working to create new blocks, two nodes might create two blocks concurrently, which results in different nodes with different valid blockchains, a fork. In order to make nodes agree on the same chain, we need a chain selection rule. In Bitcoin, nodes choose the longest proof-of-work chain i.e., the chain that required more computational power to generate. The last but maybe the most important part is the incentive. Similarly to BitTorrent [36], incentives build robustness in Bitcoin by making nodes want to follow the protocol. When a node adds a block to the blockchain by solving a proof-of-work, it gets an incentive in the form of coins in the system. This incentive ensures that nodes follow the chain selection rule. If they do not follow this rule and mine on top of arbitrary chains, they will not receive the reward. Since producing proofs-of-work on those chains still requires computational power that wastes electricity, they will end up losing money.

Proof-of-work has become popular and is used in several blockchains such as Ethereum [7], Litecoin [13] and Fastcoin [10]. Unfortunately, it comes with two crucial limitations. The amount of energy it wastes is significant. O'Dwyer and Malone [60] showed that the energy used by Bitcoin is comparable to Ireland's energy consumption. Their study was performed in 2014; since then, Bitcoin's energy consumption increased [1]. This energy consumption comes primarily from proof-of-work.

Proof-of-work also limits the scalability of blockchains. Fast record processing is important for blockchains, especially for blockchains that provide cryptocurrencies since fast payments are an essential feature of any electronic payment system. In order to accept a transaction, most Bitcoin clients require that 6 blocks are mined on top of the block that contains it so that the probability that the block will ever leave the blockchain becomes low enough [26]. Since a block is mined roughly every 10 minutes [65], users must wait for an hour before being able to accept a transaction. At current rates, Bitcoin is able to process 7 transactions per second (tps). Paypal is able to process 115 tps per second, and Visa is able to handle 47 000 tps (although it only needs about 2 000 tps) [82, 55]. One intuitive approach to increase transaction throughput would be to increase the number of transactions that each block packs. However, it has been shown that increasing block size without speeding up the dissemination of blocks weakens the security of the system [55].

An alternative to eliminate the above limitations of proof-of-work is to replace it with a radically different approach. In recent years, finding alternatives to proof-of-work has become an important research topic. This is not an easy task because a sybil-proof mechanism requires wasting non-counterfeitable resources (like processing power in proof-of-work), and there are not many non-counterfeitable resources. The most popular alternative is proof-of-stake [86]. In this model, users are weighted by the amount of digital funds they possess, instead of computational power. Generally, a coin from the system is selected randomly, and the owner of that coin gets to add a block to the blockchain. By adding a block to the blockchain, the user wins a reward, which increases his chance of being selected again. Thus adding blocks to the chain will not require spending energy. Ethereum [7] is planning to start using Casper [84, 30], a consensus protocol that uses proof-of-stake.

Although proof-of-stake mitigates some critical issues of proof-of-work, it introduces new problems. Proof-of-stake gives an advantage to wealthier participants because the probability that they are chosen to mine the next block (and receive the reward) is higher. Proof-of-stake has another important flaw - the nothing-at-stake problem. When there is a fork, the best strategy is mining on top of every chain. The miner will get the reward no matter which chain wins. This thwarts the ability of the system to converge into a consistent view of the replicated ledger.

Recently, some proof-of-stake blockchains with rigorous security proofs appeared [56, 52]. They solve some of the traditional proof-of-stake blockchains problems by generating randomness securely. To accomplish this, they use different techniques. Ouroboros [56] uses a secure coin flipping protocol, and Algorand [52] uses verifiable random functions (VRF) [63] to generate randomness. However, both these models rely on strong synchrony assumptions and require that a majority of stakeholders are online constantly.

Proof-of-space [46, 18] is another alternative to proof-of-work. In proof-of-space, the miner must dedicate memory to the protocol. Thus, the non-counterfeitable resource becomes memory space. Usually, the miner starts by doing some computations until the amount of memory he wants to allocate is full. After this initialization, he is able to start mining. Each block will have a different challenge that the miner must answer with the space he allocated. A blockchain that uses proof-of-space will be more energy-efficient than a blockchain that uses proof-of-work. Accessing memory periodically requires less energy than constantly computing proofs-of-work.

To the best of our knowledge, there are no blockchains on the wild that use proof-of-space[1], and the project that is closer to achieving that goal is Chia [5]. Chia is a recent cryptocurrency proposal that combines proof-of-space with proof-of-time. It uses an elegant proof-of-space based on inverting random functions [18]. In proof-of-time, users must prove that they invested some predetermined amount of time to the protocol. In order to accomplish this, proof-of-time uses Verifiable Delay Functions (VDF) [40, 83, 70, 25]. A VDF is a function that takes a predetermined amount of time to compute, and running the function on a parallel computer will not speed up the computation. In Chia the user starts by computing a proof-of-space, then he computes the proof-of-time. Each proof-of-space has a quality level. This quality gives the

[1]One could argue that Burstcoin [4] also uses this kind of proofs. However, since they do not provide a clear specification, we will not discuss it further.

time that computing the VDF must take. Better proofs-of-space will give lower times and vice versa. If successful in practice, this approach might change the way we think of blockchain protocols. Although it still has a component that requires some degree of computation (proof-of-time), it will not require as much energy as a pure proof-of-work blockchain. Chia uses this combination of proof-of-space with proof-of-time to prevent attacks resulting from costless mining, i.e., attacks that exploit the fact that adding blocks to the blockchain is inexpensive, such as adding multiple sequential blocks at once. Proof-of-time makes mining on more than one chain more expensive than in a proof-of-stake blockchain, preventing the nothing-at-stake problem. The objective of most miners will be having good proofs-of-space instead of having to compute bigger proofs-of-time, thus making this system more energy efficient.

The main problem in Chia is that VDFs are a recent area of research with several open questions. There are two significant VDF proposals, from Pietrzak [70], and Wesolowski [83]. Pietrzak's construction requires that the public parameters are either created by a trusted third party or created by a multiparty-computation. Wesolowski's construction requires that the adaptive root assumption holds [25], this is a problem that is not well studied. It is also unknown if there might appear Application-Specific integrated circuits (ASIC) that break VDFs' security properties. Moreover, an attacker with a faster VDF implementation, even if still sequential, will be able to compromise the system.

The use of proofs-of-space in blockchain protocols is, therefore, an under-explored area, and building practical solutions remains an open problem. This thesis aims at proposing a novel proof-of-space alternative to proof-of-work that reduces the energy requirements of blockchain protocols. To study this problem, we designed, implemented, and evaluated a novel approach to proof-of-space, called Etherspace, on top of Ethereum. Our approach uses the same proof-of-space model used in Chia [18]. However, unlike Chia, we will exclusively use proof-of-space. Etherspace tries to solve a common problem of some non-proof-of-work blockchain models, generating secure randomness. Some proof-of-stake models, Algorand [52] and Ouroboros [56], deal with this problem in different ways. Our model borrows techniques from the latter and adapts them to a new context. More precisely, we use secure coin flipping protocols to generate randomness, with some elements of blockchains that use byzantine committees to serialize transactions [57, 68, 17]. To the best of our knowledge, this has never been attempted with proof-of-space models. Moreover, Etherspace solves conceptual problems that come from combining the previous techniques and problems related to the data structure were the proof-of-space data is stored. The techniques and results of this thesis have been partially presented in the following peer-reviewed publication: Diogo Castilho, Paulo Silva, João Barreto, and Miguel Matos.

Etherspace: uma abordagem proof-of-space na blockchain ethereum. In INForum 2019 - Atas do 11 o Simpósio de Informática, pages 193–204. NOVA.FCT Editorial, 2019. ISBN 978-972-8893-75-0. URL http://inforum.org.pt/INForum2019/docs/atas-do-inforum2019.

The rest of the document is organized as follows. In Chapter 2, we discuss related work as well as some background on the models we use in our approach. In Chapter 3, we explain our approach, starting with a naive approach to explain some key concepts, moving to the Etherspace approach. In Chapter 4, we discuss some implementation details, the objectives of our evaluation, and our results. In Chapter 5, we draw the conclusion, with some crucial remarks, and discuss the future work that needs to be done.

Chapter 2

Related Work

In this chapter, we discuss the properties that blockchains must ensure. We will talk about proof-of-work, and its alternatives to better understand how we can improve blockchains. In Section 2.1, we discuss what a blockchain is, and the properties they must ensure. In Section 2.2, we address Bitcoin, Nakamoto Consensus, and the role of proof-of-work. In Section 2.3, we review some alternatives to proof-of-work. In Section 2.4, we discuss models that combine proof-of-work and proof-of-stake. In Section 2.5, we present systems that use byzantine committees to increase the throughput of blockchain protocols. In Section 2.6, we discuss Ripple and Stellar, which take a different approach towards achieving consensus based on trust assumptions. In Section 2.7, we present some alternatives to blockchains. In Section 2.8, we discuss some essential ideas we used in our blockchain model. In Section 2.9, we wrap-up with a comparison between different blockchain models.

2.1 Cryptocurrencies and Blockchains

Cryptocurrencies are the most popular applications that run on blockchains. The concept of a digital currency was probably first introduced in 1983 by Chaum [33]. This model provided strong privacy, however, it relied on a central entity. Wei Dai later published b-money [38] on the cypherpunks mailing-list (same mailing-list where the Bitcoin white paper [65] was first published), where he proposed two protocols for a distributed cash system. The first of these protocols used the concept of proof-of-work to create money. Influenced by this work, Satoshi Nakamoto (pseudonym) in 2008 released Bitcoin's white paper [65].

In this work, he presented the first decentralized currency system, now known as cryptocurrencies. We can think of any cryptocurrency as a protocol that keeps a distributed ledger. This ledger must keep an immutable append-only log of transactions that is maintained by a peer-to-peer network of nodes. Ensuring that the ledger is immutable is not trivial since nodes can behave in arbitrary ways. Let us start by defining a naive solution. In cryptocurrencies, users are represented as public keys. Each transaction has a sender, a receiver and the sender's digital signature. Cryptography ensures that only the owner of the corresponding private key can create valid transactions for a given public key. This solution allows us to keep an unordered list of transactions with valid signatures. However, we cannot get any meaningful properties from asymmetric cryptography alone, since nothing ensures that nodes keep the same list of records. Garay et al. [51] presented the concept of *public transaction ledger* as a protocol that satisfies two properties, which we now present informally. Note that these are informal overviews of the properties presented in [69, 51, 55].

Persistence: If an honest node declares a transaction tx as stable in a certain time slot then the remaining honest nodes, if queried, will report tx as stable in the same time slot.

Liveness: If transaction tx was given to an honest node (to include in the ledger), then after a certain amount of time (transaction confirmation time) all honest nodes, if queried, will report tx as stable.

In order to provide these properties, the ledger is usually stored in a data structure called blockchain. It consists of a list of blocks where each block consists of a batch of transactions and a reference to the predecessor block. This reference is the cryptographic hash of that block, ensuring that two different blocks will not have the same reference (if a sufficiently strong cryptographic hash function is chosen). Because of this, any modification to the order of blocks, or their content, will invalidate the chain. In [69, 55] it is shown that a blockchain that provides the following properties ensures both persistence and liveness. Note that these again are informal overviews of the properties. Moreover, these properties only hold as long as the percentage of resources that the attacker controls is bounded (e.g., 50% of the hashing power in Bitcoin).

Common prefix: If two honest nodes, $N1$ and $N2$ (these nodes can coincide) that have the chains $C1$ and $C2$ at times $t1$ and $t2$ respectively, such that $t1 < t2$, it holds that by removing the last k blocks (k is an arbitrary value that must be chosen a priori) of chain $C1$ we get that $C1$ is a prefix of $C2$.

Chain quality: For any honest node with blockchain C, it holds that for any sequence of l blocks there are at least μ blocks (the chain quality coefficient) created by honest nodes.

Chain growth: For any honest node N, it holds that, over the course of s rounds, the number of blocks added to the blockchain of N is $s \cdot \tau$. τ is the rate of blocks added per round and is called the speed coefficient.

There are two different kinds of blockchains, permissioned and permissionless. In permissioned blockchains such as Hyperledger Fabric [19] users that contribute to the consensus protocol have to be previously approved, usually by a trusted party. This type of blockchains allows the usage of traditional Byzantine-fault tolerant consensus protocols (such as Practical Byzantine Fault Tolerance [32]) because the protocol runs among a small set of known participants. This type of blockchains provides better throughput. For instance, Hyperledger Fabric [19] is able to achieve more than 3500 tps which is a big improvement over Bitcoin's 7 tps. In permissionless blockchains such as Bitcoin [65] and Ethereum [7], any user can contribute to the consensus protocol without any prior membership established with a trusted entity. Traditional Byzantine-fault tolerant consensus protocols (used in permissioned blockchains) would not work properly in permissionless environments because they require that a majority of users are honest. Without a central entity to validate entities, an adversary can easily spawn lots of users and ensure that he controls the majority of users. Thus they would be vulnerable to sybil attacks [43]. Due to this vulnerability, permissionless blockchains usually use consensus protocols that require the user to prove possession of non-counterfeitable resources (such as computational power or wealth). Our work focuses on permissionless blockchains because we want to reduce the trust base, requiring only trusting the protocol. This is something permissioned blockchains fall short of accomplishing since there must be an entity that chooses who can participate in the protocol.

2.2 Bitcoin and Nakamoto Consensus

In order to provide the desired properties, nodes need to agree on which blocks are added to the chain. To ensure this, Bitcoin uses what is known as Nakamoto Consensus [65, 26]. Any user can attempt to add blocks to the chain, however, in order to do this, he must solve a cryptographical puzzle called proof-of-work. The proof-of-work consists of solving a puzzle whose solution is hard for the prover to compute and easy for the verifier to check. These puzzles were originally proposed as a measure to prevent spam email and denial-of-service attacks [45, 21].

Bitcoin's proof-of-work scheme is based on Hashcash [21]. It requires finding a nonce N such that $H(N, B, h-1) < T$ where H is a hash function (in Bitcoin it is double SHA-256), B is the current block, $h-1$ is the hash of the previous block and T is the target difficulty. This scheme is probabilistic since the number of different nonces a user must try, in order to find a suitable solution, grows exponentially with the number of zero bits the target difficulty begins with. Verification, on the other hand, always takes the same amount of time, i.e., the time to run the hash function once.

In Nakamoto Consensus, any node can group valid transactions, form a block and try to add it to the blockchain (after computing the proof-of-work). The first transaction of a block creates new coins that are assigned to its miner. The miner is also able to charge fees for including transactions in his block. This creates an incentive to abide by the protocol. When nodes receive a new block with a valid proof-of-work[1], they add it to their copy of the blockchain and start mining on top of it. However, due to the stochastic nature of proof-of-work, it is possible for two different miners to find two different valid solutions concurrently, creating two valid blocks at the same chain height. This leads to a fork - two different valid paths in the blockchain. To handle forks, Nakamoto Consensus uses a chain selection rule. This rule allows nodes to know which blockchain is the active blockchain (the blockchain that they must accept as valid). Nodes must choose the longest proof-of-work blockchain (the blockchain whose proof-of-work required the biggest expenditure of computation). By summing the difficulties of all the blocks in the chain, a node can estimate how much computational power was needed to compute each chain. The active blockchain will generally end up being the chain with more blocks. This combination of proof-of-work, incentives and chain selection rule are the main pieces of Nakamoto Consensus.

The target difficulty of the puzzles is not static. If the computational power invested in mining increases, blocks will be generated at a faster rate. Hence, Bitcoin periodically (every 2016 blocks) adjusts the difficulty of the crypto puzzle to keep the average block mining time close to 10 minutes.

Nakamoto [65] does an analysis of the system's resistance to double-spending attacks. In a double-spending attack, the attacker starts by issuing a transaction. After the payee accepts the transaction (and hopefully gives something in return for the digital fund), the attacker forks the blockchain and starts building a blockchain that does not have his transaction. When his blockchain is longer than the original one, all the nodes will choose it. Then the attacker is able to spend, again, the funds that he had previously used. Nakamoto models the protocol as a Binomial Random Walk between the honest users and the attacker. He concludes that, as long as an adversary does not have more then 50% of the computational power, and the payee waits for k blocks to be added to the blockchain on top of the block that contains his transaction, the probability that the attacker is able to build an alternative chain (a chain that does not contain the attacker's transaction) drops exponentially with k. By using a sufficiently large value of k in the protocol (in most Bitcoin clients this value is 6), the probability that the attacker is able to perform a double-spending attack becomes negligible. This period the user has to wait is known as transaction confirmation time, and in Bitcoin, it is roughly an hour (10 minutes times 6 blocks).

[1]Nodes will only accept a new block if it is not malformed and does not have invalid transactions [26].

Nakamoto's analysis is not sufficient to ensure the stability of the system, as it does not consider scenarios where the attacker makes honest users disagree on the current chain instance, splitting their mining power across different instances of the chain [51]. Garay et al. [51] show that the Bitcoin protocol (Nakamoto consensus) provides a public ledger in the synchronous model (as long as the majority of the computational power is controlled by honest miners). Later, Pass et al. [69], demonstrated that these properties also hold in a partially synchronous model (messages can be delayed up to an a-priori set bound). It might be tempting to think that Bitcoin provides a public ledger in the asynchronous model (messages can be delayed arbitrarily), however as a result of FLP impossibility [50] this is not possible. Pass et al. [69] prove that neither common prefix nor chain quality holds in this model.

During the early years of Bitcoin, a general purpose CPU was enough to participate in the mining process. However, Bitcoin's increasing value led miners to search for hardware that provided better hash rates - the amount of puzzles that can be solved per second. General computers started being replaced by graphics processing units (GPU) as they are able to handle more parallel calculations (allowing trying multiple nonces in parallel). GPUs, in turn, were replaced by field programmable gate arrays (FPGA), that were later replaced by application specific integrated circuits (ASIC). Taylor [81] provides a detailed description of this evolution. Mining has become so competitive that nowadays even when the miner has good hardware the odds that he will be able to successfully add a block to the chain are low. Therefore, the variance of mining payments for solo mining is significant. In order to lower the varience, miners started collaborating in solving these puzzles and sharing the rewards by participating in mining pools [74]. Pools make mining more profitable because a lower variance of rewards provides a more reliable source of income. The existence of pools is problematic for blockchains since they lead towards centralization of the system. There is some research such as [59], where a protocol for decentralized mining pool is proposed, that tries to counter this problem. A sufficiently large mining pool can compromise the underlying blockchain. GHash.io [61] had at some point more than 50% of the mining power invested in Bitcoin, making them able to launch attacks on the network. Although this pool is no longer active [41] we should try to discourage them. Works such as [64] discuss nonoutsourceable puzzles that might have the capacity to counter this problem. In mining pools, the pool manager claims the reward of the block and distributes it across miners in the pool proportionally to the computational power they dedicated. These nonoutsourceable puzzles give whoever performed the mining work the ability to claim the reward for themselves. With such puzzles the pool manager cannot trust the miners, thus preventing the formation of mining pools.

Proof-of-work is one of the most important parts of Nakamoto Consensus, and it is used in several other blockchain protocols such as Ethereum [7], Litecoin [13] and Fastcoin [10]. However, it comes with some limitations. Transaction confirmation is slow (e.g., an hour in Bitcoin) since this type of blockchains require that some blocks are mined on top of the block that contains our transaction. In proof-of-work blockchains, the size of the blocks is limited. In Bitcoin, this corresponds to the size of the block in bytes, and in Ethereum, by the amount of gas used in transactions. Gas is the measure of the computational cost of operations that are used by transactions. Ethereum can be used to run complex code, so it is essential to charge users based on the computational cost of their code instead of the size of the code (small pieces of code might result in more computational work). Changing the size of blocks without increasing block propagation speed can be harmful to the security of the blockchain [55]. These two things limit the throughput of this type of blockchains (e.g., 7 tps in Bitcoin). Moreover, proof-of-work makes the blockchain waste big amounts of energy [60]. When put together, these issues limit the scalability and applicability of proof-of-work blockchains.

2.3 Proof-of-work alternatives

As stated before, proof-of-work has some limitations. It does not scale, and it requires too much energy. To address this, some alternatives have appeared in recent years. Most of them exploit the usage of a different kind of non-counterfeitable resource instead of computational power. This resource can be real (proof-of-space) or virtual (proof-of-stake). There is an alternative that uses trusted hardware instead of wasting resources, proof-of-elapsed-time. A short description of these follows.

Proof-of-Stake was first discussed in a Bitcoin forum [86]. Nowadays it is the most popular alternative to proof-of-work. It requires users to prove ownership of wealth (in the corresponding cryptocurrency) instead of computational resources. This method is difficult to generalize because there are lots of different algorithms. Usually, there is some source of randomness that is used to select a single stakeholder[2], that will be able to add the next block to the chain. Blockchains that use this type of proofs have the advantage of consuming less energy because, unlike in proof-of-work blockchains, users do not need to waste resources to mine blocks. In proof-of-stake blockchains attacks that require holding a majority of the resources are more expensive because the attacker needs to buy a significant amount of currency, and after performing the attack the value of the currency will probably drop. There is a huge variety of proof-of-stake protocols (e.g., [78, 52, 56, 39]) with different properties. Bentov et al. [23] define a pure

[2]The probability of a stakeholder being selected is proportional to his wealth.

proof-of-stake as a blockchain that solely uses proof-of-stake (i.e., does not use proof-of-work in any step of the protocol). They classify PeerCoin (the first implementation of a proof-of-stake cryptocurrency) [78] as a pure proof-of-stake, which is inaccurate since Peercoin also uses Proof-of-Work. According to their own concept, it cannot be considered a pure proof-of-stake.

Proof-of-stake blockchains have several well-known problems [72]. Participating on the mining protocol is computationally easy, so in case a fork happens the best strategy is mining on top of every chain because we do not know which chain will end up becoming the valid chain. This is known as the nothing-at-stake problem. Daian et al. [39] handle this problem by ensuring that miners that sign multiple forks can only increase their reward by a small fraction. More aggressive alternatives such as the punishments discussed in Slasher [29] handle this problem by punishing miners that sign multiple blocks for the same depth.

Since mining is cheap, users can simulate mining different blocks. Then, they can check which block gives him better chances of being selected again and add it to the chain. This sort of behavior is known as costless simulation [71]. This problem exists in proof-of-stake systems that use the miner as a randomness source. Ouroboros [56] solves this problem by drawing randomness from a distributed coin tossing algorithm, while in Algorand [52] this problem is solved by using a verifiable random function (VRF) [63] that randomly selects users.

Proof-of-Burn is a variant of proof-of-stake where miners have to prove that they burned some coins. A user is able to burn coins by sending them to an unspendable address. Users that burned more coins have higher chances of being able to mine the next block. In such system burning coins can be seen as buying a mining rig. Buying more mining hardware, in proof-of-work blockchains, gives you a better chance of successfully mining blocks, and so does burning more coins in proof-of-burn blockchains. To the best of our knowledge, the only system that uses this type of proof is Slimcoin [66]. However Slimcoin does not rely on this proof alone, it combines proof-of-work, proof-of-stake, and proof-of-burn. Therefore, Slimcoin cannot be considered a pure proof-of-burn system. For now, it is questionable if a pure proof-of-burn system is possible since, to the best of our knowledge, the potential gains and limitations of proof-of-burn are yet to be evaluated in peer-reviewed scientific literature.

Delegated Proof-of-Spake (DPoS) is another variant of Proof-of-Stake where each user uses his stake to vote on which users are going to be the block producers [85]. These block producers are equivalent to miners in Bitcoin, they are the ones that add blocks to the chain. Although there are several systems that use this type of proofs (like EOS [6], Steemit [15] and BitShares [3]), it lacks formal analysis and evaluation. One of the potential problems of DPoS is becoming a partially centralized system, as a result of users not changing the block producers they vote for on a regular basis.

Proof-of-Elapsed-Time (PoET) is a novel technique that uses trusted execution environments (TEE) to run a "lottery" that selects the user that will mine the next block [2]. This system uses the TEE to create a timer that once expired can be attested to check if the user really did wait the right amount of time. The user that gets the shortest timer from the TEE is selected as the leader of the round. This proof might be the closest we have to Nakamoto's vision of "one-CPU-one-vote" [65]. PoET was originally contributed by Intel and is used in Hyperledger Sawtooth Lake [2]. The TEE is Intel's Software Guard Extensions (SGX) [11]. SGX protects selected code and data from disclosure and modification.

This approach requires that we trust hardware from a single vendor, Intel. Moreover, Chent et al. [34] show that by corrupting a fraction of $\Theta(\frac{\log\log n}{\log n})$ nodes an attacker is able to compromise the system. Thus the possible impact of attacks on SGX [53, 27, 76] needs to be further studied.

Proof-of-Space has been suggested as an ecological alternative to proof-of-work [67, 46, 18]. Proofs-of-space are usually defined as a protocol where a prover (miner) must prove to a verifier that he is storing some date of size N. Moreover, this protocol is divided into two stages, the initialization stage and the execution stage. There are two significant approaches to this type of proofs. The first approach is based on graph pebbling lower bounds and has two stages [46]. During the initialization stage, the prover starts by choosing a hard-to-pebble directed acyclic graph. Then he computes labels for all the nodes of the graph. The label l_n for each node $n \in V$ is given by $l_n = H(\mu, n, l_{p1}, ..., l_{pm})$, where H is a hash function and $p1, ..., pm$ are the parents of node n. Then, the prover computes a Merkle tree from the labels of the nodes and sends the root[3] of the tree to the verifier. Note that the leaves of the Merkle tree are the nodes of the graph. Next, the verifier asks the prover to give the label of some of the nodes. For each node n, the prover must respond with the label of the node l_n and the labels of all its parent nodes $l_{p1}, ..., l_{pm}$. If for all nodes, it holds that $l_n = H(\mu, n, l_{p1}, ..., l_{pm})$, the verifier accepts the commitment. During the execution phase, the verifier asks the prover to give the value of some of the labels. After, the verifier checks if with those values he is able to get the correct value of the root of the tree. If this check passes for all the requested values, the verifier accepts the proof. Dziembowski et al. [46] prove that a malicious user either needs to use N space or to compute N times the hash function in order to make the verifier accept. A rational user will always choose the first alternative because it is cheaper and faster. These are the best security guarantees for this type of proofs.

[3]For a graph with 4 nodes the root would be $H(H(n1, n2)H(n3, n4))$.

To the best of our knowledge, Spacemint [67], was never deployed. This implementation has some problems that show that proof-of-space approach is not a good direct replacement for proof-of-work. The size of the proof is large, in the order of Mbytes. The proof cannot be made completely non-interactive since the user must send the Merkle tree commitment to the verifier. Spacemint requires another miner to include in his block a transaction with the commit before a user is able to start mining. Thus, the system loses one of the nice properties of permissionless blockchains: the node's ability to join the mining protocol just by listening to the network. If all miners stop accepting new commit transactions, new nodes cannot join the protocol.

The second type of proof-of-space is based on inverting random functions [18]. We will start by showing a simple construction for this type of proof-of-space and then proceed to explain the construction presented by Abusalah et al. [18]. During the initialization, the verifier sends to the prover the description of a random function $f : [N] \to [N]$. The prover then computes the entire function table of f. During the execution phase, the verifier sends to the prover a random $y \in [N]$. If the prover replies with x such that $f(x) = y$, the verifier accepts. This construction fails to give good security properties due to Hellman's time-memory trade-offs [54]. This gives users the ability to trade space with computational power. Fiat and Naor [49] prove that an adversary with S bits of information can invert function $f : [N] \to [N]$ by making T random oracle queries to function f where: $S^2 \cdot T \in \tilde{O}(N^2)$.

Abusalah et al. [18] present a new construction that provides better guarantees than the previous simple construction. Their construction has a better lower bound: $S^2 \cdot T = \Omega(\epsilon^2 N^2)$. It consists of using a function $g_f : [N] \to [N]$ where $g : [N] \times [N] \to [N]$ is a random function and $f : [N] \to [N]$ is a random permutation. It follows that $g_f(x) = g(x, x')$ and $f(x) = \pi(f(x'))$ for any involution π[4]. If f is a function instead of a permutation then we will not need the involution, the condition becomes $f(x) = f(x')$. By nesting this construction one gets even better lower bounds $S^k \cdot T = \Omega(\epsilon^k N^k)$ where k is the number of times we nested the construction. In this model, during the initialization phase, the verifier sends to the prover the description of the function $g_f : [N] \to [N]$. Then the prover computes the function table of g_f. During the execution phase, the verifier sends to the prover a random $y \in [N]$. If the prover replies with a pair (x, x') such that $f(x) = f(x')$ (in the case that f is a random function) and $g(x, x') = y$, the verifier accepts.

Similarly to proof-of-stake, both constructions by themselves suffer from the costless simulation problem. Since computing these proofs is cheap (does not require heavy computations), an attacker can try to compute many possible blockchains, with different blocks, starting from

[4]A involution is a function that is its own inverse, such as a bit flip.

the genesis block. If one of these chains has higher quality then the current active chain other nodes will accept it.

The proof-of-space by Abusalah et al. [18] was created to be used in Chia [5], a cryptocurrency project by Bran Cohen, creator of BitTorrent protocol. It will use proof-of-space in conjunction with proof-of-time. Proof-of-time is based on verifiable delay functions (VDF) [40, 83, 70, 25]. A VDF is a function that takes a predetermined amount of time to compute. The output of the function must be quickly verifiable, and for every input to the function, there must be a unique output. Running the function on a parallel computer will not speed up its computation. Chia uses proofs-of-time to prevent costless simulation. Since each proof-of-space requires a proof-of-time the simulation will not be for free, requiring some time to be executed. Proof-of-time is similar to proof-of-work, as both require computation. However, each proof-of-work puzzle has more than one solution and we can speed up its computation by using more computational resources. Moreover, one might get lucky and solve the proof-of-work faster than expected because it is a probabilistic scheme.

In Chia, there will be proof-of-space miners and proof-of-time miners. Each block will have a proof-of-space followed by a proof-of-time. In order to participate in Chia's mining protocol the proof-of-space miner must start by initializing the space he will use in the proof-of-space (computing part of the function table of g_f). After initializing the space he is able to start mining. He will fetch the hash of the previous block. This hash will be the y value the miner must be able to invert. However, since the range of the function is quite large, miners most likely will not have any (x, x') pair such that $g(x, x') = y$. Instead, they must present the (x, x') for which $g(x, x') = y'$ and the difference between y and y' is minimal. The closest y' is to y the higher the quality of the proof-of-space. Then proof-of-space miners will propagate the best proofs. The quality of the proofs-of-space gives the amount of time the VDF in the proof-of-time must run for. Higher quality proofs-of-space will give faster proofs-of time thus incentivizing miners to invest in space rather than time. After receiving the proofs-of-space the proof-of-time miners will compute the VDF and add the block to the blockchain.

2.4 Hybrid Proofs

Some existing work explores the idea of combining proof-of-work and proof-of-stake. This type of proofs was formally discussed by Duong et al in [44] where they present a provably secure 2-hop blockchain protocol. Their system has a blockchain that is divided into two chains $(B, \tilde{B})$, one proof-of-work chain B and one proof-of-stake chain $\tilde{B}$. Every block in B must be followed by a block in $\tilde{B}$ (with the same height). In this protocol, there are miners that extend B and stakeholders that extend $\tilde{B}$. It runs in a static environment where each miner has the same amount of computational resources and each stakeholder has the same amount of stake. The system moves in proof-of-work rounds and proof-of-stake rounds. In the proof-of-work rounds, if both chains have the same number of blocks, each miner has access to a single call to the Random Oracle (equivalent to calling a hash function once). If this call meets the target difficulty he can extend B. Note that during a proof-of-work round a solution might not be found. During the proof-of-stake round, if the number of blocks in B is bigger then $\tilde{B}$ by one block, each stakeholder checks if they have been chosen to extend the chain. All of them have the same probability of being chosen. Moreover, proof-of-work blocks do not necessarily point to a stakeholder and can become orphaned. Doung et al. prove that this protocol is secure even if the adversary controls more than 50% of the computational power, as long as honest users control the majority of stake [44]. TwinCoins [35] is an implementation of a cryptocurrency that uses this type of hybrid proofs. It follows the model presented in [44] with some adjustments. There are two types of proof-of-work blocks, attempting blocks and successful blocks. An attempting block only becomes a successful block when a proof-of-stake block is added on top of it. Attempting blocks are used to measure ratios (e.g., number of proof-of-work blocks in the last 2016 blocks) that are useful for the difficulty adjustment mechanism.

Bentov et al. in [22] proposed Proof-of-Activity, a proof-of-stake extension for Bitcoin's proof-of-work. This proof is similar to the previous hybrid proofs. In this system, the proof-of-work miner starts by generating an empty block header with the solution of the puzzle. This header does not contain any transaction. It is used to derive which N stakeholders will validate it. All of the chosen stakeholders must sign the header. The N^{th} stakeholder must add transactions to the block and the signatures of the other chosen stakeholders. Unlike in the previous systems [44, 35], there is a single blockchain and adding blocks requires the combined effort of proof-of-work miners and stakeholders.

This type of systems can lead towards some miners having undesirable behavior. Their intent might not be to attack the system but rather increase their reward. Before broadcasting the proofs-of-work, a miner can check if they are the chosen stakeholder. When they are not the chosen stakeholder they can discard the solution and try to find another more beneficial solution (where they are the stakeholder chosen to extend the chain). Moreover, they can just stop approving other proof-of-work miners' solutions to stall the system, giving them more time to compute their solutions. If this behavior becomes the norm, then this kind of blockchains will have low throughput.

2.5 Byzantine committee

We can think of proof-of-work as a sybil-proof leader election mechanism. Bitcoin uses proof-of-work to select the user that can add his block to the chain, binding leader election to transaction serialization. Some interesting recent works [48, 68, 57, 17, 57] explore separating transaction serialization from finding the solution for proofs-of-work. Bitcoin-NG [48] probably was the first work to come up with this idea. In this system, the blockchain is separated in epochs where each epoch has a single leader. There are two types of blocks, keyblocks and microblocks. When a user finds a proof-of-work solution he adds a keyblock with it to the chain, becoming the leader of the current epoch. During this, he is in charge of serializing transactions by adding microblocks which contain the transactions. Miners are supposed to mine their keyblock on top of the most recent microblock, however, nothing forbids them from mining on an older microblock (this will orphan some microblocks). To mitigate this problem, rewards are given by microblocks. 40% of each microblock reward goes to the leader of the respective epoch while the other 60% goes to the leader of the next epoch. This model is able to process more transactions then Bitcoin, however like in Bitcoin, they can only be considered final when additional keyblocks are mined on top. This leads to a transaction confirmation time similar to Bitcoin.

Efficient solutions to the Byzantine Generals Problem [58] such as Practical Byzantine Fault Tolerance (PBFT) [32] require a fixed group of $3f + 1$ nodes to tolerate f faults. In the permissionless environment, this type of protocols is hard to use because the number of participants is not known and dynamic. However using them in the permissionless environment becomes possible if the $3f + 1$ required members are chosen with proof-of-work, forming a committee. Hybrid Consensus [68] uses this method in the formalization of their protocol. It requires that $x + \lambda$ proof-of-work solutions are added to the blockchain (λ is a security parameter that ensures that the previous x blocks are a prefix of all honest nodes' chain). The miners of the x first blocks will become the committee of the protocol and, using PBFT, will decide upon the serialization of the transactions.

PBFT [32] is separated in views, each view has a single leader. This view only changes when $2f$ members of the committee send a view-change message to the leader of the new view. In this protocol, view-changes only happen when the leader is not making progress (not proposing new blocks in a blockchain). In a blockchain, this gives a single user the ability to choose which blocks are going to be verified. He would be able to blacklist transactions from other users. Moreover, this protocol uses MAC-authenticated direct communication channels (with the exception of view-change and new-view messages), and according to [57] this results in a communication complexity of $O(n^2)$. This communication complexity limits the scalability that this protocol can achieve.

ByzCoin [57] adapts PBFT to the permissionless blockchain environment. Similarly to Bitcoin-NG [48] ByzCoin has keyblocks and microblocks. Every time a new keyblock is added, a view-change occurs and the leader changes. Also, the user that added the keyblock receives consensus group shares that allow him to participate in the committee. The number of views where the user is part of the committee is given by these shares. When a fork happens (two conflicting keyblocks) Byzcoin uses a deterministic function to break the tie. However, they also state that if more then 33% of miners commit to the "losing" block before receiving the "winning" block, either the system will accept the first block or lose liveness.

The use of direct communication channels does not scale when a leader needs to establish connections with a big number of nodes. Serializing a batch of transactions requires that a majority of the consensus group approve it. Since all nodes need to be authenticated all messages that nodes send require a digital signature. Distributing, verifying and waiting for other members of the committee to also verify a large number of signatures is inefficient. For solving this problem Byzcoin uses a distributed collective signing protocol called CoSi [79]. The main benefit of using this protocol is that participants instead of receiving a message of size $O(n)$ only receive a message of size $O(1)$. Moreover, the complexity of verifying the signatures becomes $O(1)$ instead of $O(n)$.

Byzcoin does not define how the reconfiguration of the consensus group is performed (replacing old members with newer members). This is an important part of the protocol because a corrupt leader might be able to stall the reconfiguration process and slow down the system. Solida [17] improves this part of the protocol. In their design when a miner finds a proof-of-work solution it becomes an external leader. He is in charge of getting himself elected into the committee. This ensures that the current members of the group are not able to stall this process.

All of the previous systems we discussed use proof-of-work. Nevertheless, a similar approach that uses proof-of-stake exists, Algorand [52]. This system uses a VRF [63] to choose the elements of the consensus group. The probability that a user is chosen is proportional to the amount of stake they have. Algorand avoids forks by prioritizing safety and finality over liveness. A block will only be inserted into the blockchain after the committee reaches an agreement. If the committee does not reach an agreement, an empty block will be added instead.

Proof-of-Work consensus group protocols such as Solida [17] assume that an adversary is not able to corrupt other participants instantly. In this context, corruption means that the adversary is able to send messages on behalf of the corrupted node. If the adversary is able to corrupt all the members of the committee, he is able to compromise the integrity of the system.

Even if the adversary starts corrupting a node by the time he presents a proof-of-work solution, by the time he is corrupted, the node will already have left the committee. Algorand achieves instant corruption tolerance (adversary is able to instantly corrupt nodes) because the identity of the committee is only known after their task is finished. The VRF ensures that only selected users are able to verify that they were chosen. Nevertheless, this result can later be verified by other users. After being selected committee members only need to send a single message.

2.6 Ripple and Stellar

Ripple [42] and Stellar [62] are exchange networks that have intrinsic cryptocurrencies. Unlike traditional consensus protocols, they do not require a majority of honest nodes or resources controlled by honest nodes. Instead, nodes require flexible trust assumptions; Each node has a list of nodes that it trusts, unique node list (UNL) in Ripple, and quorum slices in Stellar.

In the Ripple consensus protocol, the ledger is maintained by validator nodes. Each node stores the list of validators that he trusts in its UNL. Validators are responsible for adding new entries to the ledger. The protocol works in rounds. Initially, servers gather valid transactions and group them into the candidate set. Then, validators collect the candidate sets of other servers in their UNL. In the first round, they propose the candidate set that is agreed by 50% of the servers in their UNL. They proceed to vote on which transactions will be added. In the following rounds, validators must propose new candidate sets. The percentage of servers in their UNL that agree with the new sets must increase by 10% in each round. Once the percentage becomes 80% the selected transactions are added to the ledger.

As long as $\frac{4}{5}$ of the servers in the UNL of a node work correctly, Ripple consensus is correct. According to [20] there must be an overlap of at least 40% of validators in each nodes' UNL, otherwise, forks might happen. As result, users do not change their UNL [62]. Ripple provides a default list of 5 trusted validators that are controlled by Ripple Labs [31]. This results in users needing to trust a third party and leads to centralization of this cryptocurrency.

Stellar is the intellectual descendant of Ripple. The Stellar consensus protocol (SCP) uses a consensus model called federated Byzantine agreement (FBA). In this protocol, each validator chooses his own quorum slice. A quorum slice is the sufficient set of nodes that convinces a particular node of agreement(all nodes agree on the same version of the ledger). This is based on the principle that once a sufficient number of trusted nodes agree on which transactions must be added to the ledger, no honest node will ever disagree. This quorum slice can be kept in a hierarchical structure. Different sets of nodes are organized in levels. Different levels can have different agreement requirements, on the top tier a validator might need $\frac{3}{4}$ of servers in that set to agree on new transactions, while on a low level only requiring $\frac{1}{4}$. Quorum slices must overlap in order to prevent forks.

According to [31] there is not enough information to determine the fault tolerance of this system. Moreover, this model can be vulnerable to sybil attacks. A malicious user can create multiple identities and behave correctly over the years so that other validators end up trusting his nodes. After achieving a significant influence over the protocol he might be able to disturb the integrity of the system.

2.7 Alternatives to Blockchain

Tangle [73] is the data structure used to store the distributed ledger in IOTA [12]. It uses a novel model that replaces the blockchain with a directed acyclic graph (DAG). Transactions are the vertices of this graph, each transaction is connected to two previous transactions (that may coincide) forming the edges of the graph. In order to add a transaction to the tangle, a user must approve two transactions that are already in the DAG and solve a proof-of-work. Solving this proof-of-work does not give a monetary reward as all coins were created in the genesis transaction. If two transactions *t1* and *t2* are connected by an edge from *t1* to *t2*, then *t1* approves *t2*. If *t1* and *t2* are not connected by an edge but there is a path that connects *t1* to *t2*, then *t1* indirectly approves *t2*. If there are two conflicting transactions one of them will become orphaned, meaning that it will not be indirectly approved by new transactions. Each transaction has an associated weight that is proportional to the difficulty of its respective proof-of-work. How strongly a transaction approves other transactions depends on this weight, meaning that the stability of the system depends on the assumption that the majority of the computational power is controlled by honest nodes.

IOTA's developers initially decided to use their own hash function, Curl, in their system. This was not a good decision, since later on a vulnerability that allowed forging signatures was found in it [47]. This vulnerability was fixed and the hash function was replaced by a new hash function based on SHA-3. On top of this, the lack of peer-reviewed analysis (as pointed out in [31]) makes it unclear whether this system provides the same correctness guarantees that other blockchain systems have.

Serialization of Proof-of-work Events (SPECTRE) [77] is a consensus protocol for cryptocurrencies. It replaces the blockchain with a DAG, similarly to tangle [73]. The vertices of this DAG are blocks, equivalent to blockchain's blocks. New blocks should reference all 0-depth blocks that the miner knows off. Each block orders the blocks he references (past blocks), and the blocks that reach him (future blocks, i.e., blocks that will be mined on top of it). The union of these sets is called the *cone*, the blocks that the current block is able to serialize. The *anticone* is the complementary set of the *cone*. In this system if two transactions share a common input they are conflicting transactions, i.e., they spend the same money. If there are conflicting transactions, the first transaction should be accepted. To ensure this, SPECTRE uses a voting mechanism that is able to order blocks. One limitation of this model is that it does not guarantee anything if two conflicting transactions are issued at the same time. It requires that the user waits before issuing the second transaction. Their justification is that the payee is able to see the conflicting transaction before considering his transaction accepted. However, if the attacker is able to partition the network (e.g., separating honest nodes from the rest of the network), the payee might only see the conflicting transaction when it is too late.

2.8 Background

In this section, we will discuss some topics that are not directly related to blockchains. Nevertheless, they are used in some blockchain models [56] as well as on our approach. We will mainly discuss protocols where two (or more) parties generate a random bit (or a group of bits) in a secure, non-biased way.

Classical Coin flipping

A coin flipping protocol is a protocol where two (or more) parties generate a non-biased random bit. In the original setting, presented by Blum [24], Alice and Bob want to flip a coin over the telephone. They use a commitment scheme, a cryptographic primitive that allows Alice to choose a value and to send a message with a commitment of that choice to Bob. Bob can't recover the value from this message. Later on, Alice can reveal the value, and Bob can check with the commit if the value she revealed is, in fact, the value she chose initially. In order to produce a random bit, Alice starts by sending Bob a commitment of a value x_1. Then Bob sends Alice a value x_2. Alice reveals x_1 to Bob, and they can both compute $x = x_1 + x_2 \mod 2$.

The classical approach to this problem suffers from a problem, after receiving x_2 from Bob, Alice can abort before revealing her commit. This behavior prevents Bob from knowing x_1 and computing x. In the two-party setting, this problem is hard to avoid. Following, we will discuss the n-party setting.

Coin flipping using Publicly Verifiable Secret Sharing

To avoid the aborting problem from the classical coin flipping setting, Kiayias et al. [56], run the protocol in a multi-party setting and use a publicly verifiable secret sharing (PVSS) scheme

[75]. A PVSS is essentially a verifiable secret sharing (VSS) scheme. A VSS scheme works in two phases. In the distribution phase, a dealer divides a secret s into n shares $s_1, ..., s_n$ and sends them to participants $p_1, ..., p_n$. Then, in the reconstruction phase, the participants can reconstruct the secret by pooling together the shares from at least t participants. Moreover, participants can check if the share they received is correct. VSS schemes are prepared to resist malicious users, such as a dealer that sends incorrect shares, and participants that submit invalid shares.

In a PVSS [75], the goal is that everyone (not just the participants), can verify the protocol. The dealer sends encrypted shares $E(s_i)$ as well as a non-interactive zero-knowledge proof (NIZK) that $E(s_i)$ is the encryption of a valid share. During the reconstruction phase, participants must publish s_i and a NIZK that prooves that s_i is the correct decryption of $E(s_i)$. Like in the VSS scheme, by combining at least t shares, it is possible to recover the secret.

In a coin flipping protocol that uses PVSS, the participants start by sharing the shares of the secret (a string chosen at random). After this phase, even if the dealer of the secret aborts and as long as there is an honest majority amongst the participants, they can reconstruct all secrets. By combining all the secrets, they create a random bit. In Ouroboros [56], Kiayias et al. use this protocol in a way that is not adaptable to most blockchain models. Ouroboros is a proof-of-stake blockchain, that is divided into epochs that are divided into slots. A random coin is selected for each slot; the owner of this coin is the slot leader. The slot leader is allowed to generate a block and insert it into its slot. During each epoch, each slot leader (only the leaders of the slots of the current epoch participate in this protocol) must choose a secret, split it into shares, and distribute them. Then, leaders must reveal their secrets. If any leader does not disclose its secret, the other leaders will use the shares to reconstruct it. The result of this protocol is a random string that is used to select at random the slot leaders of the next epoch. This protocol exploits the fact that the identity of the slot leaders is known at the start of each epoch, i.e., we know who is going to add the following blocks. However, in most blockchain protocols, we do not know who will be the users generating the following blocks.

Syta et al. created some coin flipping protocols that are resistant against Byzantine adversaries, RandShare and RandHound [80]. RandShare is a protocol that securely creates a random string as long as at least $2f + 1$ out of the $3f + 1$ participants are honest. Nevertheless, this protocol is not scalable, with a message complexity of $O(n^3)$, where n is the number of participants.

RandHound is similar to RandShare, as it also creates a random string, and can tolerate f malicious nodes, but uses a client/server model. RandHound is able to scale better than

RandShare by dividing nodes into smaller sub-groups. Nodes send the shares to the members of their sub-group, instead of sending them to all nodes. This reduces the communication complexity to $O(nc^2)$, where c is the average size of the sub-group. Moreover, the client can produce a log that proves that the random string was created correctly. RandHound ensures unbiasability, and unpredictability, even if an adversary controls the client and at most $f-1$ node. Nonetheless, RandHound does not account for clients that abort. If the client aborts, the protocol will not return the random string. We will use this protocol in our approach with a small change that allows us to account for the case where the client aborts.

2.9 Summary

There are many protocols for blockchains in permissionless environments with different properties. To finish this section we present a summary of the main properties of some blockchain models in Table 2.1

The classical method, proof-of-work, has some limitations that impair the potential of blockchains. Alternatives such as proof-of-stake and proof-of-space are vulnerable to simulation attacks because computing simulations is cheap. A possible solution is combining these proofs with proof-of-work, trying to get the best of both worlds. Existing methods that combine proof-of-work with proof-of-stake such as [22, 35] require that different users take turns collaborating to extend the chain. This is something that must be avoided because it gives the second user the ability to stall the system. Moreover, they inherit the energy consumption from proof-of-work and, like proof-of-stake, are unfair to smaller stakeholders.

Blockchain protocols that use committees [57, 17, 68] are an interesting alternative to proof-of-work. These protocols achieve better throughput, however, they require that the members of the committee are online.

Protocol	Pros	Cons
Proof-of-work	Well studied and proven to work in pratice.	Poor throughput, high energy consumption.
Proof-of-stake	Energy efficient. Does not require wasting resources.	Nothing-at-stake, costless simulation and unfair to smaller miners.
Proof-of-elapsed-time	Energy efficient. Does not require wasting resources.	Requires having to trust hardware from a single vendor. It is not safe if a small fraction of nodes is corrupted.
Proof-of-space	Energy efficient. ASIC resistant.	Costeless simulation.
Hybrid Proofs	Attacks require a majority of both types of resources.	Energy consumption similar to proof-of-work.
Byzantine committee	Good throughput.	The members of the committee must be online.

Table 2.1: Blockchain models summary.

Chapter 3

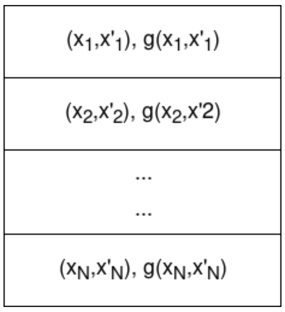
Etherspace

In this chapter, we start by presenting a naive approach where we replace proof-of-work with proof-of-space in the Ethereum protocol. Then we discuss why this approach fails. We then present an abstract design of Etherspace, which solves the identified problems. Finally, we describe two concrete implementations of that abstraction.

3.1 Intuition

We aim at implementing a permissionless blockchain protocol that is more energy efficient, and that provides similar levels of performance and security as proof-of-work blockchains. Some blockchains such as Ouroboros [56] and Algorand [52] achieve the energy-efficiency goal at the expense of tight timing assumptions, which in turn harm the security of the protocol. To accomplish our goal, we will use an underexplored alternative to proof-of-work, proof-of-space.

Replacing proof-of-work with proof-of-space is not an easy task; multiple problems arise when making this change. In a proof-of-work blockchain, miners race to find the solution to the proof-of-work puzzle. The first miner that finds the solution broadcasts its block, and other miners will start mining on top of that block. In a proof-of-space blockchain, all miners try to add their block to the chain. Therefore, we must be able to select one of these blocks as the winner of that round. Moreover, we need to prevent miners from compromising the winning criteria. In our approach, the winning criteria is the quality of the proof. Miners must generate answers to proof-of-space challenges; to each answer, we assign a quality level. This challenge must be randomly generated; if miners can bias the challenge, for instance, by drawing randomness from the blockchain, they can compromise the winning criteria. To ensure the unbiasability of the challenge, we draw randomness from a source other than the blockchain. To the best of our knowledge, this is the first proposal of a proof-of-space blockchain that uses this approach.



(x_1,x'_1), $g(x_1,x'_1)$
(x_2,x'_2), $g(x_2,x'2)$
... ...
(x_N,x'_N), $g(x_N,x'_N)$

Figure 3.1: This figure depicts the table of the function g_f.

3.2 From a Naive Approach to the Requirements of Etherspace

In this section, we discuss a naive approach that we will use as the base for our solution. This naive approach introduces some important aspects of our model.

From the two proof-of-space models that we discussed in Section 2.3, we have chosen the one that is based on inverting random functions [18]. The model based on graph pebbling lower bounds gives better security guarantees [67], however, it is not possible to make the proofs completely non-interactive. This hinders its usability as a proof-of-work alternative because new miners will have to be approved by previous miners. Moreover, this proof-of-space model has proofs with larger sizes. The model based on inverting random functions [18] can be made completely non-interactive, making it a better choice.

Before the miner is able to start mining, he must allocate his target amount of memory, which corresponds to computing the function table of g_f [18]. This is done as follows - the miner starts by computing the table of f. Every time he finds a collision in function f, it means that he has a suitable input pair (x, x') for the function g, i.e., the condition $f(x) = f(x') : x \neq x'$ must hold. Then he must compute $g(x, x')$ and store it along with x,x' as portrayed in Figure 3.1. When the table g_f reaches the desired size, the miner can discard the table of f and start mining.

In this approach, the last n bits of the hash of the last block will be used as the proof-of-space challenge y, i.e., the value that the miner must invert. Since the size of the full table of g_f is very large, and because no miner can generate the entire table, there is the probability that no miner has the pair (x, x'), such that $g_f(x) = y$, for a given challenge y. Instead of presenting the exact inversion of y, the miners will have to present the inversion of y', from the locally stored pairs, such that the absolute difference between y and y' is minimal. Then, we can assign quality levels to each answer of the miners. The bigger the difference between y and y' the lower the quality of the proof-of-space. The worst possible answer for a given challenge will be given either by 0 or 2^n. The value of the worst possible answer will be used as a reference for the

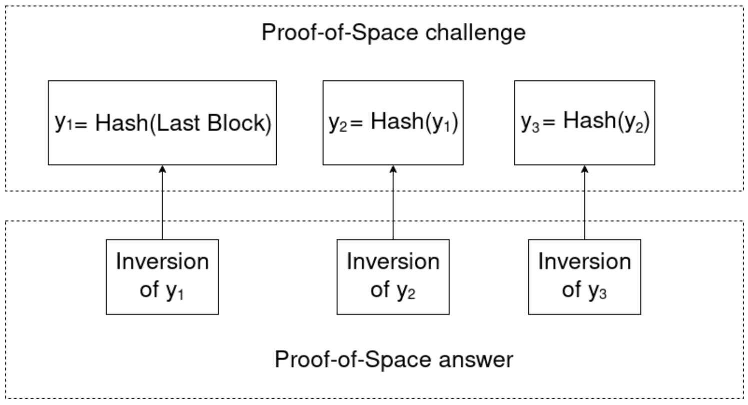


Figure 3.2: This figure shows how the vales the miner must invert are generated, and an example of a proof-of-space with three iterations.

quality of each proof-of-space.

Since there is the probability that different miners have answers with the same quality for the same challenge y, we make them invert more than one challenge. This will reduce the odds of a tie. The hash of the last block gives the first of these values; the remaining ones are given by the hash of the previous challenge, as can be seen of Figure 3.2.

In this approach, blocks will be slightly different from blocks in the Ethereum blockchain. Blocks in Ethereum do not have signatures. At first, this might seem like a security flaw; without signatures, integrity is not guaranteed. However, we have to take into account the fact that blocks must contain a valid proof-of-work. If we change a single bit from a valid block, the hash of the block will change, invalidating the proof-of-work. So in order to change an existing block, a miner would need to compute a new proof-of-work. We can conclude that proof-of-work gives us integrity guarantees over the block. Nevertheless, we do not have non-repudiation. It is, therefore, possible to mine blocks on behalf of other miners. Although this is possible, it is unlikely that there are miners wasting resources while someone else gets the reward. By removing proof-of-work from our model, we lose integrity. Thus, in our approach, all blocks must contain the signature of the miner that created them.

One of the most critical parts of blockchain protocols is the chain selection rule. In our approach, we will use a variation of the chain selection rule used in Nakamoto Consensus [26]. While on Nakamoto Consensus, the active chain is the chain that needed more computational power, in our model, the active chain will be the one that needed more space (i.e., the sum of the proof-of-space qualities of all blocks in the chain is higher).

At first, this approach might seem correct. However, it has two flaws. By using the hash of the last block as the challenge of the next block, we are giving the miner control over the challenge of the next block. Allowing a malicious miner to manipulate the challenge of the next block. Before adding his block to the chain, the miner can check if he has a good answer (i.e., an answer with high quality) to the proof-of-space challenge that is generated by his block. If his answer has low quality, he is able to change the block that he is adding to the chain, by changing the order of the transactions, which will change the hash of the block and generate a different proof-of-space challenge.

The second problem is that nothing ensures a time interval between blocks. If the time that a miner takes to create a block is less than the time that it takes to propagate the block to the network, it will be harder for the blockchain to converge. Besides, since proofs-of-space can be generated quickly because they essentially boil down to looks on local disk, a malicious miner can quickly create a long sequence of blocks. Even if the quality of the proof-of-space of each block is low, the total quality of the chain might be higher than that of a chain with few blocks with proof-of-space with high quality. Due to these problems, a malicious miner with limited space can rewrite the blockchain starting at any point.

After considering these problems, we conclude that any blockchain needs to satisfy the following requirements:

1. Concurrent deterrence - No miner should be able to gain an advantage by generating blocks concurrently for the same chain height.

2. Gradual generation - No miner should be able to create long sequences of continuous blocks quickly on top of the blockchain.

3. Chain integrity - No miner should be able to rewrite the blockchain by producing a block with a high-quality proof-of-space for a lower height.

The first two requirements are related to the problems of our naive approach. Concurrent deterrence is related to the first problem. In this blockchain model miners have an incentive to to produce blocks concurrently. When they create a block, they get to peak at the challenge of the next block. Creating multiple blocks allows miners to select the block that generates the challenge for which they have the best answer. In a traditional proof-of-work blockchain, miners do not win anything by producing blocks concurrently for the same height. The probability that they solve one proof-of-work puzzle does not increase. Gradual generation is related to the second problem. If we can ensure that there is a time interval between each block, we also assure that miners cannot generate a continuous chain of blocks quickly. Although the third requirement is not related to the problems of the naive approach, it is essential and transversal to blockchains in general. We want to prevent adversaries from rewriting the blockchain.

3.3 Etherspace

In this section, we present the general approach used in Etherspace, which solves the previous problems.

To solve the problems of the naive approach we start by defining an oracle. All nodes (note that miners are the subset of nodes that add blocks to the blockchain and some nodes do not add blocks to the blockchain) have access to this oracle, which ensures all the requirements presented in the previous section. The oracle recieves as argument a number $h > 0$ that represents a blockchain height, and returns the proof-of-space challenge for which the block at height h must contain an answer. Moreover, the oracle only returns the challenge for height h, after an interval of t seconds has passed since it returned the challenge for height $h-1$. Initially, the oracle only returns the challenge for height 1, and for any other height, it returns nothing. When a node asks the challenge for height 1 for the first time, the oracle starts a timer. After t seconds, when the timer finishes, the oracle, when prompted, returns the challenge for height 2. When a node asks for the challenge for height 2 for the first time, the oracle will once again start the timer, and so forth. In the next subsections, we present two possible implementations for this oracle, but before, we discuss how the oracle satisfies the requirements.

In a system that uses the naive approach discussed in the previous section, miners could bias the challenge of the following block by generating blocks concurrently. Thus, granting malicious miners an advantage. This problem comes from using the blockchain as a source of randomness. Etherspace uses the oracle as a source of randomness. Therefore, preventing miners from biasing proof-of-space challenges and assuring concurrent deterrence.

The oracle directly ensures gradual generation. It is only possible to create blocks for the top of the blockchain when the oracle releases a challenge. If the oracle only releases the challenge on strict intervals, miners will always have to wait before creating another new block for a higher blockchain height. The speed by which miners can add continuous chains of blocks to the top of the blockchain becomes bounded by the oracle.

Proof-of-space grants the last requirement. In order for a miner to cause a rewrite of the blockchain from an older block, most of the blocks of his private chain would need to have proofs-of-space with higher quality. However, he will only be able to produce better proofs-of-space for every height if he has more space dedicated to the protocol then the others. We can use a variation of the common assumption used in blockchains; the resources of the attacker are bounded (persistent memory in this case). Further research is needed to define the bound that this approach can tolerate. The closest blockchain model, Chia [37], is reported to be secure as long as honest miners control at least approximately 61.5% of the space dedicated to the protocol.

This oracle enforces a strict notion of rounds in our protocol. In each round, every miner will try to add its block to the chain. This will lead to a large number of blocks flooding the network. All but one of the blocks will be orphaned since there can only be a winner per round. In order to reduce the number of blocks that will end up orphaned in the network, each node keeps a record with the quality of the best proof-of-space it saw in each round. If a node receives a block with a proof-of-space with lower quality than the best of that round it will discard it immediately. Conversely, if the quality of the proof-of-space is higher than the best, the node will update the register and broadcast the block.

3.3.1 Etherspace using SGX

The first option to implement our oracle is using a trusted execution environment (TEE) such as SGX [11]. We can use the TEE to generate the same timer (across all nodes). When the timer finishes, it will return the proof-of-space challenge for the current blockchain height (the SGX will ensure that this challenge is the same for all nodes in the network). Moreover, we can ensure by using the TEE that the challenge in incoming blocks is correct.

Using this approach would require miners to trust the protocol as well as hardware from an individual vendor. Moreover, if an attacker was able to compromise its TEE into returning the challenges earlier, he could use the time advantage to find a better proof-of-space. Due to these problems, we decided not to explore this alternative.

3.3.2 Etherspace using Coin Flipping

This section, covers the second possible oracle implementation. This is the approach we chose to implement and further explore in this work. It consists of combining the usage of committees, similar to what is used in Byzcoin [57], with a coin flipping protocol, akin to what is used in Ouroboros [56], to generate a random challenge for the next block. As long as the space resources of a malicious miner are bounded, we know that the majority of the members of the committee will be honest. This protocol requires that miners communicate with each other; its execution will take some time that is limited by the latency of the network. This way, we ensure that there is a time interval between each block, bounded by the time that the protocol takes. The protocol that we use for coin flipping is Practical Hound, a variant of the RandHound protocol [80], further discussed in Section 3.4. This protocol has a useful feature, it generates a log. This log can be used to check if the challenge was created correctly.

Since a large number of miners add blocks asynchronously to the blockchain, the last k blocks

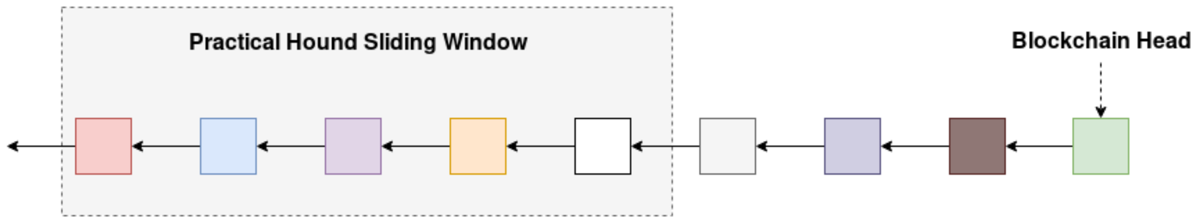


Figure 3.3: Example of a sliding window of size $w = 5$ and $k = 4$

might change. Let H be the height of the block at the top of the blockchain. If we select a big enough k, we get that, with high probability, the blocks below height $H - k$ will never leave the blockchain. Then, we can select a fixed-size continuous group of w blocks from height $H - k$ to $H - (k + w)$. This fixed-size group of w blocks will be the Practical Hound *sliding window*, described in more detail in the next section and similar to what is used in Byzcoin [57]. In Figure 3.3 we can see an example of a *sliding window*. The miners of the blocks in the *sliding window* will run the Practical Hound protocol to generate the challenge of the next block. When an iteration of the Practical Hound protocol ends, the challenge of the next block is released, the *sliding window* moves one block forward, and a new iteration of Practical Hound starts. After each round finishes, all nodes will learn of the challenge in two different ways. Either they will receive the challenge creation log, or they receive a new block. In this approach, whenever a node sends a block, it also sends the challenge creation log. Even if a malicious miner wanted to *hide* the result of the challenge, he would need to reveal it in order for its block to be accepted.

In this model, the blockchain will end up having a slightly different structure. Practical Hound ensures that there will be a log of the creation of the challenge. This log proves that the challenge was created correctly. All blocks will have a reference to a proof-of-space challenge and its creation log. Each Practical Hound iteration depends on the challenge of the previous iteration. In turn, we get that the challenge creation logs will be connected and that each iteration cannot start before the previous one finished. Thus, apart from maintaining the regular blockchain, nodes also maintain a parallel chain of proof-of-space challenges, as depicted in Figure 3.4.

3.4 Practical Hound

Out of the two protocols we discussed in Section 2.8, we decided to explore RandHound. RandHound has better scalability than RandShare and generates a log that allows checking whether or not the randomness was created correctly. Nevertheless, in order to use RandHound, we had to make some modifications. RandHound uses a client/server model. The client is the partici-

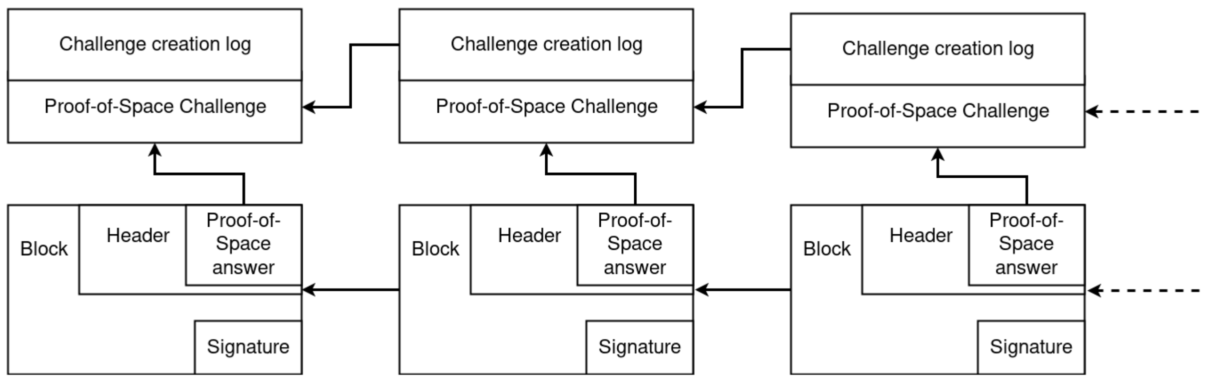


Figure 3.4: New structure of the blockchain

pant that divides the servers into smaller sub-groups and moves the protocol forward. In order to bias the protocol, the client would need to control $f - 1$ other nodes, but he can prevent the protocol from finishing by aborting.

We can consider the client in RandHound the leader of the protocol; we will use this denomination from now on. We need a mechanism that enables us to detect if the leader is not making progress and replace it. To achieve this, we can use a view-change mechanism like in Practical Byzantine Fault Tolerance [32]. This mechanism allows us to replace the leader if he is not making progress. We will call Practical Hound to this version of RandHound.

Let n be the size of the committee, let h be a height of the blockchain (the blockchain height for which the challenge must be connected), let v be the current view number (which will start as zero for every new committee), let C be the most recent proof-of-space challenge, and let H be a secure hash function. The leader of the new committee will be given by $H(h, v, C) \mod n$. If, at any step of the protocol, the leader stops making progress, the members of the committee will start sending *view-change* messages. Once a node receives $2f$ *view-change* messages, he will increase the view number and check who is the leader of the new view. The leader of the new view will send a new initialization message to signal the start of the Practical Hound run of the new view. Whenever a node receives a *view-change* message, it will add it to his copy of the log.

When the current committee finishes creating the challenge at height h, the *sliding window* moves one step forward, changing one of the members of the committee. The new committee will start creating the challenge for height $h + 1$. The Practical Hound run for height $h + 1$ cannot start before the challenge of height h because each run uses the challenge of the previous run as entropy for selecting the leader. The rest of the Pratical Hound protocol is equal to RandHound [80].

3.5 Vulnerabilities

To the best of our knowledge, our approach has two possible security vulnerabilities. The first affects the general Etherspace approach, while the second comes from the implementation of the oracle. In our model, blocks can create more than one block for each height and more than one block with the same proof-of-space. A malicious miner can try to compromise the protocol by broadcasting a large number of different blocks for the same heigh with the same proof-of-space. If the quality of the proof-of-space is low, then miners will drop these blocks, as discussed in Section 3.3. Moreover, if the quality of the blocks is high, then the possible effects of this attack are not clear.

To mitigate this vulnerability, we can extend the responsibility of our committee. Besides creating randomness, it could also serialize transactions, like in Byzcoin [57]. The committee would choose which transactions are to be added for each blockchain height. Moreover, we could remove uncles entirely from our protocol. With these changes, miners can only produce one block for each height using the same proof-of-space. Even if a miner produces a second block using his second-best proof-of-space, the new block will most likely be dropped by the protocol.

The second vulnerability comes from implementing the oracle using a coin flipping protocol. Note that this vulnerability will not be present when implementing the oracle with a TEE. A malicious miner can choose a point in the blockchain, and start growing a private chain simulating the committee (even if its blocks have low quality proofs-of-space). At some point, if it continued to add blocks, the miner would control the committee of his private chain. The miner would then be able to produce challenges at a faster rate than in the active chain and eventually outgrow it, making his private chain the active chain. He would be successful even if he only added low quality blocks because, eventually, the sum of the qualities of all blocks in his chain would be higher.

This vulnerability is trickier to solve. It comes from the fact that the reconfiguration of the committee is not ingrained into the blockchain. In our approach, only the chain selection rule can solve a disagreement on who is the committee. We need to ensure that there is no room for doubting who the next committee will be. By modifying the protocol, we can ensure that there is an agreement on the reconfiguration of the protocol. We can do reconfigurations in a similar way to Solida [17]. Let h be the height for which the committee will produce a challenge, w the size of the committee, and k the security parameter that defines the number of blocks that we have to prune from the blockchains of all honest nodes to find a common prefix. The committee that will create the challenge for height h consists of the miners that generated the blocks from height $h - k$ to $h - (k + w)$. However, before producing the next challenge, the miner that will enter the committee for creating the challenge for height $h+1$ must be selected in a similar way to how the committee reconfiguration is done in Solida. To do this, the miners that generated the blocks from height $h-(k-1)$ to $h-(k+w-1)$ will run a consensus protocol to select the new member. The miner of block $h-(k-1)$ will be the *external leader* in charge of selecting the new member. At first, this might look like a bad decision. There is a probability that nodes do not have the same block at height $h-(k-1)$. However, due to the different qualities of the proofs-of-space of the different blocks, we can rank the different possible *external leaders.* Miners will only accept as the *external leader*, the miner that is trying to elect himself as a member of the next committee if he has the best proof-of-space amongst all miners that are trying to elect themselves. Once a miner claims to be the *external leader*, he must actively work to perform the reconfiguration. To protect the protocol from *external leaders* that stalls (e.g., a malicious miner that has the highest proof-of-space for height $h-(k-1)$, makes his leadership claim but is trying to prevent the protocol from moving forward), we allow the remaining members of the committee to vote on removing the *external leader.* As a consequence, the claim of the miner with second-highest proof-of-space will be accepted. Once an *external leader* is accepted and elected as the new member, the block at height $h-(k-1)$ is finalized, and the miners of blocks $h-k$ to $h-(k+w)$ will execute the Practical Hound Protocol. The blockchain will also have to include the log of messages of the new member selection protocol, which can be stored alongside the Practical Hound log.

With this mitigation to produce two different valid new members for the next committee, an attacker needs to control more than f of the members of the committee. Our previous assumption covers this scenario; honest miners control more than $2f+1$ of the members of the committee.

It is necessary to explore these vulnerabilities further. In this work, we were not able to study them enough. A continuation of this work will have to address them adequately.

3.6 Summary

This chapter discussed the key design ideas behind Etherspace. Etherspace is a novel proof-of-space approach to blockchain protocols that aims at reducing the energy requirements of blockchains. To accomplish this goal, it combines coin flipping to create randomness and byzantine committees to solve problems that come from replacing proof-of-work with proof-of-space.

Chapter 4

Evaluation

In this chapter, we evaluate Etherspace. The main goal of the evaluation is to answer the following questions:

- How much energy does Etherspace consume? This question is essential to our evaluation since the primary goal of this work is reducing the energy consumption of blockchain protocols.
- Is Etherspace able to converge? We want our approach to offer similar levels of security to a proof-of-work blockchain, and thus ensure that the blockchain converges.
- How much throughput is Etherspace able to handle? We want our approach at least to have the same performance as Ethereum.

To evaluate our approach, we deployed a private Ethereum network with 50 miners, spread across three machines, with the following specifications:

- one machine with an Intel(R) Xeon(R) Gold 6138 CPU with 20 cores with a clock rate of $2,00$ GHz and 62GB of RAM
- one machine with an Intel(R) Xeon(R) CPU $E5-2660$ v4 with 28 cores with a clock rate of $2,00$ GHz and 62GB of RAM
- one machine with an Intel(R) Xeon(R) CPU $E5-2648L$ v4 with 28 cores with a clock rate of $1,80$ GHz and 31GB of RAM

For the workload, we used real Ethereum transactions (from block 0x50f8f4 to block 0x5f308) taken from Etherscan [9]. The transactions were fed to the miners by a Python script running in the corresponding machine. We ran each experiment for 1 hour and discarded the first and last ten minutes of each experiment.

4.1 Implementation

To test our approach, we implemented the Etherspace prototype on top of the Go Ethereum implementation version 8. To instantiate the functions f and g required for the proof-of-space, we followed the proposal of Abusalah et al. [18], and used truncated AES with mode CBC (by truncating the output of AES, it becomes a one-way function). The 128 bit block that serves as input is formed by a nonce (this nonce has a fixed size of n bits), preceded by $128 - n$ zeros. We use as initialization vector (IV) the hash of the nonce combined with the address of the miner. After computing AES, we will remove the most significant bits of the result, leaving only n bits.

To store the proof-of-space data, we started by using the database used in geth, goleveldb[1], an implementation of the LevelDB[2] key-value store. We used as database key the value that the miner is able to invert (y) and, as the value, the input pair that originated it (x, x'). We chose to store it like this because we only perform searches on this database based on the inversion y. Unfortunately, the time to construct the proof-of-space using goleveldb was significant. Besides, most of the time, miners will not have the exact inversion of the challenge, and goleveldb does not have a successor or a predecessor method. The only solution becomes iterating the table until we find the closest value, which will have a time complexity of $O(n)$, where n is the number of entries in the table. Since the miner has to invert m values in each proof-of-space, the time complexity of computing the proof-of-space will be $O(mn)$.

To reduce the time that constructing proof-of-space took with goleveldb, we decided to use a B-tree that offered successor and predecessor methods. We based our B-tree implementation on a publicly available implementation[3]. We had to modify the original implementation because it was an in-memory B-tree, and we needed persistent storage B-tree. By using a B-tree, we reduced the time complexity of computing proof-of-space from $O(mn)$ to $O(m \log_n)$

In Ethereum, blocks do not have a size limit. However, in practice, they are limited by the amount of gas used in transactions. The gas limit is not constant; miners can increase it or decrease it over time. We decided to use the gas limit registered in 30/09/2019, which was 9 976 323 in the Ethereum blockchain, according to Etherscan [9], as the starting gas limit.

For simplicity, transactions do not affect the state of the system in our experiments, i.e., they are not executed.

[1]https://github.com/syndtr/goleveldb
[2]https://opensource.googleblog.com/2011/07/leveldb-fast-persistent-key-value-store.html
[3]https://github.com/google/btree

4.1.1 Probabilistic Approximation in Etherspace

Testing the protocol with a large number of miners would require large amounts of space, which is not practical to obtain for our experiments. Therefore, we decided to replace the proof-of-space with a probabilistic approximation. In order to do this, we measured the distribution of the quality of the proofs-of-space. We instantiated the size n of the values required by the proof-of-space (x, x', y) to be 16^4. Theoretically, with this parameter, the maximum number of entries of the table of g_f is 65 536 (a single account cannot produce the entire table of g_f). In reality, the tables of all accounts will be different, and some of them will not contain inversions of some of the possible values. We started by generating the proof-of-space table of 500 addresses. These tables ended up having a maximum of nearly 48 000 entries. We also instantiated the number of inversions that each miner must present for each proof to be 100. The first of the values the miner must invert was created at random. The next ones were created by applying two times the hash function Keccak-256[5] to the previous value.

We created a total of 42 693 proof-of-space challenges and measured the quality of the answers generated by each table. The quality of the proofs-of-space follows a normal distribution with a mean of 95 514.2 and a standard deviation of 470.7. Then, we replaced the proof-of-space in our Etherspace prototype with a random number generator that follows this distribution.

4.1.2 Probabilistic Approximation Ethereum

In order to run experiments with a much larger number of miners than our available number of physical nodes, we disabled the proof-of-work component. This enabled co-locating multiple miners on the same machine. Proof-of-work was replaced by a probabilistic mining selection process, implementing a poisson distribution. After tunning the process, we got an average block time of 14.26 seconds which is approximately one second higher than the current Ethereum average block time (i.e., approximately 13 seconds) and similar to last year's block time [9].

4.1.3 Practical Hound Prototype

For our tests, we did not develop a full Practical Hound implementation due to time constraints. We consider that our system is in a *steady state* when the Practical Hound committee does not go through reconfiguration (changing the leader). In our evaluation, we only take into account the steady state, so we did not implement the *view-change* mechanism. We use 30 as the k parameter, the difference between the height of the head of the blockchain and the block of

[4]In a real proof-of-space implementation, this value would be higher. However, reducing the size of the problem will most probably not affect the distribution of the quality significantly.

[5]We chose to use Keccak-256 because it is the hash function used by the Etheruem protocol.

the last member of the committee. Note that choosing a safe k parameter will require further research. For our testing purposes, we only need a high value.

The Practical Hound protocol requires the usage of collective signatures known as CoSi [79]. Since we only use honest miners, we did not implement this signature scheme. The computational cost of Cosi in a group of 512 members is approximately 0.5 seconds [79]. We used smaller groups in our tests, so we expect this time to be negligible with respect to the time to run the protocol, which requires the exchange of messages in a WAN environment. Moreover, not using Cosi in our prototype does not reduce message complexity. We can conclude that the time difference between a Practical Hound round that uses Cosi, and one round that does not, is negligible. Even though we do not use Cosi, all messages that are sent through our prototype still contain an ECDSA signature that is verified by all miners that receive it (even when using Cosi, Practical Hound still requires that all messages are signed).

We also implemented Schoenmakers's PVSS scheme [75]. We started by trying to reuse EPFL DEDIS lab's Cothority framework[6] implementation, but unfortunately, we could not reuse the code since they used a different elliptic curve from Ethereum, and some of the required operations were not implemented in Ethereum. Instead, we implemented a similar algorithm that only used what is available in Ethereum, based on EPFL DEDIS lab's PVSS implementation.

Moreover, we do not check if the log produced by the Practical Hound protocol is correct. In our testing setting, we know that this log will always be correct because all miners are honest.

Naturally, in a full Practical Hound implementation, these simplifications are not acceptable. However, for our testing purposes and because of the reasons provided above, these simplifications do not affect the main experimental conclusions.

4.1.4 Practical Hound Committee Size

The size of the Practical Hound committee size is an important parameter. Moreover, even with a fixed committee size, we can change the number of sub-groups and the cardinality of sub-groups. The sub-groups should be balanced, i.e., approximately have the same number of members. Increasing the size of the committee increases the number of malicious participants that the committee tolerates, at the expense of increasing the running time of Practical Hound, and in turn, the time between blocks. Note that increasing the time between blocks is not necessarily a bad thing as it allows bigger blocks with more transactions.

In an ideal setting, we want a large committee. If the Etherspace network had the same size as the Ethereum Mainnet (7 369 nodes [8]), a committee of 100 nodes would only account for

[6]https://github.com/dedis/cothority

Committee Configuration	Number of sub-groups	Participants per sub-group	Total number of participants	Average Block Time (sec)	Standard deviation (sec)	Number of blocks
Configuration 1	2	4	9	14.89	2.53	161
Configuration 2	4	4	17	18.35	4.19	131
Configuration 3	6	4	25	23.88	6.74	101
Configuration 4	2	6	13	18.15	3.72	132
Configuration 5	4	6	25	24.78	5.49	97
Configuration 6	2	9	19	23.17	5.55	104

Table 4.1: Configuration of the Pratical Hound committee used in our experiments.

approximately 1.36% of the nodes. However, the available resources limit the number of nodes and, in turn, the size of the committee. We are only running 50 nodes, and we want no more than 50% of the nodes participating in the committee.

We tested our protocol with different committee configurations to determine how it impacts the protocol (e.g., its effect on block time and throughput). Table 4.1 shows for the different configurations used, the number of blocks added to the blockchain during the experiment, the average block time, and the standard deviation. Note that the leader of the protocol also counts as a participant but is not in any of the sub-groups.

From the results in Table 4.1, we can see that increasing the size of the sub-groups has a significant effect on the average block time. While configuration 6 and configuration 3 have a similar block time,the latter has more participants. This result is expected since the message complexity of this protocol is $O(nc^2)$, where n is the number of participants, and c is the average size of the sub-groups. In a real setting, using a large number of small sub-groups seems to be the best alternative if we want shorter block times. Note that if a miner has mined more than one block in a Practical Hound sliding window, he will play the role of more than one participant in the Practical Hound protocol. Since the participants of this protocol make up a significant amount of all miners (18% for configuration 1 and 50% for configuration 3 and 5), it is highly probable that some miners had to play more than one role. Whenever this happens, the miner will have to perform the roles sequentially (our implementation does not parallelize these cases), which will slow down the average block time.

4.2 Energy Consumption

To compare the energy consumption of Etherspace and Ethereum, we estimate how much energy is needed to add a block to both blockchains. At the time of this writing, the current number of nodes in the Ethereum main network is 7 369 [8]. If Etherspace had the same amount of miners as the Ethereum network has nodes, each miner would call the hash function 198 times to generate all the proof-of-space challenge values for every block. (if we use the same number of iterations that we used to measure the quality distribution of the proof-of-space answers, each miner will have to generate 99 values. To generate each value, it will need to call the hash function 2 times. All miners, in total, will call the hash function 1 459 062 times for each block. At the time of the writing of this work, the proof-of-work difficulty in the Ethereum main network is 2 466 TH [9], meaning that a solution to the proof-of-work is found, on average, once every $2\ 466 * 10^{12}$ calls to the hash function. We can use this number as an estimate of the number of calls to the hash function that are required for adding a block to Ethereum blockchain. We can see that the number of calls to the hash function that is needed per block for Ethereum is several orders of magnitude higher than for Etherspace.

To have a rough idea of how those hash functions calls translate to energy, we will use as a reference an Nvidia Geforce GTX 1070 (not overclocked) that has a hash rate of 27 MH/s and a consumption of 135 W [14]. With this GPU, adding a block to the Ethereum blockchain would cost 12 329 999 999.99 Joule (J), while adding a block to the Etherspace blockchain would cost 7.30 J. As expected, the energy required to add a block to the Ethereum blockchain is several orders of magnitude higher than in Etherspace. We can conclude that Etherspace achieves its goal of being substantially more energy efficient than Ethereum.

4.3 Blockchain convergence

To analyze the convergence of the blockchain, we will count the number of times that miners switched active chains, i.e., the occurrence of forks. Note that when a miner adds a block to its active chain, it is not switching chain, it is extending its active chain. If the miner switches between two chains with the same number of blocks, where the only different block is the block with the highest height, we consider it a fork (i.e., a fork with a single block). We consider that the size of a fork is equal to the number of blocks that the miner pruned to switch between chains.

The number of forks can be seen in Figure 4.1. Note that the figure shows the total number of forks over 40 minutes and does not show the number of forks for each blockchain height. We can see that Etherspace has approximately ten times more forks of one block than Ethereum. This result is expected since in Etherspace, all miners create blocks for every blockchain height, while on Ethereum, only a limited (sometimes only one) miner create blocks for each height. Nevertheless, Etherspace had no forks bigger than one block, while, Ethereum had forks of two and three blocks. This is probably due to Etherspace having a better differentiation mechanism for blocks at the same height (i.e., the quality of the proof-of-space).

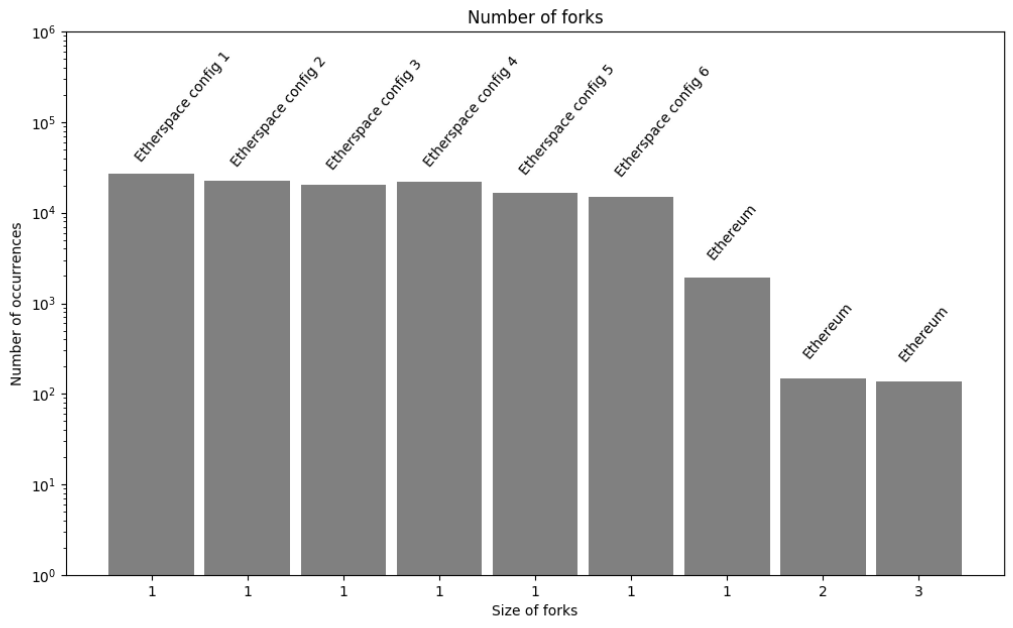


Figure 4.1: Size and number of occurences of forks during the course of the experiments.

4.4 Throughput

To evaluate the throughput of Etherspace and Ethereum, we must start by answering the question: How many blocks should we wait until we consider that a block is finalized? If we want to compare both models on a similar level, it is essential to use the same number of blocks for both of them. The time between blocks is similar in both models; if we give one of the models a larger margin, then the transaction confirmation time (TCT) of that model will always be better. According to Ethereum's creator [28] in a blockchain with an estimated block time of 17 seconds, ten confirmation blocks give a secure waiting margin, which corresponds to the number of blocks added in three minutes. For this evaluation, we decided to use a slightly larger margin, 12 blocks, which corresponds approximately to the number of blocks added in three minutes in the Ethereum experiment. Note that since the most extensive fork that we had observed was of 3 blocks, we could have used a smaller margin (i.e., four blocks).

In our test environment, we have a Python script that feeds transactions to the miners. Every time a miner receives a transaction, we store the timestamp at which the transaction entered the network (note that miners only store the timestamp when they receive the transaction from the script, if they receive it from another miner, they do not store the timestamp). Whenever a block stored in the chain of a miner gets 12 blocks on top of it, we store a timestamp and associate it with all the transactions of the block. These timestamps allow us to measure how much time it took for each transaction to be accepted in the blockchain - the transaction confirmation time (TCT). We can also consider the number of transactions that were finalized during the experiment, divide it by the time of the experiment (i.e., 40 minutes since we discard the first and last 10 minutes) and get an idea of how many transactions our system can handle, we measure the throughput in transactions per second (TPS).

Protocol	TCT (sec)	TPS (tx/sec)
Etherspace config 1	3 min 07	5.26
Etherspace config 2	3 min 50	4.46
Etherspace config 3	5 min 04	3.60
Etherspace config 4	3 min 47	4.44
Etherspace config 5	5 min 13	3.48
Etherspace config 6	4 min 52	3.67
Ethereum	3 min 06	4.11

Table 4.2: Transaction confirmation time and transactions per second of each protocol.

We can see in Table 4.2 both the TCT and TPS of each protocol. We can conclude that both Ethereum and Etherspace have similar throughputs. Moreover, in Figure 4.2, we can see the relation between the Etherspace block time and the number of transactions per second that the system can handle. This relation is probably caused by the significant ratio of committee members to miners in the network. A decrease of the ratio of committee members to miners will probably increase the throughput of Etherspace significantly.

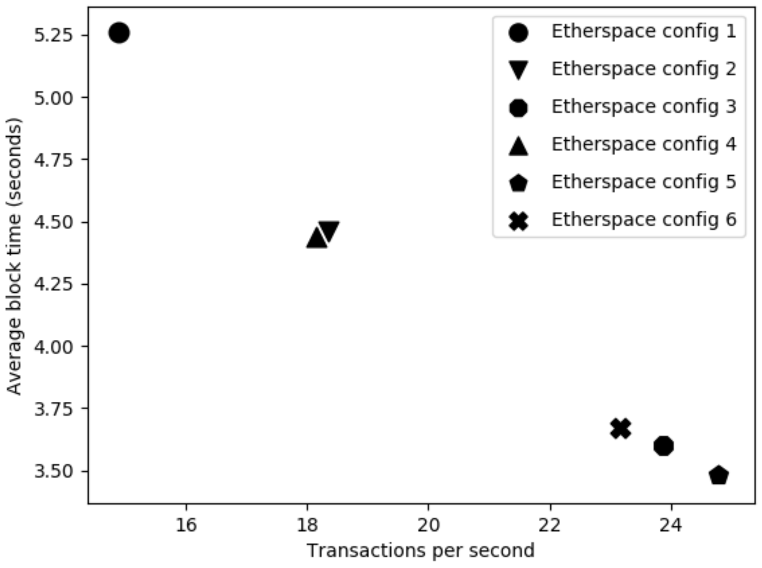


Figure 4.2: Relation between average block time and transactions per second

4.5 Discussion

In this chapter, we presented the evaluation of Etherspace and described the experimentation environment. Our main results are the following:

- The energy consumption of Etherspace is several orders of magnitude lower than Ethereum.
- Etherspace converges faster than Ethereum.
- The throughput of Etherspace is similar to Ethereum.

In this evaluation, we only considered honest miners. Measuring the fault tolerance and security of our protocol is an important task that we leave for future work. Moreover, some questions about this work remain open, and further research is needed. Nevertheless, this work shows promising results and Etherspace achieves its primary goal, it is an energy-efficient alternative to traditional blockchain protocols.

Chapter 5

Conclusions

Proof-of-work has been successful in maintaining the security of blockchain protocols while wasting vast amounts of energy. This work explored the usage of proof-of-space in blockchains. We analyzed the problems that come from replacing proof-of-space with proof-of-work, such as secure randomness generation, and proposed a novel approach that combines multiple techniques used in different blockchain models.

With this work, we were able to implement a practical proof-of-space blockchain on top of Ethereum, named Etherspace. Etherspace is an important milestone in making proof-of-space more than a promising theoretical alternative. Etherspace consumes considerably less energy than Ethereum. Each block in Etherspace requires ten orders of magnitude less energy to be generated than an Ethereum block. Besides, Etherspace has shorter forks than Ethereum and similar levels of throughput. We can conclude that Etherspace provides a promising blockchain model that can one day replace traditional proof-of-work blockchains and reduce the environmental impact of blockchain protocols.

A blockchain that uses proofs-of-space will consume less energy. Therefore, mining will also be cheaper, which will result in more small miners and, in turn, increase the decentralization of the protocol while providing a sustainable blockchain. A sustainable future demands green blockchains. The techniques and results of this thesis have been partially presented in the following peer-reviewed publication: Diogo Castilho, Paulo Silva, João Barreto, and Miguel Matos. Etherspace: uma abordagem proof-of-space na blockchain ethereum. In INForum 2019 - Atas do 11 o Simpósio de Informática, pages 193–204. NOVA.FCT Editorial, 2019. ISBN 978-972-8893-75-0. URL http://inforum.org.pt/INForum2019/docs/atas-do-inforum2019.

5.1 Future Work

This work focused on exploring proof-of-space in blockchain protocols as a replacement for proof-of-work. While the results look very promising, several research and implementation questions remain open.

The approach that we explored needs to be tested with dishonest nodes that use different strategies, such as trying to double-spend transactions or meddling with the Practical Hound protocol. It is crucial to measure the resistance of the system against attacks and the impact on performance.

We also need to study further the possible vulnerabilities of this protocol. Furthermore, we need to implement mitigations to the vulnerabilities mentioned in Section 3.5.

Recently, Chia released a document with more details about their proof-of-space construction [16]. This document will have to be analyzed to find possible improvements for our proof-of-space implementation.

The implementation of the Practical Hound protocol has to be completed and possibly improved. We used a naive approach to the broadcast of Practical Hound messages, which definitely can be improved. A better implementation of Practical Hound will lead to faster block times. Moreover, parameters used in the Practical Hound protocol will have to be further studied, such as the size of the committee.

Bibliography

[1] Bitcoin energy consumption index. https://digiconomist.net/bitcoin-energy-consumption. Accessed: 17-11-2018.

[2] Hyperledger sawtooth. https://sawtooth.hyperledger.org. Accessed: 29-11-2018.

[3] Bitshares. https://bitshares.org. Accessed: 4-12-2018.

[4] Burstcoin. https://www.burst-coin.org/. Accessed on 2-01-2019.

[5] Chia network. https://chia.net. Accessed: 16-11-2018.

[6] Eos.io. https://eos.io. Accessed: 4-12-2018.

[7] Ethereum white paper. https://github.com/ethereum/wiki/wiki/White-Paper, . Accessed: 28-11-2018.

[8] Ethernodes. https://ethernodes.org, . Accessed: 23-10-2019.

[9] Etherscan - ethereum blockchain explorer. https://etherscan.io/, . Accessed: 23-10-2019.

[10] Fastcoin. https://www.fastcoin.ca/. Accessed on 20-12-2018.

[11] Intel software guard extensions (intel sgx). https://software.intel.com/en-us/sgx. Accessed: 25-10-2019.

[12] Iota. https://www.iota.org/. Accessed: 30-11-2018.

[13] Litecoin. https://litecoin.com/. Accessed on 20-12-2018.

[14] Geforce gtx 1070 ethereum mining. https://www.legitreviews.com/geforce-gtx-1070-ethereum-mining-small-tweaks-great-hashrate-low-power_195451. Accessed: 23-10-2019.

[15] Steemit. https://steemit.com. Accessed: 4-12-2018.

[16] Chia proof of space construction. `https://www.chia.net/assets/proof_of_space.pdf`, 2019.

[17] Ittai Abraham, Dahlia Malkhi, Kartik Nayak, Ling Ren, and Alexander Spiegelman. Solida: A Blockchain Protocol Based on Reconfigurable Byzantine Consensus. In James Aspnes, Alysson Bessani, Pascal Felber, and João Leitão, editors, *21st International Conference on Principles of Distributed Systems (OPODIS 2017)*, volume 95 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 25:1–25:19, Dagstuhl, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik. ISBN 978-3-95977-061-3. doi: 10.4230/LIPIcs.OPODIS.2017.25. URL `http://drops.dagstuhl.de/opus/volltexte/2018/8640`.

[18] Hamza Abusalah, Joël Alwen, Bram Cohen, Danylo Khilko, Krzysztof Pietrzak, and Leonid Reyzin. Beyond hellman's time-memory trade-offs with applications to proofs of space. Cryptology ePrint Archive, Report 2017/893, 2017. `https://eprint.iacr.org/2017/893`.

[19] Elli Androulaki, Artem Barger, Vita Bortnikov, Christian Cachin, Konstantinos Christidis, Angelo De Caro, David Enyeart, Christopher Ferris, Gennady Laventman, Yacov Manevich, Srinivasan Muralidharan, Chet Murthy, Binh Nguyen, Manish Sethi, Gari Singh, Keith Smith, Alessandro Sorniotti, Chrysoula Stathakopoulou, Marko Vukolic, Sharon Weed Cocco, and Jason Yellick. Hyperledger fabric: A distributed operating system for permissioned blockchains. *CoRR*, abs/1801.10228, 2018. URL `http://arxiv.org/abs/1801.10228`.

[20] Frederik Armknecht, Ghassan Karame, Avikarsha Mandal, Franck Youssef, and Erik Zenner. Ripple: Overview and outlook. volume 9229, 08 2015. doi: 10.1007/978-3-319-22846-4_10.

[21] Adam Back. Hashcash - a denial of service counter-measure. Technical report, 2002.

[22] Iddo Bentov, Charles Lee, Alex Mizrahi, and Meni Rosenfeld. Proof of activity: Extending bitcoin's proof of work via proof of stake. Cryptology ePrint Archive, Report 2014/452, 2014. `https://eprint.iacr.org/2014/452`.

[23] Iddo Bentov, Ariel Gabizon, and Alex Mizrahi. Cryptocurrencies without proof of work. In Jeremy Clark, Sarah Meiklejohn, Peter Y.A. Ryan, Dan Wallach, Michael Brenner, and Kurt Rohloff, editors, *Financial Cryptography and Data Security*, pages 142–157, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg. ISBN 978-3-662-53357-4.

[24] Manuel Blum. Coin flipping by telephone a protocol for solving impossible problems. *SIGACT News*, 15(1):23–27, January 1983. ISSN 0163-5700. doi: 10.1145/1008908.1008911. URL `http://doi.acm.org/10.1145/1008908.1008911`.

[25] Dan Boneh, Benedikt Bünz, and Ben Fisch. A survey of two verifiable delay functions. Cryptology ePrint Archive, Report 2018/712, 2018. `https://eprint.iacr.org/2018/712`.

[26] Joseph Bonneau, Andrew Miller, Jeremy Clark, Arvind Narayanan, Joshua A. Kroll, and Edward W. Felten. Sok: Research perspectives and challenges for bitcoin and cryptocurrencies. 2015:104–121, 07 2015. doi: 10.1109/SP.2015.14.

[27] Marcus Brandenburger, Christian Cachin, Matthias Lorenz, and Rüdiger Kapitza. Rollback and forking detection for trusted execution environments using lightweight collective memory. *CoRR*, abs/1701.00981, 2017. URL `http://arxiv.org/abs/1701.00981`.

[28] Vitalik Buterin. On slow and fast block times. https://blog.ethereum.org/2015/09/14/on-slow-and-fast-block-times/, . Accessed: 17-10-2019.

[29] Vitalik Buterin. Slasher: A punitive proof-of-stake algorithm. https://blog.ethereum.org/2014/01/15/slasher-a-punitive-proof-of-stake-algorithm/, . Accessed: 28-11-2018.

[30] Vitalik Buterin and Virgil Griffith. Casper the friendly finality gadget. *CoRR*, abs/1710.09437, 2017. URL `http://arxiv.org/abs/1710.09437`.

[31] Christian Cachin and Marko Vukolic. Blockchain consensus protocols in the wild. *CoRR*, abs/1707.01873, 2017. URL `http://arxiv.org/abs/1707.01873`.

[32] Miguel Castro and Barbara Liskov. Practical byzantine fault tolerance. In *Proceedings of the Third Symposium on Operating Systems Design and Implementation*, OSDI '99, pages 173–186, Berkeley, CA, USA, 1999. USENIX Association. ISBN 1-880446-39-1. URL `http://dl.acm.org/citation.cfm?id=296806.296824`.

[33] David Chaum. Blind signatures for untraceable payments. In David Chaum, Ronald L. Rivest, and Alan T. Sherman, editors, *Advances in Cryptology*, pages 199–203, Boston, MA, 1983. Springer US. ISBN 978-1-4757-0602-4.

[34] Lin Chen, Lei Xu, Nolan Shah, Zhimin Gao, Yang Lu, and Weidong Shi. On security analysis of proof-of-elapsed-time (poet). In Paul Spirakis and Philippas Tsigas, editors, *Stabilization, Safety, and Security of Distributed Systems*, pages 282–297, Cham, 2017. Springer International Publishing. ISBN 978-3-319-69084-1.

[35] Alexander Chepurnoy, Tuyet Duong, Lei Fan, and Hong-Sheng Zhou. Twinscoin: A cryptocurrency via proof-of-work and proof-of-stake. Cryptology ePrint Archive, Report 2017/232, 2017. `https://eprint.iacr.org/2017/232`.

[36] Bram Cohen. Incentives build robustness in bittorrent, 2003.

[37] Bram Cohen, , and Krzysztof Pietrzak. The chia network blockchain. hhttps://www.chia.net/assets/ChiaGreenPaper.pdf, 2019.

[38] Wei Dai. b-money. http://www.weidai.com/bmoney.txt, 1998.

[39] Phil Daian, Rafael Pass, and Elaine Shi. Snow white: Provably secure proofs of stake. Cryptology ePrint Archive, Report 2016/919, 2016. https://eprint.iacr.org/2016/919.

[40] Benedikt Bünz Dan Boneh, Joseph Bonneau and Ben Fisch. Verifiable delay functions. Cryptology ePrint Archive, Report 2018/601, 2018. https://eprint.iacr.org/2018/601.

[41] Helga Danova. Ghash.io pool is closed. https://blog.cex.io/news/announcement-ghash-io-pool-closing-15878. Accessed: 22-11-2018.

[42] Arthur Britto David Schwartz, Noah Youngs. The ripple protocol consensus algorithm. https://ripple.com/files/ripple_consensus_whitepaper.pdf.

[43] John R. Douceur. The sybil attack. In *Revised Papers from the First International Workshop on Peer-to-Peer Systems*, IPTPS '01, pages 251–260, London, UK, UK, 2002. Springer-Verlag. ISBN 3-540-44179-4. URL http://dl.acm.org/citation.cfm?id=646334.687813.

[44] Tuyet Duong, Lei Fan, and Hong-Sheng Zhou. 2-hop blockchain: Combining proof-of-work and proof-of-stake securely. Cryptology ePrint Archive, Report 2016/716, 2016. https://eprint.iacr.org/2016/716.

[45] Cynthia Dwork and Moni Naor. Pricing via processing or combatting junk mail. In *Proceedings of the 12th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '92, pages 139–147, London, UK, UK, 1993. Springer-Verlag. ISBN 3-540-57340-2. URL http://dl.acm.org/citation.cfm?id=646757.705669.

[46] Stefan Dziembowski, Sebastian Faust, Vladimir Kolmogorov, and Krzysztof Pietrzak. Proofs of space. Cryptology ePrint Archive, Report 2013/796, 2013. https://eprint.iacr.org/2013/796.

[47] Thaddeus Dryja Ethan Heilman, Neha Narula and Madars Virza. Iota vulnerability report: Cryptanalysis of the curl hash function enabling practical signature forgery attacks on the iota cryptocurrency. https://github.com/mit-dci/tangled-curl/blob/master/vuln-iota.md. Accessed: 30-11-2018.

[48] Ittay Eyal, Adem Efe Gencer, Emin Gün Sirer, and Robbert van Renesse. Bitcoin-ng: A scalable blockchain protocol. *CoRR*, abs/1510.02037, 2015. URL `http://arxiv.org/abs/1510.02037`.

[49] Amos Fiat and Moni Naor. Rigorous time/space trade-offs for inverting functions. *SIAM J. Comput.*, 29(3):790–803, December 1999. ISSN 0097-5397. doi: 10.1137/S0097539795280512. URL `https://doi.org/10.1137/S0097539795280512`.

[50] Michael J. Fischer, Nancy A. Lynch, and Michael S. Paterson. Impossibility of distributed consensus with one faulty process, 1985.

[51] Juan Garay, Aggelos Kiayias, and Nikos Leonardos. The bitcoin backbone protocol: Analysis and applications. Cryptology ePrint Archive, Report 2014/765, 2014. `https://eprint.iacr.org/2014/765`.

[52] 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*, SOSP '17, pages 51–68, New York, NY, USA, 2017. ACM. ISBN 978-1-4503-5085-3. doi: 10.1145/3132747.3132757. URL `http://doi.acm.org/10.1145/3132747.3132757`.

[53] Johannes Götzfried, Moritz Eckert, Sebastian Schinzel, and Tilo Müller. Cache attacks on intel sgx. In *Proceedings of the 10th European Workshop on Systems Security*, EuroSec'17, pages 2:1–2:6, New York, NY, USA, 2017. ACM. ISBN 978-1-4503-4935-2. doi: 10.1145/3065913.3065915. URL `http://doi.acm.org/10.1145/3065913.3065915`.

[54] M. Hellman. A cryptanalytic time-memory trade-off. *IEEE Transactions on Information Theory*, 26(4):401–406, July 1980. ISSN 0018-9448. doi: 10.1109/TIT.1980.1056220.

[55] Aggelos Kiayias and Giorgos Panagiotakos. Speed-security tradeoffs in blockchain protocols. Cryptology ePrint Archive, Report 2015/1019, 2015. `https://eprint.iacr.org/2015/1019`.

[56] Aggelos Kiayias, Alexander Russell, Bernardo David, and Roman Oliynykov. Ouroboros: A provably secure proof-of-stake blockchain protocol. Cryptology ePrint Archive, Report 2016/889, 2016. `https://eprint.iacr.org/2016/889`.

[57] Eleftherios Kokoris Kogias, Philipp Jovanovic, Nicolas Gailly, Ismail Khoffi, Linus Gasser, and Bryan Ford. Enhancing bitcoin security and performance with strong consistency via collective signing. In *25th USENIX Security Symposium (USENIX Security*

16), pages 279–296, Austin, TX, 2016. USENIX Association. ISBN 978-1-931971-32-4. URL `https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/kogias`.

[58] Leslie Lamport, Robert Shostak, and Marshall Pease. The byzantine generals problem. *ACM Trans. Program. Lang. Syst.*, 4(3):382–401, July 1982. ISSN 0164-0925. doi: 10.1145/357172.357176. URL `http://doi.acm.org/10.1145/357172.357176`.

[59] Loi Luu, Yaron Velner, Jason Teutsch, and Prateek Saxena. Smartpool: Practical decentralized pooled mining. In *26th USENIX Security Symposium (USENIX Security 17)*, pages 1409–1426, Vancouver, BC, 2017. USENIX Association. ISBN 978-1-931971-40-9. URL `https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/luu`.

[60] David Malone and K.J. O'Dwyer. Bitcoin mining and its energy footprint. pages 280–285, 01 2014. doi: 10.1049/cp.2014.0699.

[61] Jon Matonis. The bitcoin mining arms race: Ghash.io and the 51 https://www.coindesk.com/bitcoin-mining-detente-ghash-io-51-issue. Accessed: 15-11-2018.

[62] David Mazières. The stellar consensus protocol: A federated model for internet-level consensus, 2015.

[63] Silvio Micali, Michael Rabin, and Salil Vadhan. Verifiable random functions. In *Proceedings of the 40th Annual Symposium on the Foundations of Computer Science*, pages 120–130, New York, NY, October 1999. IEEE.

[64] Andrew Miller, Ahmed Kosba, Jonathan Katz, and Elaine Shi. Nonoutsourceable scratch-off puzzles to discourage bitcoin mining coalitions. In *Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security*, CCS '15, pages 680–691, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3832-5. doi: 10.1145/2810103.2813621. URL `http://doi.acm.org/10.1145/2810103.2813621`.

[65] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system," http://bitcoin.org/bitcoin.pdf, 2008.

[66] P4Titan. Slimcoin a peer-to-peer crypto-currency with proof-of-burn "mining without powerful hardware". https://github.com/slimcoin-project/slimcoin-project.github.io/raw/master/whitepaperSLM.pdf. 17 May 2014.

[67] Sunoo Park, Albert Kwon, Georg Fuchsbauer, Peter Gaži, Joël Alwen, and Krzysztof Pietrzak. Spacemint: A cryptocurrency based on proofs of space. Cryptology ePrint Archive, Report 2015/528, 2015. `https://eprint.iacr.org/2015/528`.

[68] Rafael Pass and Elaine Shi. Hybrid consensus: Efficient consensus in the permissionless model. Cryptology ePrint Archive, Report 2016/917, 2016. `https://eprint.iacr.org/2016/917`.

[69] Rafael Pass, Lior Seeman, and abhi shelat. Analysis of the blockchain protocol in asynchronous networks. Cryptology ePrint Archive, Report 2016/454, 2016. `https://eprint.iacr.org/2016/454`.

[70] Krzysztof Pietrzak. Simple verifiable delay functions. Cryptology ePrint Archive, Report 2018/627, 2018. `https://eprint.iacr.org/2018/627`.

[71] Andrew Poelstra. On stake and consensus, . March 2015.

[72] Andrew Poelstra. Distributed consensus from proof of stake is impossible. https://download.wpsoftware.net/bitcoin/old-pos.pdf, . May, 2014.

[73] Serguei Popov. The tangle. `https://iota.org/IOTA_Whitepaper.pdf`.

[74] Meni Rosenfeld. Analysis of bitcoin pooled mining reward systems. *arXiv preprint arXiv*, 1112, 12 2011.

[75] Berry Schoenmakers. A simple publicly verifiable secret sharing scheme and its application to electronic. In *Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '99, pages 148–164, Berlin, Heidelberg, 1999. Springer-Verlag. ISBN 3-540-66347-9. URL `http://dl.acm.org/citation.cfm?id=646764.703956`.

[76] Michael Schwarz, Samuel Weiser, Daniel Gruss, Clémentine Maurice, and Stefan Mangard. Malware guard extension: Using SGX to conceal cache attacks. *CoRR*, abs/1702.08719, 2017. URL `http://arxiv.org/abs/1702.08719`.

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

[78] Scott Nadal Sunny King. Ppcoin: Peer-to-peer crypto-currency with proof-of-stake. 2012. URL `http://www.peercoin.net/`.

[79] E. Syta, I. Tamas, D. Visher, D. I. Wolinsky, P. Jovanovic, L. Gasser, N. Gailly, I. Khoffi, and B. Ford. Keeping authorities "honest or bust" with decentralized witness cosigning. In *2016 IEEE Symposium on Security and Privacy (SP)*, pages 526–545, May 2016. doi: 10.1109/SP.2016.38.

[80] Ewa Syta, Philipp Jovanovic, Eleftherios Kokoris Kogias, Nicolas Gailly, Linus Gasser, Ismail Khoffi, Michael J. Fischer, and Bryan Ford. Scalable bias-resistant distributed randomness. Cryptology ePrint Archive, Report 2016/1067, 2016. `https://eprint.iacr.org/2016/1067`.

[81] Michael Bedford Taylor. Bitcoin and the age of bespoke silicon. In *Proceedings of the 2013 International Conference on Compilers, Architectures and Synthesis for Embedded Systems*, CASES '13, pages 16:1–16:10, Piscataway, NJ, USA, 2013. IEEE Press. ISBN 978-1-4799-1400-5. URL `http://dl.acm.org/citation.cfm?id=2555729.2555745`.

[82] Marko Vukolić. The quest for scalable blockchain fabric: Proof-of-work vs. bft replication. pages 112–125, 05 2016. doi: 10.1007/978-3-319-39028-4_9.

[83] Benjamin Wesolowski. Efficient verifiable delay functions. Cryptology ePrint Archive, Report 2018/623, 2018. `https://eprint.iacr.org/2018/623`.

[84] Vlad Zamfir. Casper the friendly ghost a "correct-by-construction" blockchain consensus protocol draft v0.1.

[85] User "babubekkel21". Dpos consensus algorithm - missing white paper. https://steemit.com/dpos/@babubekkel21/dpos-consensus-algorithm-missing-white-paper. Accessed: 4-12-2018.

[86] User "QuantumMechanic". Proof of stake instead of proof of work. https://bitcointalk.org/index.php?topic=27787.0. Accessed: 20-11-2018.