Sustainability in mining protocols for public blockchains

Cecilie Daae Nilsen

Supervisor: Håvard Raddum

Department of Informatics
University of Bergen

March 2019

Acknowledgements

I would like to thank my family and friends for their support and encouragement. Also, I would like to thank my part-time employer for being flexible and giving me the chance to work on new areas related to my education. A special thanks to my supervisor for always having the time to discuss, give input and explanations when needed.

Abstract

Blockchains are a somewhat new technology with much potential and it is meeting growing public interest. However, with wider use of the technology we face the challenge of sustainability since the current mining protocols use a considerable amount of electricity in the process of creating new blocks. In practice some blockchains require specialized hardware designed for the sole purpose of mining blocks. Several new mining protocols are proposed to deal with this issue. This thesis will start by studying the current mining protocols of Bitcoin and Ethereum. Next, we discuss the different proposals for new protocols with regards to which are most suited for wider, long term use.

Table of contents

Chapter 1

Introduction

IT and the Internet is a big part of most peoples' everyday life, both in our work relations and in our private life. Most of us use web sites and web applications several times a day, either on our computer, on our tablet or on our smart phone. With the widespread use of the Internet, security should be something of high priority. Computer security is what enables us to use different web applications and a big part of computer security is cryptography, which makes communication on an unsecured communication line secure against eavesdropping or alterations. Before the Internet, cryptography was confined to the military and governments and was synonymous with encrypting and decrypting messages.

Early packet-switching networks from the 1960s and the 1970s between different universities led to the World Wide Web (WWW). Before this, circuit-switching networks like the analog telephone network was the standard. The WWW is an information space and is accessed via the Internet, a networking structure, which has since the mid-1990s had an enormous impact on our everyday life and on our culture. The Advanced Research Projects Agency Network (ARPANET) was the first packet-switching network that used the Internet Protocol (TCP/IP). This network consisted of four nodes: University of California L.A, Stanford Research Institute, University of California Santa Barbara and the University of Utah. More nodes attached to the network and by 1973 it expanded outside the USA to include Norwegian Seismic Array (NORSAR) via satellite. From here it expanded further to England and the rest of Europe [2].

Other such early networks include NPL network, Merit Network, CYCLADES and Telenet which used several different communication protocols. TCP/IP became the standard protocol for networking and some commercial Internet service providers (ISPs) emerged in the late 1980s and in the early 1990s [38]. The first web browser was published in August 1991 and was used to access WWW via Hypertext Transfer Protocol (HTTP) requests and

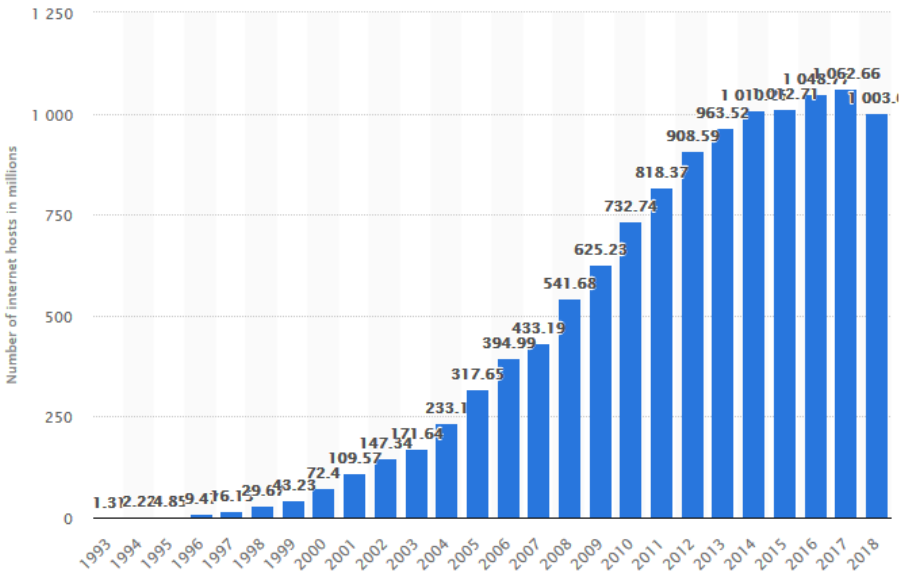


Fig. 1.1 Number of hosts in millions on the Internet from 1993 to 2018 [39].

responses. HTTP is still used today. Since the mid-1990s the use and impact of the Internet has grown substantially.

The Domain Name System (DNS) is essential for the Internet to function. It provides translations from IP addresses to human-readable addresses and vice versa to locate and identify servers. In Figure 1.1 we can see the growing number of Internet hosts in DNS from 1993 to 2018.

Cryptography is what makes us being able to access our bank online and use a variety of different web applications today. With all new applications on the WWW, new ways to use cryptography has emerged to provide security for data in transit and at rest.

As we can see in Figure 1.2, the Network model has five layers where the application layer on is at the top. Cryptography is usually implemented at the application layer or the transport layer (HTTPS). However, cryptography can also be found at the network layer (IPSec) and the datalink layer (WPA2).

Cryptography delivers several security services, like confidentiality, authentication, privacy, non-repudiation and integrity. Commonly used cryptographic primitives include:

- Cryptographic hash functions. These are one-way functions that map data of arbitrary size and produce an output string of a fixed size. These functions are infeasible to invert.

Layer #	Layer Name	Protocol	Protocol Data Unit	Addressing
5	Application	HTTP, SMTP, etc...	Messages	n/a
4	Transport	TCP/UDP	Segments/ Datagrams	Port #s
3	Network or Internet	IP	Packets	IP Address
2	Data Link	Ethernet, Wi-Fi	Frames	MAC Address
1	Physical	10 Base T, 802.11	Bits	n/a

Fig. 1.2 Layers in the Network model together with their protocols [56].

- Symmetric key cryptography. The same key is shared by both parties and used for both encryption and decryption.
- Asymmetric key cryptography/Public key cryptography. Here there are two keys, one is public and used for encryption of the message to send and the other is private and used to decrypt the message that is received.
- Digital signatures. Verifies that the message came from said sender by using public key cryptograhy and cryptographic hash functions. A code made by the sender's private key is attached to the message and acts like a signature to provide authenticity.
- Cryptographically secure pseudo random number generator (CSPRNG). To generate random numbers for key generation, nonces and salt.
- Message authentication code (MAC). Confirms that the message came from said sender and that it is not tampered with by using symmetric cryptography and cryptographic hash functions.

To have a proper function these primitives must be combined in a security protocol [22]. An example of such a protocol is Transport Layer Security (TLS) where several of these primitives are used to provide a reliable, private connection over a network where the parties are authenticated. Message integrity in TLS is achieved by using MACs for each message. Confidentiality is secured by symmetric cryptography of the messages and authentication is provided by using public key cryptography, see Figure 1.3. By using TLS we protect ourselves from eavesdropping and man-in-the middle attacks on the Internet by using Hypertext Transfer Protocol Secure (HTTPS) protocol instead of HTTP.

Since the widespread use of the Internet we have seen a growing use of cryptography to secure the communication. The last fifteen years online banking, secure storing of passwords

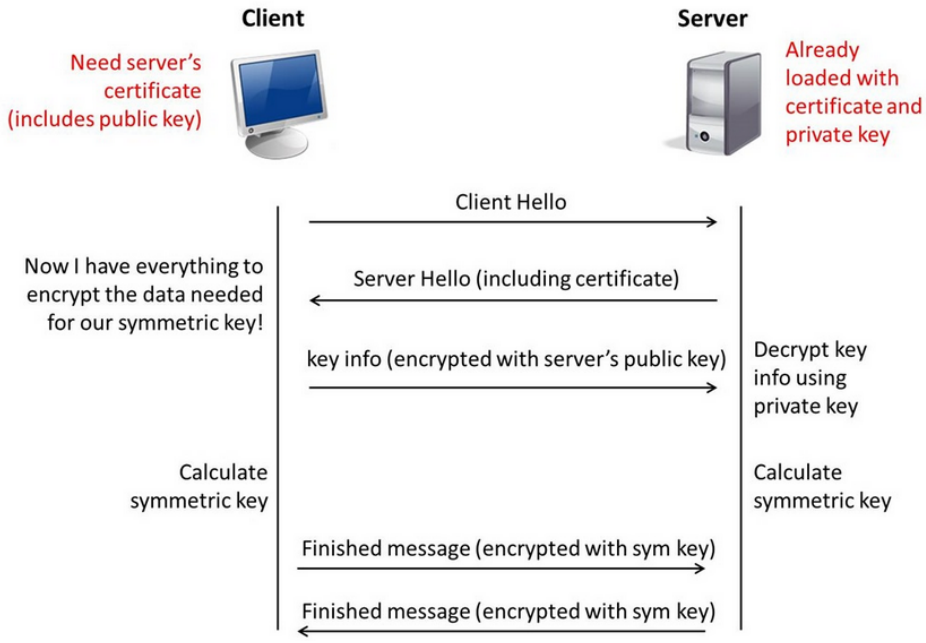


Fig. 1.3 TLS handshake to establish secure communication [57].

and digital signing of documents has become the new norm because of the security that cryptography provides. Online banking uses TLS for a secure communication channel and digital signatures to authenticate the correct user of the bank. To store passwords securely, the password is hashed together with a randomly generated value called salt. This prevents the attackers from retrieving the passwords if there should be a data leak. Storing passwords is something that very many web applications do so that users can log into their account. Examples of such web application are web shops, customer accounts of mobile phone companies and your private e-mail. Practically everything that you as a user has a customer relation to should store passwords securely if there exist a customer account that users can log into.

Signing documents digitally is the equivalent of signing a document by hand and is just as binding as a hand-written signature. In Norway, BankID is a way of authenticating your identity and signing such documents. BankID has different methods for authenticating. One of them identify the user by birth number, a personal password and a single-use code generated from a hardware token. Another method to authenticate your BankID uses the mobile phone. BankID is used in many scenarios, but most commonly used to log into your online bank, verify payments online and to sign documents.

One of the newest technologies to use cryptography in yet another way are blockchains. Blockchains mainly use two different cryptographic tools: digital signatures and hash functions.

The blockchain concept was first introduced in 2009 and today there are many different blockchains in the world. The blockchains in focus for the purpose of this thesis will be the public blockchains Bitcoin and Ethereum. Bitcoin is the pioneer blockchain and its purpose is to be a decentralized cryptocurrency, whereas Ethereum improves upon the concept by using the blockchain for more than just cryptocurrency; it serves as a general application platform and represents a decentralized internet paradigm. This is done by using programs called 'Smart Contracts' that are, together with their functions, treated as transactions in the chain. Introducing such programs in the chain makes the chain programmable and users can program whatever they like.

The mining protocol decides which node in the blockchain network gets to mine the next block that is added to the chain. It is this protocol that makes the system decentralized since in such a system there is no central entity to trust and in a system involving currency the order of transactions is very important. Even though this system is decentralized, it is made secure by mining in the sense that the transactions in the blockchain are correct. The security in the mining process is based on cryptography. The various new types of 'money' that exists in many blockchains are therefore called cryptocurrencies.

1.1 Motivation

The motivation for this thesis is the growing interest and use of blockchains. The mining protocol used in both Bitcoin and Ethereum today may not be sustainable in the long run because of their high electricity use. As of today Ethereum is working on a transition to another protocol called Proof of Stake (PoS) by first implementing a hybrid protocol called 'Casper'. We will have a look at the protocols used in both chains today of the type called Proof of Work (PoW) and also go into PoS, Casper and other alternatives that are proposed as new mining protocols.

I do believe that blockchains will keep increasing in popularity, not only because of its cryptographic security, but mostly because of its decentralized view of the internet. The value of cryptocurrency has so far been highly fluctuating, so perhaps it is better to view it as an asset rather than a currency. In the case of Ethereum, my own opinion is that the coin Ether can be viewed as a 'ticket' you can exchange for different uses within the application platform. However, to keep the interest for the technology growing and thereby reach its full potential, maybe for purposes and applications that are not yet thought of, the challenge of

electricity waste has to be dealt with. Another issue with some mining protocols is that they pull the system in a more centralized direction. This is a contradiction to a block chain's intention which is to replace the use of servers and therefore also centralized entities. Even if blockchains' popularity should decrease, the idea about a decentralized Internet can be built upon and it could be a stepping stone to some other technology.

In this thesis we will first go into details of how the mining in Bitcoin and Ethereum works, and then discuss which of the more recently proposed mining protocols that are most likely to be sustainable over time with more global usage of block chains. A note about the reference list: Practically all information regarding blockchains and mining exists only as white papers and various web pages, and is not published as traditional academic journal or conference papers. Hence the reference list consists almost only of web pages pointing to where information has been found.

Chapter 2

Blockchains

The idea that resulted in the blockchain concept was to make a decentralized cryptocurrency. There have been multiple previous attempts at creating electronic cash, but decentralization and the double spending problem proved to be a challenge. The first to succeed in implementing a truly decentralized system is an unknown person, or group of people, that uses the pseudonym "Satoshi Nakamoto". This entity introduced Bitcoin in 2009. Since its original intention was to create and maintain a currency with its transactions, the ability to create coins and to send them to others were the requirements of the system.

2.1 Blockchain concept

Previous attempts to create a cryptocurrency did not solve the problem with decentralization as they all relied on a central server to ensure trust. In a cryptocurrency system the *order* of the transactions is of high importance and the participants have to agree on the same order of transactions. Satoshi Nakamoto's solution to this was to gather transactions with timestamps

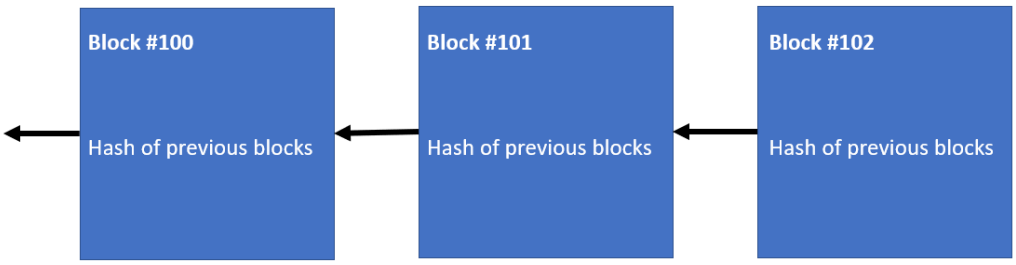


Fig. 2.1 Simplified example of the block chain concept. We see that each block is dependent on all the blocks that came before.

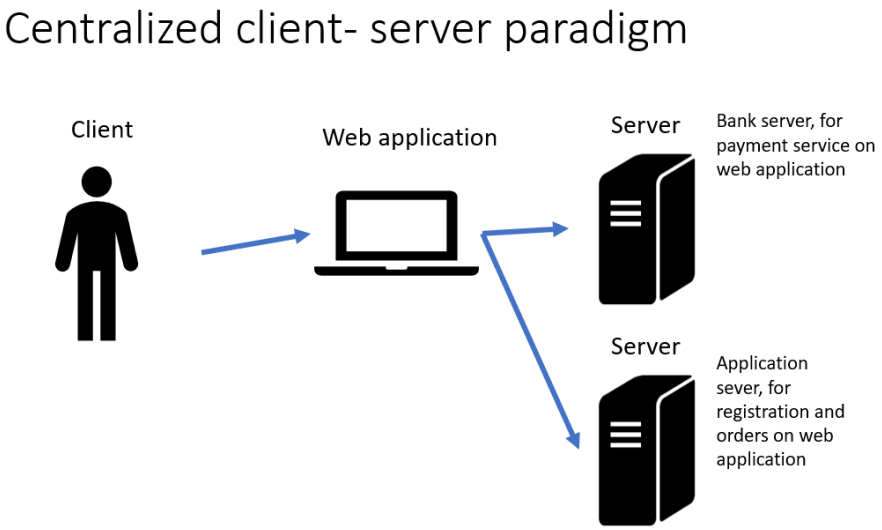


Fig. 2.2 The usual client-server paradigm.

into a block and making a chain of the blocks, where each block is dependent on all the previous blocks [34]. To prevent double spending of coins there also has to be no possibility to rewrite the history of the transactions, i.e the blocks in Bitcoin. This is ensured by use of a PoW protocol which we will have a closer look at later. The blockchain concept is illustrated in Figure 2.1.

2.2 Extension of concept

The idea to build applications on top of cryptocurrency blockchains emerged, i.e applications on top of a blockchain, to use the technology for more than just creating and sending coins [34]. There were a few ways of doing this. The first was to build a new blockchain for each application, which is a very limited way of doing things as each would have its own currency and would only work for that specific application. It would also be expensive in terms of developing costs. Another way is to build applications wrapped in ordinary transactions in the existing Bitcoin blockchain. This second solution also has its flaws because Bitcoin can not decide whether a foreign transaction is valid or not, since it does not keep track of other states than that of its own coins.

Ethereum's purpose is to combine these methods and make a 'general' block chain. Here users can create what has become known as *smart contracts* with their own rules. A smart contract is handled like any other transaction such as transferring coins. With Ethereum we

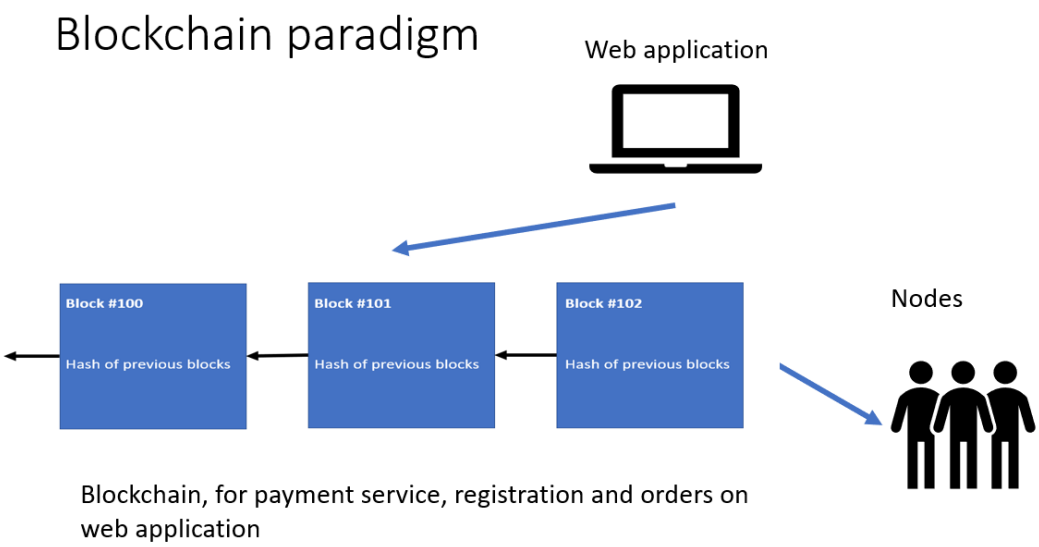


Fig. 2.3 New decentralized paradigm. The main difference is that the users are part of both payment, registration, orders and other operations. It also provides 'trust' between unknown people, so that a trusted third party is not needed. The latter also eliminates any payment to third parties.

get a new internet paradigm. Figure 2.2 illustrates the usual client-server paradigm while Figure 2.3 illustrates the new blockchain paradigm.

Ethereum can also be used without a web application, e.g making a smart contract between someone who may or may not know each other and want to enforce some sort of agreement. In this case, there is no need for a web application, but the contract is still on the blockchain. We will discuss more about use cases in Chapter 4.

Private blockchains

There are also private blockchains, where all nodes are known and trusted. These kinds of blockchains can be used within a company and serve as a distributed database that can not be tampered with. Here, there are no web applications on top and no currency is generated. There are several different ways to implement a private blockchain, some of them do not include mining and therefore do not have a mining protocol. Others may include mining and hence also a mining protocol where mining new blocks is efficient since all nodes trust each other. Participants in this scenario are given different permissions. This thesis has focus on decentralization, and private block chains are therefore not discussed further.

2.3 Blockchain security

Blockchains are considered secure for several reasons. Let us look at the features that prevent certain attacks from being possible to carry out.

Cryptography attacks

These kinds of attacks on the system's data are prevented by the use of cryptographic hash functions and digital signatures. The hash functions and signature algorithms have been through thorough analysis by cryptologists and have very conservative security estimates. It is therefore assumed that these functions are secure as they are based on well known, difficult mathematical problems. The digital signature provides a secure way to identify every user of the blockchain.

51 % attack

This attack is related to double spending of coins where one tries to convince the rest of the network that some wrong order of the transactions is correct by controlling the majority of the network. The attacker can also prevent other transactions from being processed, both of which are more likely to succeed if one controls the majority of the mining power of the network. Such an attack could also be carried out with less than the majority, but is much less likely to succeed [1].

Let us assume that an attacker has less than 50 % of the mining power of the network and tries to spend the same coins that have already been spent in a transaction in a previous block. He then has to make a fork of the chain, see illustration in Figure 2.4. A fork is a new separate chain made out of a block that is not the last in the chain. This means that he/she can not simply put another block into the original chain with data that is not correct; this would not be accepted by the other nodes.

The other nodes verify the state after processing each transaction in each block, so the other nodes would see that the coins already had been spent before. Therefore, the attacker has to fork the chain and make this the longest chain since the longest chain is taken to be the true chain by the network. In this way, the attacker can alter the state by mining several blocks so that the false transactions are verified by the others. To have the longest chain, the attacker has to have the majority of the mining power to produce valid blocks quicker than the rest of the network. The same coins would in this scenario first be spent, then reverted, and finally sent back to himself.

However, the mining protocols prevents this kind of attack as follows. In the current PoW protocol:

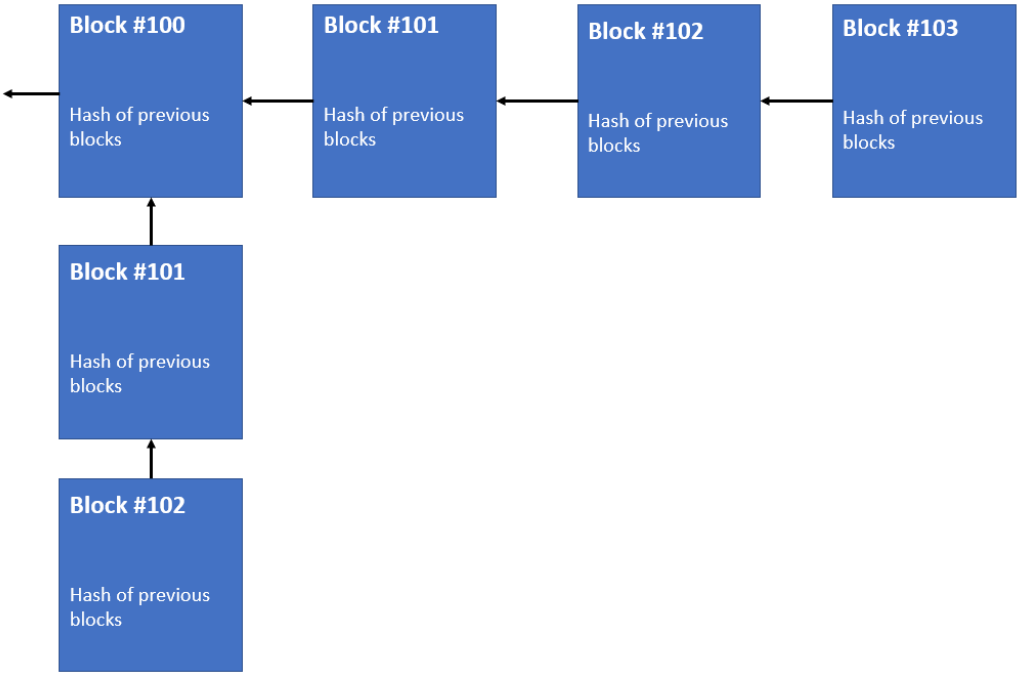


Fig. 2.4 Illustration of a fork (downwards). Block number 100 is the same as the original, but block number 101 and block number 102 are different blocks with different transactions. These blocks will have different hashes than the original ones so they form a completely different chain.

1. The longest chain in case of a fork is taken as the true chain by the network because this has the most computational work combined to back it up.

2. To mine a new block, the miner has to put in a lot of computational work. This implies that the attacker wouldn't be able to make a longer chain than the rest of the network.

3. One has to solve the computational 'puzzle' which is to find a hash value that is below a given target. This is done by trial and error and as this is impossible to guess without doing the brute force work, it is computationally very heavy. However, it is very easy to verify that a given hash value is indeed correct. This prevents succeeding in creating a fork by not having to do more work than the rest of the miners.

Small blockchains *are* vulnerable to 51 % attacks because there are few miners in the network. Such an attack was made in 2016 against the blockchain Krypton [40]. This particular blockchain is a side chain of Bitcoin which means that although it is a different chain, it is linked to Bitcoin. The idea behind side chains is that one can own Bitcoins and still use another blockchain without having to buy multiple currencies. The currency is sent

to a specific address in its parent chain (in this case Bitcoin) which locks the funds and the same amount is given to the user in the side chain (in this case Krypton). A side chain is much smaller than its parent and has a substantial smaller network of nodes, hence a '51 % attack' may be possible in practice. With a small network of nodes it is easier to obtain the majority of the processing power since the entire network has limited computing power in total. This fact leads to the conclusion that *small* blockchains are vulnerable to such attacks, but the larger ones, like Bitcoin and Ethereum, are assumed to be safe.

Sybil attack

A sybil attack is when someone creates multiple identities and by this tries to control the majority of the network, so this attack is related to the '51 % attack'. However, multiple identities will not give an attacker more computing power. The computational power would be divided with the number of entities created, so if someone created 1000 entities, each would have 1/1000 computational power. The sum would still be the same so this does not give any advantages to the attacker in the blockchain concept [34].

DoS (Denial of Service) attack

This involves sending a lot of data to a node to overload it so that it can not process normal transactions. In both the Bitcoin protocol and the Ethereum protocol there are several built-in mechanism to prevent this type of attack, but the simplest is the protection the transaction fees provide. The sender has to pay a fee for each transaction, in Ethereum this is called 'gas'. So if someone tries to overload the system with a lot of fake transactions it would be expensive. The miners also decide what transactions to include, typically they choose the ones with a high reward and these types of transactions would not be one of them. They would eventually be picked up by a miner and included in a block based on waiting time, but they would still have to send many of these at once to overload a node.

DDoS (Distributed Denial of Service) attack

This is in principle the same as a DoS attack, but in this scenario the attacker uses a botnet so he/she has more resources and therefore more damage can be done if it succeeds. Even with the use of a large botnet the attacker would not be able to 'crash' the network because of the transaction fees. However, the attack on the Bitcoin side chain Krypton [41] was a two-step attack were the second part was a DDoS attack made possible by first performing a '51 % attack'. So with a combination of attacks it would be possible, but since the larger chains such as Bitcoin and Ethereum are assumed safe from the first of these attacks they are

also considered safe from the second part of the attack. Without the first part of the attack it would be very expensive, so it is only feasible when someone controls the majority of the network.

Chapter 3

Description of Bitcoin

3.1 Overview

Bitcoin was made to meet the requirements of a decentralized cryptocurrency. The currency is called Bitcoin (BTC) and since its launch its value has been highly fluctuating. The value was stable at about $13 or less up until 2013 when it rose to reach $1 155,05 by December 4^{th} 2013 as shown in Figure 3.1. By the end of the same year, the prices started dropping from its first peak and by the first quarter of 2014 it was halved. After this date the price to USD have fluctuated with multiple peaks until suddenly it reached $19 535,70 by the end of 2017 [7]. As of 29^{th} of June 2018 the price has gone down to $5 905,97 as displayed in Figure 3.2. This sudden, drastic peak in value also happened for other cryptocurrencies in the same time period, but Bitcoin's peak was far more dramatic. As of March 14^{th} 2019 the price for one Bitcoin is $3 917,27.

Units of Bitcoins are usually expressed as one of the following [14]:

- BTC- Bitcoin. This is one full Bitcoin.
- dBTC- DeciBitcoin. This is 1/10 of a full Bitcoin.
- cBTC- CentiBitcoin. This is 1/100 of a full Bitcoin.
- mBTC- MilliBitcoin. This is 1/1000 of a full Bitcoin.
- uBTC- MicroBitcoin. This is 1/1000 000 of a full Bitcoin.
- Satoshi. This is 1/100 000 000 of a full Bitcoin.

Mining Bitcoin has become somewhat of a business since you can buy mining rigs with specialized, expensive equipment. Because of this the system has become more centralized

Fig. 3.1 Chart from CoinMarketCap [7] of Bitcoins prices in 2013 and in 2014.

Fig. 3.2 Chart from CoinMarketCap [7] of Bitcoins prices from 2013 to March 2019.

Fig. 3.3 The Bitcoin blockheader.

where some entities control a significant part of the computing power, and mining is now out of scope for the common user who only has a laptop or desktop computer. This is in contrast to its original purpose; *decentralization* of currency. The PoW kind of mining in Bitcoin is also very environmentally hostile because of its high use of electricity.

The total number of Bitcoins to mine is fixed at 21 million. It was designed to be scarce and resembles the rate that gold is mined and this is were the term 'miner' comes from. Its inflation rate also resembles gold and hence is decreasing over time and reaches zero when all Bitcoins are mined [10]. As of February 2019 about 84% of all Bitcoins are mined and in circulation.

The block reward for mining a new block was at first 50 BTC, but this reward is halved every 210 000th block. In 2012, the block reward was 25 BTC, and as of today it is 12,5 BTC. When all Bitcoins are mined, the network is still dependent on the miners to mine new blocks with transactions to keep the network going, but mining will be less lucrative than it is today. A possible solution is to increase the transaction fees after all coins are mined. The Bitcoin blockchain will most likely never change their mining protocol since their largest mining pools have invested in expensive hardware for PoW mining which then would become useless.

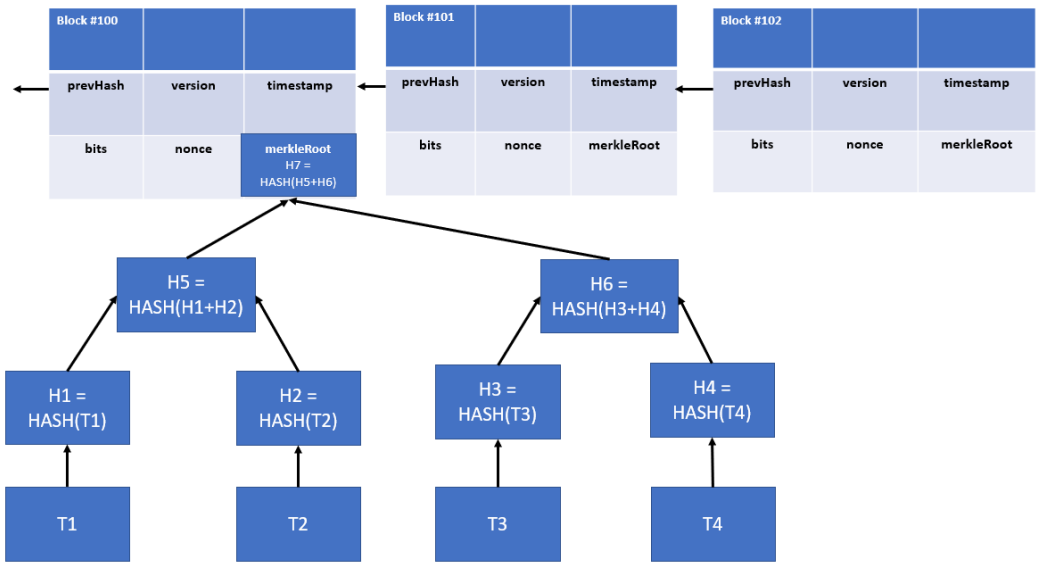


Fig. 3.4 The structure of Bitcoin.

3.2 Bitcoin blockheader

We will now go into more technical details about Bitcoin and how the system works. We will start by explaining the blockheader and what information each block contains which is a good starting point to later see how the blocks are connected and mined.

Figure 3.3 shows the content of the blockheader which is made up of six fields:

- prevHash: The hash value of the previous block; double hashed with SHA256.
- version: Decides what validation rules to follow for validating a new block.
- timestamp: Timestamp of when the miner started to hash the block header.
- bits: Difficulty of mining the block.
- nonce: Value for verifying the Proof of Work is valid.
- merkleRoot: The root of the Merkle Tree of the block's transactions. We will go into Merkle trees in Chapter 4.3.4.

Adding the transaction list for each block, the chain structure is shown in Figure 3.4.

This is essentially what Bitcoin is; a chain of blocks which are dependent of all that came before and where each block has a list of transactions that it executes when it is mined and validated. This has two interesting parts to it. The first is how the blocks are added and at the

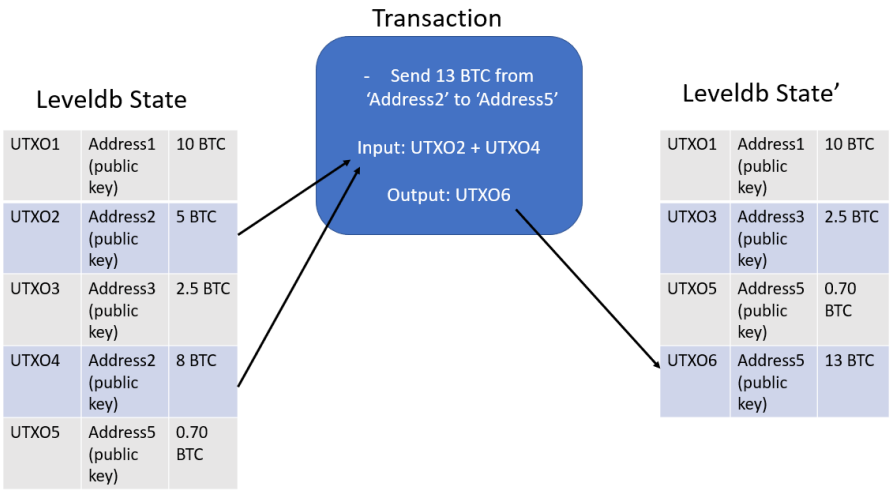


Fig. 3.5 Illustration of how the state transition function changes the state with UTXO concept visualized.

same time the transactions executed, and the second is how this is accepted as the truth by every other node in the network since only miners add new blocks to the chain. We will now have a look at what is called the 'State Transition Function', the mining protocol, the mining algorithm and how blocks are validated by others.

UTXO - Unspent Transaction Output

The coins within the Bitcoin protocol are actually called Unspent Transaction Output, or UTXO for short. Each time a coin is spent as an input in a transaction, the coin is removed from the state and the output coin is added. So the only coins that exist are those that are outputs and not yet spent in a new transaction. An UTXO has an owner and a denomination, because the UTXO's are of different value depending of the sum of the inputs to the transaction they are outputs of.

3.3 State and the State Transition Function

Bitcoin's state is made up of all UTXOs together with their owner and denomination. Each node has a database with all UTXOs which makes up the state, but it is not encoded in the blocks in any way. Every transaction changes this state [34] and the state transition function is carried out multiple times per block, once for every transaction in the block.

Figure 3.5 demonstrates a simplified example where UTXO2 and UTXO4 is owned by the same address. UTXO2 and UTXO4 are removed from the state and the new resulting UTXO6 is added.

The owners address is a 160- bit hash of the public part of the Elliptic Curve Digital Signature Algorithm (ECDSA) key pair [3].

3.3.1 Transaction fees

There is a transaction fee for each transaction and this fee varies over time depending on supply and demand. The fee is calculated *per byte* because miners can only include transactions up to about 1 million bytes in total per block [9]. Miners choose which transactions to include in their block depending on the fee they receive, so if someone wants their transaction to be picked up quickly one should set the fee accordingly. All transactions eventually gets mined depending on a waiting period.

3.4 Proof of Work mining - processor intensive

Bitcoin uses the PoW mining protocol and the protocol essentially states that the hash of every block, in this case the SHA256 hash, has to be smaller than a given target. Since the hash has a fixed number of digits, this means that the hash value must have a certain number of leading zeroes to be valid. The only way to find this is by brute force; simply trying to combine the fixed parts of the block's header with a new nonce until one matches the target. This is done by incrementing the nonce for each try. The target is defined as [8]:

$$target = \frac{2^{256}}{Hd},$$

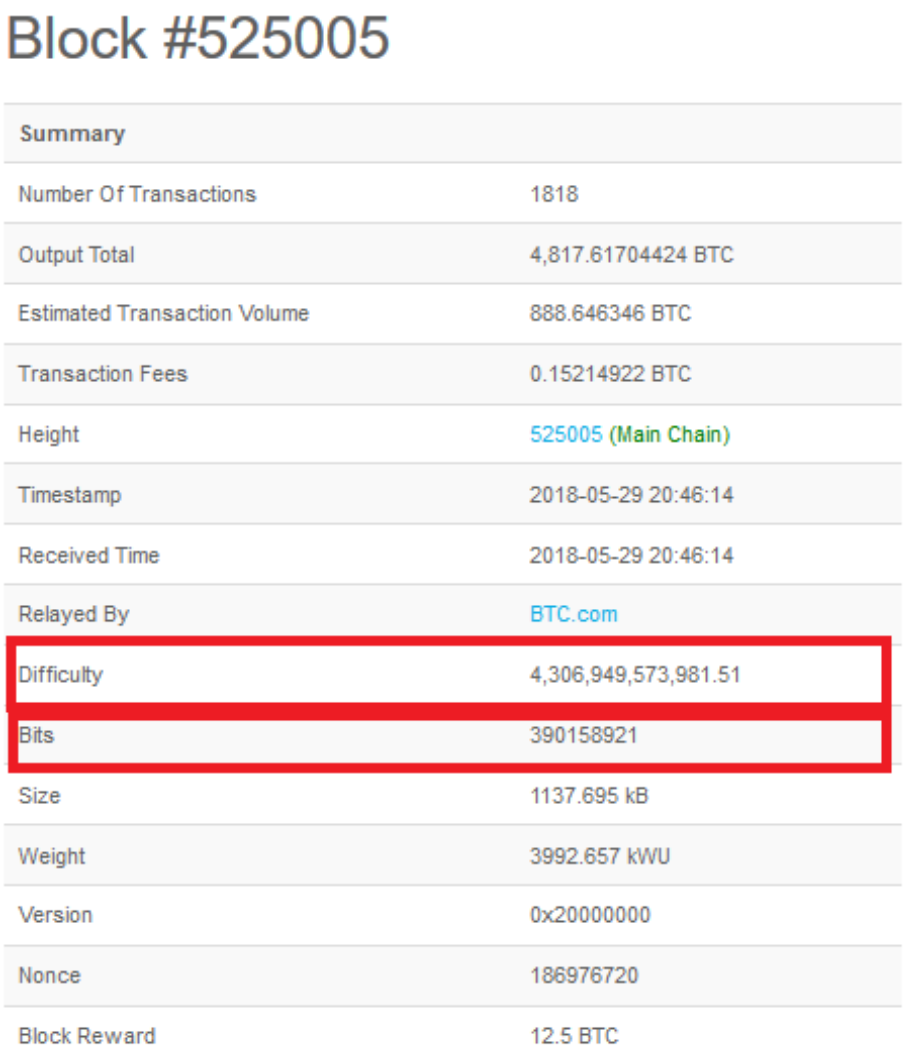
where 2^{256} is the highest possible target and Hd is the current block's difficulty. Let us look at an example from a real block [12];

We will take a closer look at the two marked components; 'Difficulty' and 'Bits'.

Difficulty

The component is used to calculate how many possible valid hash values there are by dividing *all* possibilities on this number.

The window of valid hash values in Figure 3.7 is in itself quite big since we are dealing with big numbers, but since there are so many more possibilities, the chance of finding such a valid value results in many failures before success. Because the difficulty is adjustable,



Block #525005

Summary	
Number Of Transactions	1818
Output Total	4,817.61704424 BTC
Estimated Transaction Volume	888.646346 BTC
Transaction Fees	0.15214922 BTC
Height	525005 (Main Chain)
Timestamp	2018-05-29 20:46:14
Received Time	2018-05-29 20:46:14
Relayed By	BTC.com
Difficulty	4,306,949,573,981.51
Bits	390158921
Size	1137.695 kB
Weight	3992.657 kWU
Version	0x20000000
Nonce	186976720
Block Reward	12.5 BTC

Fig. 3.6 Information from a real block in Bitcoin.

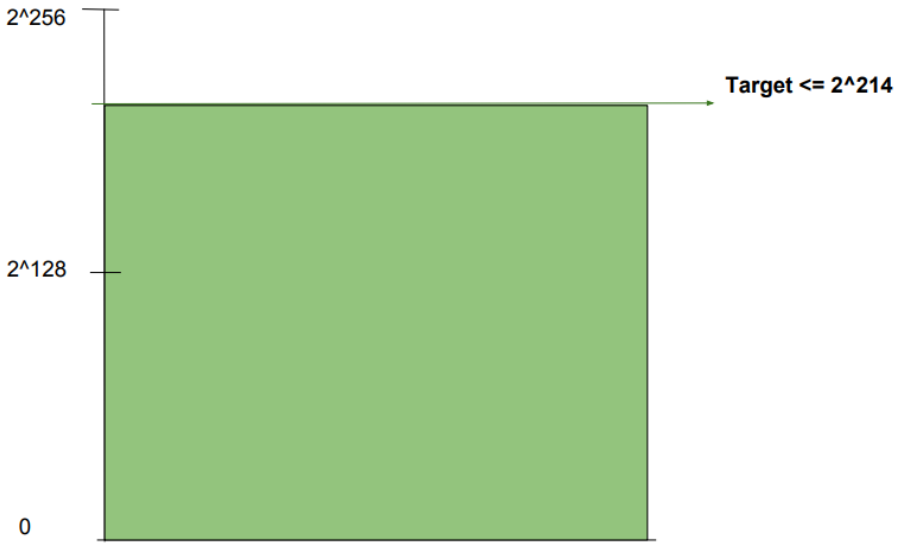


Fig. 3.7 The difficulty in Figure 3.6 results in this 'window' of possible valid hash values.

the time to mine a new block can be regulated as new miners enter or leave the network. If the difficulty goes up, the target space to find a valid hash gets smaller so it becomes more difficult to find a nonce matching the target space and hence it takes longer to mine. If the difficulty goes down, the target space to find a valid hash gets larger so it becomes less difficult to find a nonce matching the target space and hence it takes lesser time to mine. This is adjusted in every 2016th block such that approximately 6 new blocks are mined every hour, so about one block every 10 minutes.

Bitcoin's PoW algorithm is processor intensive. As mentioned before, specialized expensive hardware known as ASICs (Application-Specific Integrated Circuits) can be used for faster mining. Adjusting the difficulty prevents block creation from going faster than it is intended to, but the system is still becoming more centralized because few people have the resources to buy this specialized hardware, especially in large quantities. The adjustments in difficulty keep the mining rate somewhat stable, but it becomes increasingly harder to do the computational work behind it.

Bits

This component gives us the target hash value. The hash value of the block that is currently being mined has to be smaller than the target hash value. The bits are given in decimal; 390158921 which is in hex: 17415A49. The first byte; 17, gives us the length of the target

Fig. 3.8 Example of target hash with a byte length of 23.

Fig. 3.9 Example of target hash with a full byte length of 32.

which in decimal is 23. So this particular target hash has a byte length of 23. We have 4 bytes so adding another 19 bytes at the end gives us the example target in Figure 3.8.

SHA256 is a 32-byte hash function, so the target has to have even more zero padding, this time the zeros are added in the front. To this particular target we need to add another 9 bytes and we get the 32-byte value in Figure 3.9 as a target. This means that our example target hash value has to have at least 18 leading zeros to be considered a valid hash value. Our example shows zeros as nibbles, so in bits this would be 72 leading zero-bits.

3.4.1 Hashcash

The algorithm Hashcash is used in Bitcoin mining and other systems as well, so it is not made particularly for the purpose of mining cryptocurrency. The concept is however improved upon in the case of Bitcoin by using SHA256 instead of SHA1 and double hashing it.

Algorithm 1 Hashcash

Input: New block's header + current nonce guess

Output: Hash value, nonce (to be put into the new block for broadcasting and validation)

```
1: function INITIALIZE(Hash value)
2:     Hash value = SHA256(SHA256(Header + nonce))
3: end function
4: while Hash value > target do
5:     Increment nonce
6:     Hash value = SHA256(SHA256(Header + nonce))
7: end while
8: return Hash value, nonce
```

Algorithm 1 shows the Hashcash algorithm written in pseudocode. At the end, when the hash value and the nonce are found, they are put into the new block and the new block is broadcast to the network for validation.

The original Hashcash only hashes once with SHA1 and the proof it provides is meant to detect e-mail spammers by checking whether the hash value of the senders' address is not marked as valid, or if the hash value is registered in the receivers database of possible spammers [37]. In the case of Bitcoin it is used to verify valid blocks to add to the ever growing chain.

Block validation

The new block must be validated by all other nodes in the network, this way users of the network do not have to trust the miner; the entire network is in consensus. The miner that first creates a new block broadcasts it to the network and each node validates and executes the transactions in it. Their local version of the block chain is then updated as well as their local copy of the state.

A valid block has a reference to the previous block that exists and is valid. The new block's time stamp is also checked and has to be greater than the time stamp of the previous block and less than 2 hours into the future. The proof of work on the block is validated before any of the transactions are executed. This is done by double hashing with the given nonce and checking the result against the target. If this is within the target, the transactions are sequentially executed and changes to the state are made if none of them returns an error [34].

3.4.2 Simplified Payment Verification - SPV

Light nodes are nodes in the network that do not download the entire blockchain which is roughy the size of 190 GB, but obtains blockheaders from other, full nodes. These light nodes use a method called simplified payment verification, SPV, which is described in the Bitcoin Whitepaper [10]. The method relies on trusting the full nodes since the transactions are not verified by the light nodes themselves, but obtained from full nodes.

To summarize, one block is added to the chain as shown in Figure 3.10.

3.5 Bitcoin mining in the real world

ASICs are microchips that are customized for a particular use. In 2013, such a microchip was designed and released for Bitcoin mining [5]. It is much faster than any other technologies used to mine before it. Earlier mining was CPU based, then it switched to GPU which is

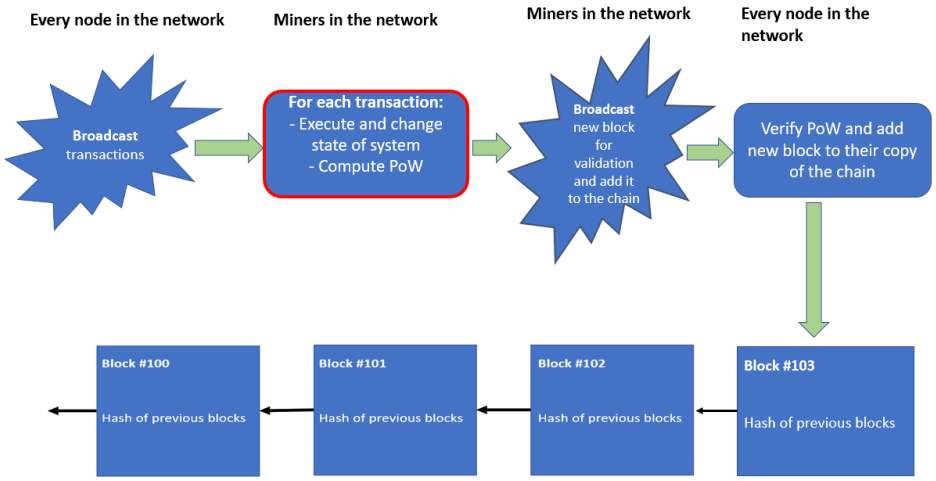


Fig. 3.10 The blockchain itself is pictured at the bottom which every node has a copy of. The majority of the work is done by the miner in the PoW computation which is highlighted with red edges. The miner also decides which transactions to include.

faster than CPU mining [21]. Other specialized hardware has also been used for mining purposes, but today ASIC mining is the only profitable choice. ASICs are expensive hardware and thus makes mining out of scope for the common user of the network. If one is to join the network, own and use Bitcoins, the easiest way to do so is to buy BTC and not get into the mining business.

Miners organize themselves in so called 'Mining pools' where miners work together and share the block reward. It is estimated that about 81% of the Bitcoin mining is taking place in China [6] in such pools. This is due to China's low electricity costs [24]. Because of this fact, Bitcoin currently suffers from centralization, which is a contradiction to the blockchain's purpose of being decentralized. One such company is called Bitmain and is located in Beijing [11]. This company also operates the mining pool called 'Antpool' which is one of the largest mining pools in the world.

Figure 3.11 shows a graph from the site www.blockchain.com [20] of Bitcoin's growing hashing rate over time. As we can see, the hashing rate has grown substantially in 2018. This means that the network is more secure because it has more computational power, but also that the difficulty increases as we can see in Figure 3.12 and this makes mining out of scope for most people. So the introduction of both ASICs and mining pools have indeed had one positive impact; making the network more secure.

With these computer halls, the electricity waste is also an issue because of the PoW protocol which has the flaw of wasting a lot of electricity since all miners compete to find the next block to include in the chain.

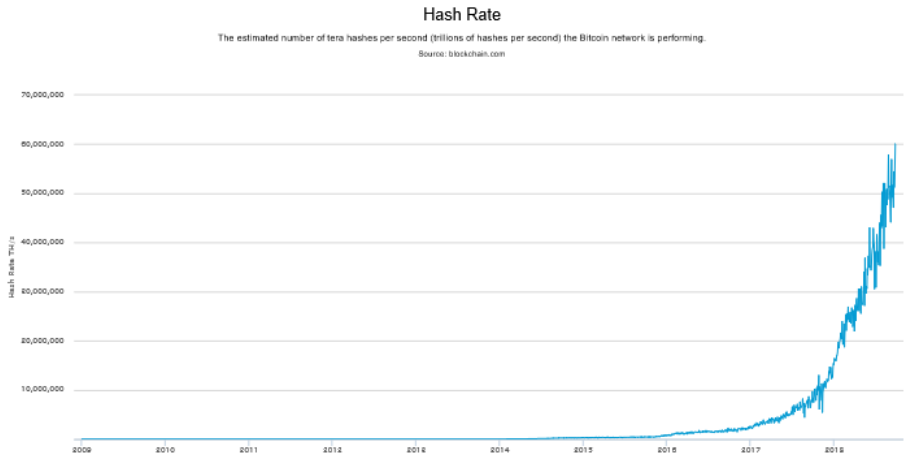


Fig. 3.11 Graph of Bitcoins growing hashing rate [20].

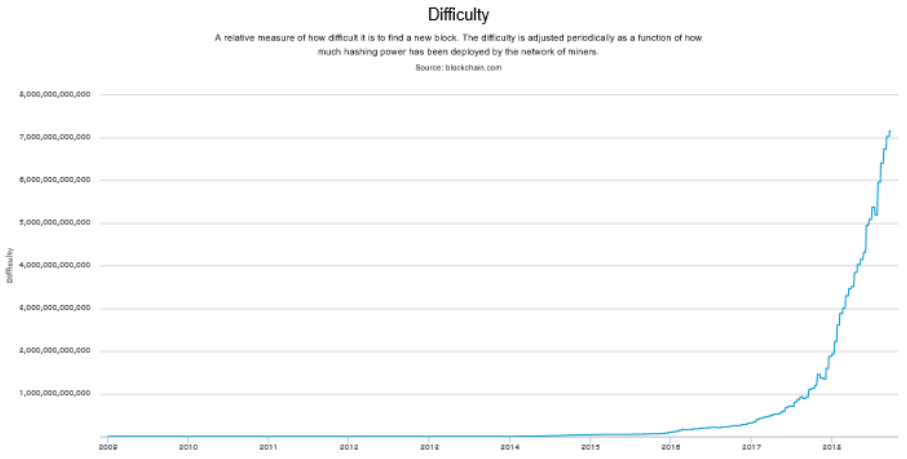


Fig. 3.12 Graph of Bitcoins growing difficulty [20].

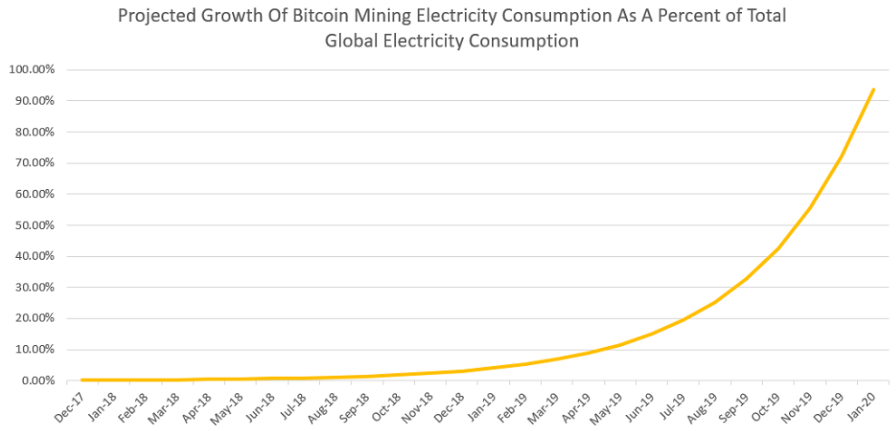


Fig. 3.13 Graph of Bitcoins' electricity consumption.

Bitcoin's electricity consumption is growing fast, Figure 3.13 shows a prediction of its percentage of the worlds electricity use up until January 2020 if the current growth is to continue [4]. This is an environmental threat and should be taken seriously. Also, most of the mining is centralized to China as mentioned, and although electricity is cheap it is not being generated in a green manner, rather it is mostly generated by coal power plants [25].

Chapter 4

Description of Ethereum

4.1 Overview

Ethereum's purpose is to be both a cryptocurrency and in addition; to be a general blockchain [34] where anyone can build applications on top of the chain. This kind of use of the blockchain makes up a new internet paradigm, as stated in Chapter 2 and illustrated by Figure 2.3. Because of its use and purpose, the blockchain architecture differs from that of Bitcoin. The blocks have more information stored in them [34] and the most important difference is that the blocks store the most recent state by a pointer to a modified Merkle Patricia trie.

Ethereum's currency is called Ether and its value has also fluctuated and peaked 14^{th} of January 2018 at $1 338,67. As of March 2019, the price for one Ether is $137,26. Figure 4.1 shows the chart from Coinbase with the previous and current prices of Ether.

Each unit of Ether has its own unique name. They are listed as units of Wei, which is the smallest units of Ether and the base unit [27]:

- Wei.
- Babbage - Kwei. This is 10^3 Wei.
- Lovelace - Mwei. This is 10^6 Wei.
- Shannon - Gwei. This is 10^9 Wei.
- Szabo - microEther. This is 10^{12} Wei.
- Finney - milliEther. This is 10^{15} Wei.
- ETH - Ether. This is one full Ether and 10^{18} Wei.

Fig. 4.1 Chart from Coinbase [26] of value of Ether in USD.

Block #100			
parentHash	ommersHash	beneficiary	logsBloom
difficulty	number	gasLimit	gasUsed
timestamp	extraData	mixHash	nonce
receiptRoot	transactionsRoot	stateRoot	

Fig. 4.2 The Ethereum block header.

In contrast to Bitcoin, there is not a fixed total number of Ether to be mined, but only a maximum of 18 million ETH can be mined per year. This means that the relative inflation is decreased over time, also since some Ether are lost every year by their owners [29].

4.2 Ethereum blockheader

Before we will go into more details about the whole system, we will have a look at the blockheader.

Figure 4.2 shows the Ethereum block header with the following fields:

- parentHash: Keccak256 hash value of the previous block header.
- ommersHash: Keccak256 hash value of the ommers list.

- beneficiary: Address (160 bit public key) of the block's miner.
- logsBloom: Bloom filter of all logs from all receipts of every transaction in the block.
- difficulty: Value that describes the block's difficulty.
- number: Number of the block.
- gasLimit: Limit of gas to be used per block.
- gasUsed: Gas used in total for this block; for all its transactions.
- timestamp: The time the miner started mining this block.
- extraData: Byte array with data for this block where array $\leq$ 32 bytes.
- **mixHash**: Keccak256 hash value of all other header fields in this block except itself and 'nonce'.
- **nonce**: 64-bit random value.
- stateRoot: Keccak256 hash value of the root node of the state trie.
- transactionsRoot: Keccak256 hash value of the root node of the transaction trie for the block.
- receiptRoot: Keccak256 hash value of the root node of the receipt trie for the block.

The components 'mixHash' and 'nonce' together proves that the computational work has been done for this block and are used to verify that the proof of work is valid. Each block has a list of transaction receipts in addition to a list of the transactions. The receipts are made up of 4 items which are found after the current transaction is executed; the state, the amount of gas used, set of logs created and Bloom filter composed of these logs [31]. The component 'logsBloom' is included to make it easier for an application to find relevant logs; it can scan the Bloom filter of each block header to see if the block contains relevant logs and if it does; execute those transactions and get the logs that the application needs [13].

To get the full picture of Ethereum's chain structure we have to look at the data structures in Ethereum to better see how it differs from Bitcoin and hereby improves upon the block chain concept.

4.3 Data structures

The data structure 'Merkle tree' is used in both Bitcoin and in Ethereum, but Ethereum uses the more advanced structure 'Modified Merkle Patricia Trie'. DAG creation and DAG usage is part of Ethereum's Ethash algorithm for mining.

4.3.1 DAG - Directed Acyclic Graph

The data structure used in the Ethash algorithm is an array, but this array can be viewed as a directed acyclic graph. A directed acyclic graph is a graph with a finite number of nodes where every node point to another node without any cycles. This means you can not start at a node v and follow a path in the graph and end at node v again. Ethashs' DAG and its generation will be explained in details in Section 4.6.2.

4.3.2 Trie

This structure is used for storing an associative array. The indices of the array is the nodes' positions in the trie and the values in the nodes are the values of the array. The edges in the trie represents the alphabet and the words of the alphabet are retrieved by traversing down a branch of the trie, adding letters down the path. One special type of trie is a Radix trie where the radix is the maximum number of children a node can have, that is, the number of letters in the alphabet.

4.3.3 Patricia trie - Practical Algorithm To Retrieve Information Coded In Alphanumeric

This is a compressed form of a trie and therefore it requires less memory to implement. It compresses nodes by merging the key with its parent if it is the only child.

The example in Figure 4.3 shows five words; Ether, Ethereum, Ethereal, Bitcoin and Bittorrent. To include a new unrelated word, for instance 'WEI', a node with the key 'WEI' would be added to the root as a third child. This trie has a radix of 26; one for each character in the alphabeth used. It could be a radix 2 tree if the words were interpreted binary, but this would be a deeper trie.

4.3.4 Merkle tree

A Merkle tree is a hash tree where every leaf node contains the cryptographic hash of a data item and every non-leaf node contains the cryptographic hash of its one or two children. This

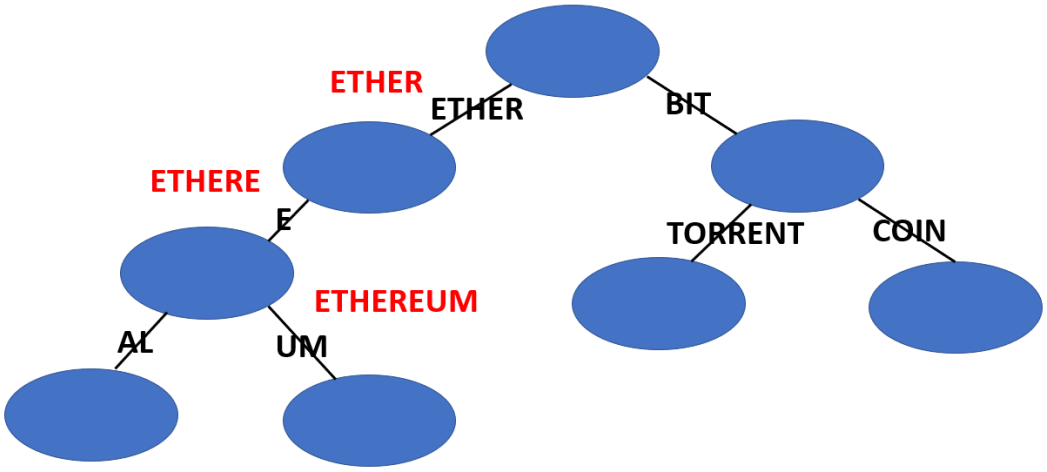


Fig. 4.3 Example of a Patricia trie and the search for the word 'Ethereum'.

tree type is a hash chain and propagates *upwards* to the root, or to the top hash. This implies that making changes within the tree will make changes upwards in the tree and eventually change the root hash as well. Figure 4.4 shows an example of a binary Merkle tree.

4.3.5 Modified Merkle Patricia trie

By using a combination of a Merkle tree and a Patricia trie we obtain a secure and efficient trie structure. In Ethereum, the root hash is publicly known meaning that anyone can give a proof that a [path, value] pair exists by providing the nodes that goes up that particular path to the root of the tree [43]. An attacker can not convince others that a 'fake' [path, value] pair exists, since this will change the root hash at the top that is public. Because of this we can say that the database, that is made up of the data in the tree structure, is immutable.

To implement this kind of trie structure, different types of nodes are used [55]:

1. *Leaf nodes:* Lists of [key, value] pairs with prefix either 2 or 3. Prefix 2 means that the node contains an even number of nibbles (for instance the key ends in "13"), and prefix 3 means that the node contains an odd number of nibbles.

2. *Branch nodes:* Lists of length 17 with the first 16 elements being every character in the hexadecimal alphabet and the final holding a value if there is a [key, value] pair where the key ends at this branch node.

3. *Extension nodes:* Lists of [key, value] pairs with prefix either 0 or 1. The 'value' is different in this type of node; it is the hash of another node and is used as a pointer.

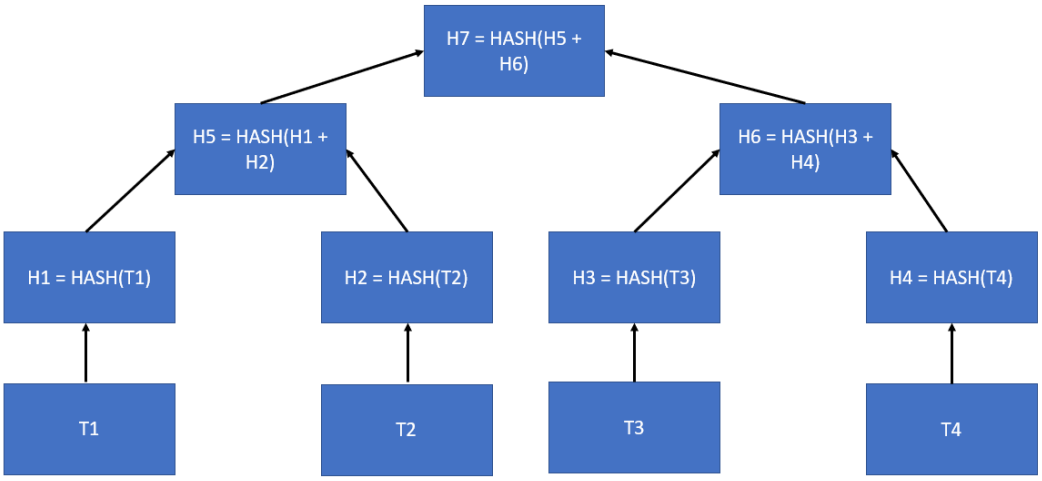


Fig. 4.4 Example of a binary Merkle tree. The data blocks are at the bottom, in this case Transactions T1.. T4 and the hash values H1.. H7 are hashes of its two children.

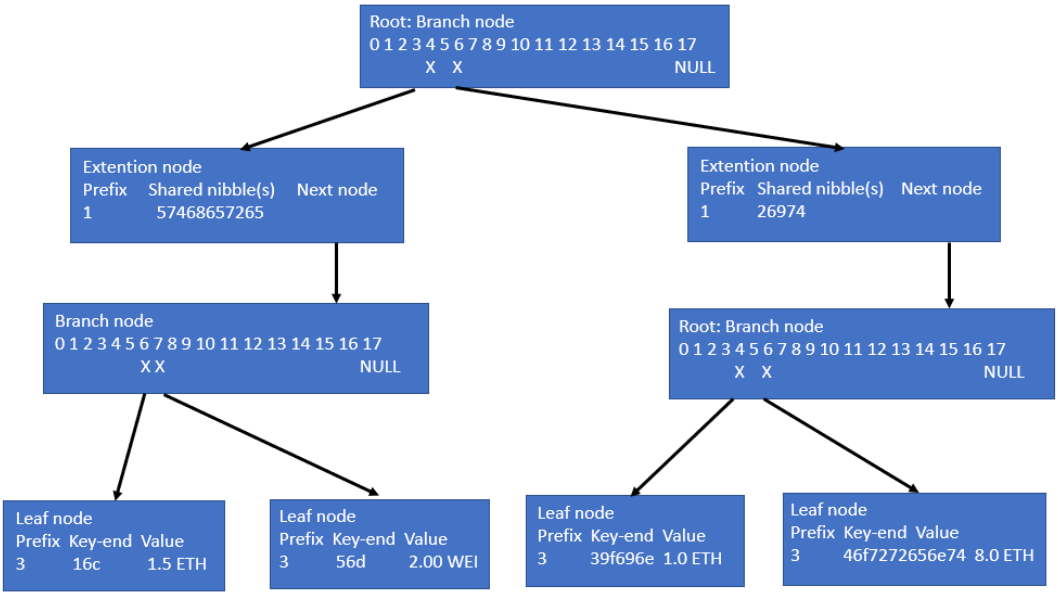


Fig. 4.5 Example of a Modified Merkle Patricia Trie with a simplified World State.

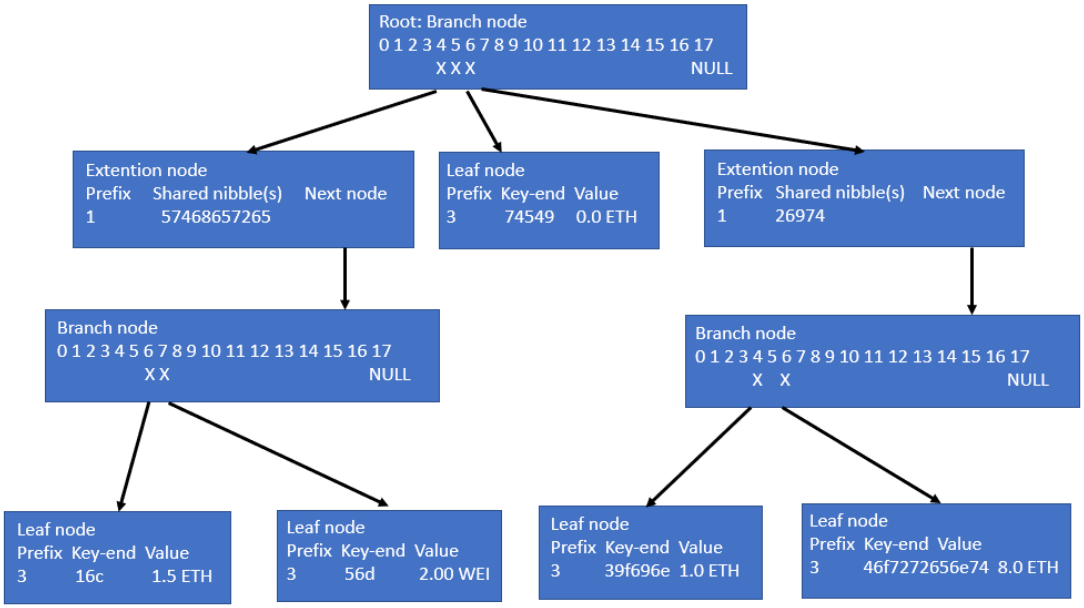


Fig. 4.6 Example of a Modified Merkle Patricia Trie with a simplified World State after adding the unrelated word 'WEI'.

> Prefix 0 means that the node contains an even number of nibbles and prefix 1 means that it contains an odd number of nibbles.

In Figure 4.5 there are four words with the following hexadecimal notation: 'Ethereum' with hexedecimal '457468657265756d', 'Ethereal' with hexadecimal '457468657265616c', 'bitcoin' with hexadecimal '626974636f696e' and 'bittorrent' with hexadecimal notation '626974746f7272656e74'. To include a new unrelated word; 'WEI' with hexadecimal notation '574549', a leaf node would be added to the root node with a pointer from '5' as shown in Figure 4.6 since the root is a branch node that can have a maximum of 16 children.

If instead the word 'CASH' with hexadecimal notation '43415348' that also starts with '4' should be included, the roots' child would be changed to be a branch node with one leaf node containing '3415348' for the new word and one extension node with shared nibbles '7468657265' as before. This is shown in Figure 4.7. In case of deleting the words 'bitcoin' and 'bittorrent', the root node would be altered to be an extension node with '457468657265' as a shared nibble, see Figure 4.8 for the resulting new World State after this deletion.

With a mix of the Merkle trie and the Patricia trie and by using additional types of nodes we obtain a trie that stores cryptographically authenticated data, and that has a run time of

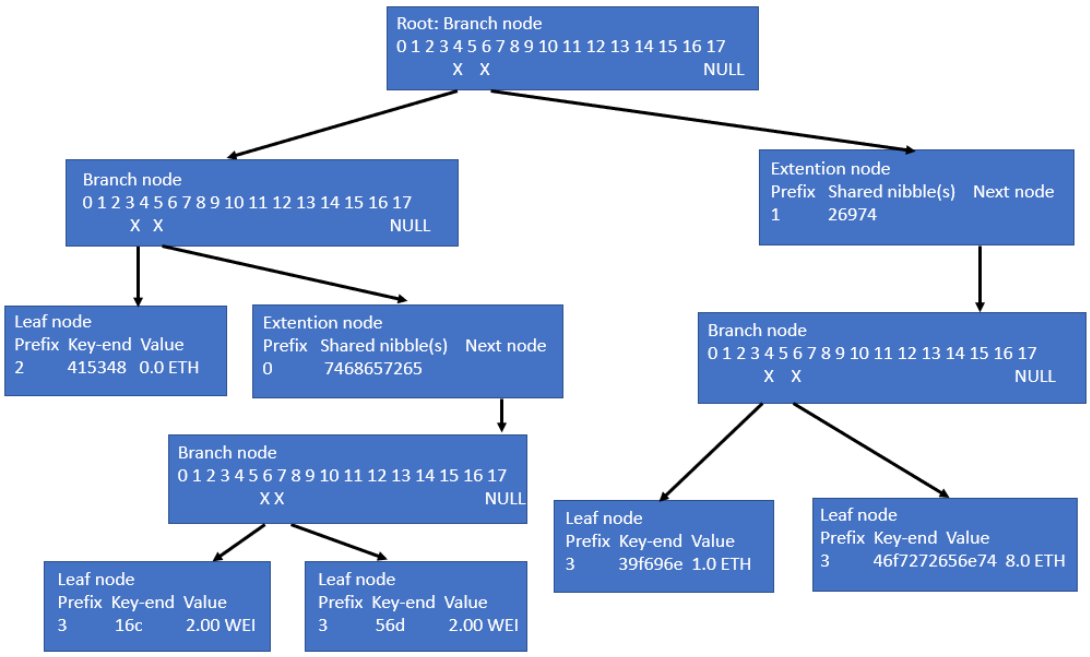


Fig. 4.7 Example of a Modified Merkle Patricia Trie with a simplified World State after adding the word 'CASH'.

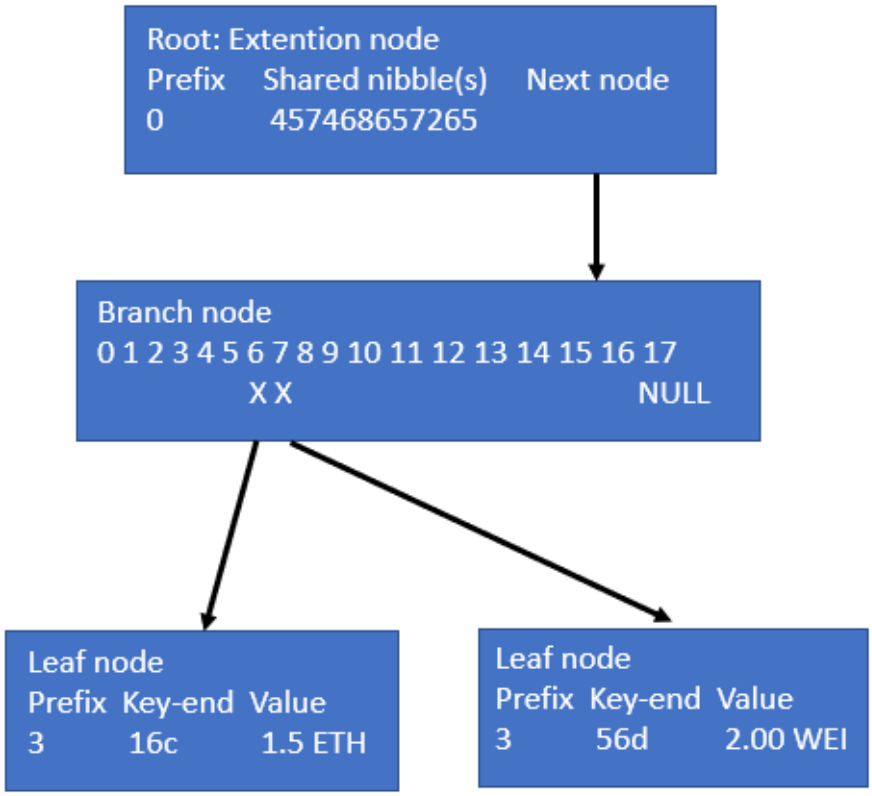


Fig. 4.8 Example of a Modified Merkle Patricia Trie with a simplified World State after deleting the two words 'bitcoin' and 'bittorrent'.

$O(log(n))$ for searching, deleting and inserting. Note that light nodes in the network only need to download certain branches related to their data queries.

4.3.6 Accounts

Accounts are objects with 4 fields; nonce, balance, contract code (if any) and a pointer to a storage trie. The nonce in this context is a counter to make sure each transaction is only processed once [34]. The accounts are identified by a public key, or an address, like the UTXOs in Bitcoin also have a public key for their owner. Ethereum views the system from the users' point of view; the accounts, and not from the coins' (or UTXOs, as in Bitcoin).

The account addresses are equal to Bitcoin addresses; a 160- bit hash of the ECDSA public key in a key pair [23].

4.4 State and the State Transition Function

The state in Ethereum contains all the account objects with their associated data. Unlike Bitcoin, the state in Ethereum is not abstract, but stored in the World State Trie which is global and changes for each transaction. In Figure 4.9 we see that all the blocks in the chain point to the same State trie, i.e all transactions change the same trie. The State Trie is a modified Merkle Patricia trie. Each account in the State Trie also has a Storage Trie as a modified Merkle Patricia trie, only one of these are shown in the figure to show the concept. Each block has its own transaction and receipt trees as well, these are Merkle trees.

Ethereum as a state transition function is illustrated in Figure 4.10. The figure is very simplified, the account's storage trees are not pictured and the transaction only sends 0.5 ETH from one account to another with no code execution.

4.4.1 EVM - Code Execution and Smart Contracts

Ethereum is a programmable blockchain and the Ethereum Virtual Machine, EVM, was created since no existing VM met the requirements for code execution. EVM is also called 'The World Machine' since all full nodes have EVM installed to compile and run the same code, so when all nodes can agree on the same computational results it can be viewed as one single global computer.

Each transaction in the blockchain contains a data byte array with EVM code. EVM code is written in a stack-based bytecode language and contains elements from Bitcoin Script, assembly and Lisp. An EVM program is made up of a sequence of operations called opcodes [35].

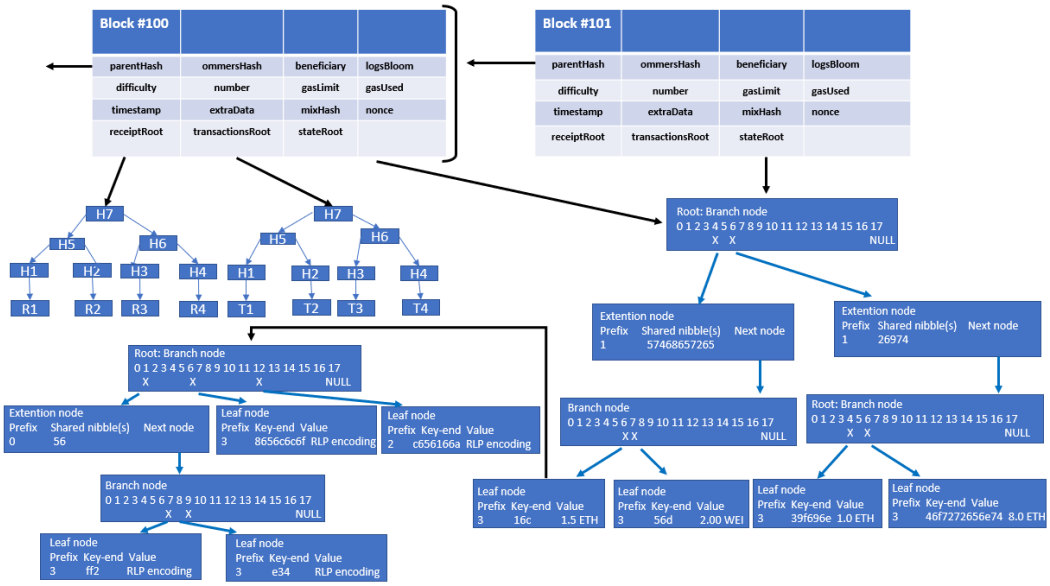


Fig. 4.9 Small example of Ethereum's tree structure.

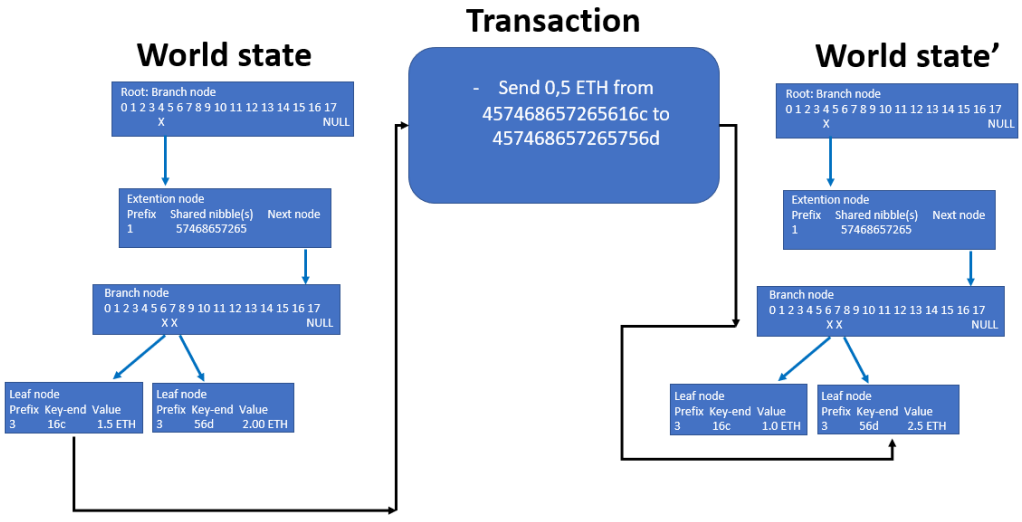


Fig. 4.10 Ethereum as a state transition function.

As mentioned, Ethereum is programmable. This means that one can write programs called Smart Contracts. These are written in a high-level language, such as Solidity [54], which compiles the program to EVM code. A contract consists of a constructor and different functions, has its own address and its own Ether balance. They are also called 'first class citizens' of Ethereum because they have equal rights as externally owned accounts and can also create other contracts. The deployment of any contract and interaction with it is treated like any other transaction in the blockchain. Every node in the network compiles and runs the code and the owner pays gas for creating it, while the ones that interact with it pays gas for use of its functions. Every account in Ethereum has its own code storage for such contracts, regardless of whether the account owner has deployed smart contracts on their own. Every smart contract also has their own account so it can hold its own currency.

There are several high-level languages to make Smart Contracts, all of which are Turing complete. This means that, in contrast to scripts, they have loops and conditional statements in addition to storage of variables. In principle, EVM can do anything a regular computer can and this opens up to much more use cases for blockchains.

4.4.2 Transaction fees - gas

Transaction fees are called gas and are calculated for each operation that the EVM performs. Each operation has its own cost, and more complex operations cost more to perform. Every transaction also has a base cost which varies over time. The same mining priority principle as for Bitcoin also applies in Ethereum; transactions with a higher set fee, or gas, are included by the miners first [32]. The gas prices are set in Gwei and are paid to the miner of the block where the transaction is included.

The Halting Problem

The Halting Problem is when a program goes into an infinite loop. Bitcoin avoids this by using a scripting language that does not allow any loops in the code. Ethereum on the other hand, is programmed in high-level Turing complete languages so an infinite loop *could* happen. This is solved by defining a gas limit for each block and each transaction so that if such a loop was programmed in a smart contract, it would stop when the transaction ran out of gas. See Appendix A for some details and examples written in Solidity.

4.5 GHOST - Greedy Heaviest Observed Subtree

The Ghost protocol was first introduced by Yonatan Sompolinsky and Aviv Zohar in December 2013 and calculates the longest chain by including orphan blocks in this calculation. An orphan block is a block that a miner has successfully mined, but another miner has gotten his block verified by most nodes first. This means that the miner with the orphan block has to discard his block if the GHOST protocol is not applied. A version of the Ghost protocol is implemented in Ethereum and it includes 7 orphan blocks which is then called Uncle blocks, or Ommers [34].

This version of the GHOST protocol is implemented to decide which chain is the true chain in case of a fork, and also to help avoid centralization of the system because blocks take time to propagate through the network. When a new block propagates through the network another miner might have mined a new block within that time frame, and without this protocol all of these blocks would have been wasted. With Ghost, orphan blocks also contribute to the security of the system [36] because they also provide PoW. The miners of Uncle/Ommers blocks that are included in a new block also get a block reward; 2.625 ETH, while the miner of the block that is added to the main chain is given 3 ETH as a block reward.

4.6 Proof of Work mining - memory intensive

Recall that both Bitcoin and Ethereum uses PoW mining where the goal is to find a hash value, in this case the KECCAK256 hash, with a certain number of leading zeros. The target is also here defined as:

$$target = \frac{2^{256}}{Hd},$$

where Hd is the difficulty of the current block and 2^{256} is the maximum possible difficulty.

Ethereums PoW algorithm is memory intensive, also said to be ASIC resistant. ASIC resistance means that ASICs can not be used to mine as they lack the memory needed for mining. This implies that mining is, in contrast to Bitcoin which uses the same protocol but not the same algorithm, in the scope of the common user. Ethereum chose this memory intensive solution to avoid the use of ASICs where mining is out of scope for most people making mining somewhat centralized and in contrast to its original intent.

4.6.1 Ethash

The mining algorithm in Ethereum is called Ethash [28] and uses a data structure known as a *directed acyclic graph*. This is essentially an array where every index in the array is called a 'page'.

4.6.2 DAG generation

The whole data structure is generated from a public seed. The first element in the data structure is $DAG[0] = y^w$ where $y = Keccak256(seed)$ and $w = 32$. The rest of the elements follow the formula $DAG[k-1] = (y^{kw} + DAG[y^{kw} mod (k-1)])^w$, where all exponentiations are computed modulo a large prime p. As we can see, the elements in the DAG consists of multiplications of the hashed seed added with some pseudo random previous elements. This formula in addition to the *seed* and the variable w is known to all nodes so that everyone can compute it and store it [28]. The number of pages to compute, N, is another global parameter of the system.

The whole resulting DAG is stored into the miners' memory:

$$
\begin{aligned}
&DAG[0] = y^w \\
&DAG[1] = (y^{2w} + DAG[0])^w \\
&DAG[2] = (y^{3w} + DAG[y^{3w} mod\ 2])^w \\
&DAG[3] = (y^{4w} + DAG[y^{4w} mod\ 3])^w \\
&... \\
&... \\
&DAG[k-1] = (y^{kw} + DAG[y^{kw} mod\ k-1])^w \\
&... \\
&... \\
&DAG[N-1] = (y^{Nw} + DAG[y^{Nw} mod\ N-1])^w
\end{aligned}
$$

We now look at a small example to illustrate the graph properties. Let $y = 2, w = 2, p = 97$, $N = 11$ and every exponent be computed modulo 97. To generate the DAG with these parameters we compute the following:

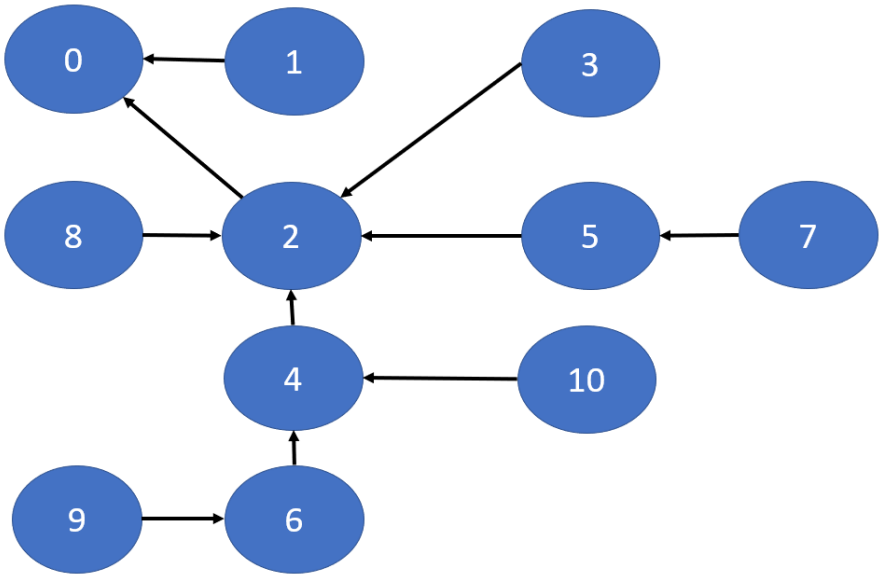


Fig. 4.11 Graph of the pointers in the example array which constitutes a DAG. The nodes are marked with the index numbers.

$$
\begin{aligned}
DAG[0] &= 2^2 = 4 \\
DAG[1] &= (2^4 + DAG[0])^2 = 12 \\
DAG[2] &= (2^6 + DAG[0])^2 = 65 \\
DAG[3] &= (2^8 + DAG[2])^2 = 27 \\
DAG[4] &= (2^{10} + DAG[2])^2 = 96 \\
DAG[5] &= (2^{12} + DAG[2])^2 = 3 \\
DAG[6] &= (2^{14} + DAG[4])^2 = 3 \\
DAG[7] &= (2^{16} + DAG[5])^2 = 22 \\
DAG[8] &= (2^{18} + DAG[2])^2 = 33 \\
DAG[9] &= (2^{20} + DAG[6])^2 = 81 \\
DAG[10] &= (2^{22} + DAG[4])^2 = 44
\end{aligned}
$$

The example array makes up a directed acyclic graph as illustrated in Figure 4.11. Each node corresponds to a page in the array, and points to the node with a lower index number that is needed to compute its value. Therefore, the data structure used in the Ethash algorithm is not just an array, but a directed acyclic graph.

The same DAG is used in the whole algorithm, but since the algorithm requires both storage of and many look ups in the data structure, Ethash is memory intensive. The length of the DAG, N, is determined by the current length of the blockchain and grows in size as

the blockchain grows. Every epoch (every 30.000 blocks, or about every 4^{th} - 5^{th} day) a new DAG is generated. To generate the DAG we need these parameters:

- *Seed*: The seed hash is always a hash of the previous seed hash, where the first was a hash of a series of 32 bytes of zeros. Every epoch a new seed hash is defined based on the previous.
- w: $\frac{Mix\ length\ in\ bytes(128)}{Bytes\ in\ word(4)} = 32$

The length of the DAG grows by 2^{23} for every epoch. The epoch number is calculated as: $Epochnumber = \frac{Blocknumber}{Epochlength}$.

Algorithm 2 Ethash

Input: New block's header + current nonce guess
Output: mixHash, nonce (to be put into the new block for broadcasting and validation)

```
1:  mix[0] = KECCAK256(Header + nonce)
2:  for i = 1 ... 64 do
3:      mix[i] = mix[i-1] + DAG[N/2 + (mix[i-1] mod N/2)]
4:  end for
5:  Hash value = Mix64 reduced to 32 bytes
6:  while Hash value > target do
7:      Increment nonce
8:      mix[0] = KECCAK256(Header + nonce)
9:      for i = 1 ... 64 do
10:         mix[i] = mix[i-1] + DAG[N/2 + (mix[i-1] mod N/2)]
11:     end for
12:     Hash value = Mix64 reduced to 32 bytes
13: end while
14: return mixHash, nonce
```

Algorithm 2 shows the Ethash algorithm given in pseudocode. The input of the algorithm is the new block's header, but without the components 'mixHash' and 'nonce' since these are the components the algorithm is computing. When a hash value below the target is found, the block is broadcast with this found hash value as the newly mined block's 'mixHash' component and the nonce that was used to find it as its 'nonce' component.

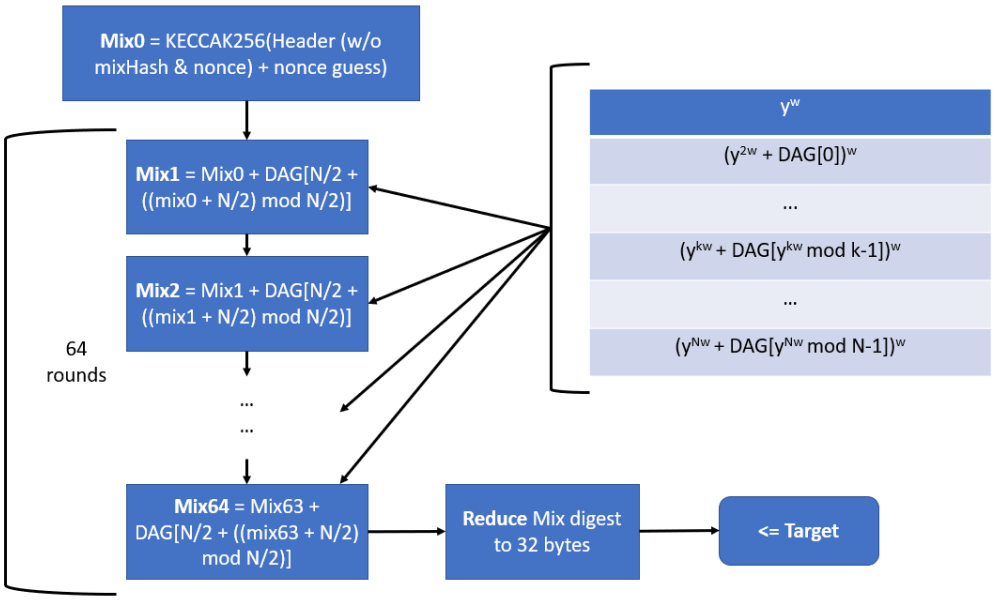


Fig. 4.12 The Ethash algorithm visualized per nonce guess.

4.7 Light Client Protocol

Since the blocks store a pointer to the most recent state, light nodes in the network only need to download block headers and not the entire blockchain itself [33] and it is still as secure. The light nodes in Ethereum verifies the state themselves in contrast to Bitcoin where these nodes trust the full nodes and the information that is given to them. In order to verify the transactions themselves, the light nodes do not need to store the whole DAG data structure which is roughly the size of 2.5 GB data as of June 2018. They only need to store a few MB to hold some of its first elements, called the cache, and they compute the elements they need from that.

Since they only need to check if the final nonce is within the threshold they do not need to access every element to do so. They simply run the Ethash algorithm and the mixing function gives them the indexes they need in order to verify the given nonce. If the index they need is not in the cache, the index will point to the DAG-page that is necessary to compute it. If this page is not in the cache, it will point to another one with lower index, and so on. They keep going upwards in the data structure until they get an index in the cache. Then they can go back to the found indexes and compute the elements.

The run time for the miners' computations of building the whole DAG is $O(n)$, while the run time for the light nodes are $O(log(n))$ to verify that a hash is valid.

Let us look at a small, concrete example: Say a light node has the following cache stored

$$
\begin{aligned}
DAG[0] &= y^w \\
DAG[1] &= (y^{2w} + DAG[0])^w \\
DAG[2] &= (y^{3w} + DAG[y^{3w} mod\ 2])^w \\
DAG[3] &= (y^{4w} + DAG[y^{4w} mod\ 3])^w
\end{aligned}
$$

and the nonce that needs to be verified requires a look up at index 50 which is not in the node's cache. The node then has to compute $DAG[50] = (y^{51w} + DAG[y^{51w} mod 50])^w$, which needs an element from the previous index $y^{51w} mod 50$. Say this index turns out to be 28, which the node also has not stored in the cache. To find this, the element in $DAG[28] = (y^{29w} + DAG[y^{29w} mod 28])^w$ needs to be computed. The index $y^{29w} mod 28$ also points to another previous index and so on. The node keeps going backwards until it ends up among the elements that it has stored and knows the values of. Now, the node can add the values sequentially until he has the value of the element that it originally needed to verify the given nonce. This is done 64 times according to the Ethash algorithm for verification of the nonce.

Block validation

Each node validates the newly broadcast block and checks if the previous block it refers to exists and is valid. The nodes also checks the time stamp; it has to be greater than that of the previous block. Other things to check for include block number, difficulty, transaction root, uncle root and gas limit. Before all the transactions are executed and thus changes in state is made, the rroof of work has to be validated. This is done by running the Ethash algorithm with the given nonce and checking whether it yields a valid hash value according to the target. Remember that the validation part of PoW is quick and not memory intensive, in contrast to the miners' part. Once PoW is validated, the transactions are executed and the gas limit in the block can not be exceeded in this process. Also, no transactions can return an error. The state root has to be equal to the state root given of the block after all transactions are executed and a reward for miner is included. If all of this checks out, the block is valid [34].

The system in its simplest form can be viewed very much the same as Bitcoin, see Figure 3.10. The difference in this simple form is that for each transaction, the EVM code has to be compiled and run. This is also true for each node when they have validated the new block.

Adding one block to the chain, and thereby the 'whole' system, can be viewed very much the same as for Bitcoin in Figure 3.10, but with the addition of code execution for the

transactions that involve smart contracts. These transactions results in code execution for every single node in the network.

4.8 Usage of smart contracts

The technology of smart contracts can be used in a variety of ways and we will now explain some of its use cases. Some of its applications refer to and benefit from the new internet paradigm, while some provides a more secure option than those used today. There are also some areas where blockchains are *not* the best option, we will have a brief explanation of these cases as well. The website www.stateofthedapps.com has a list of all applications built on blockchains.

4.8.1 Tokens - your own currency

Ethereum is meant to be a general application platform, so there has to be a way of defining your own currency within the system. This is done by creating Tokens that can be bought with the native coin Ether. For an example implementing this, see Appendix A.1.

4.8.2 Decentralized web applications (Đapps)

The use of decentralized web applications, also called 'Eth apps', eliminates the use of a trusted 3rd party. In these kinds of applications one can make good use of the option to create ones own tokens.

Streaming It might be a bit ambitious, but with this general application platform some might create their own streaming website for their music and/or movies. This could be the case for a record company, a movie company or even an unsigned artist. With some basic skills to write a contract and a decent website with their products there is no need for a 3rd party to provide the server to register/manage subscriptions or a bank to validate and transfer the payment. The users of their application validates their own payment and registration in addition to eliminating the third party payment 'bridge' between the two parties that actually wanted to make a deal with each other. Because these two do not know or trust each other this is usually solved by subscribing to a known streaming site where users make payments to the central and trusted third party, which again compensates the company owning the product.

As mentioned, this might not be an option for all people, but if we look at it the other way around; someone who has the skills (or the means to learn it) now has a platform which provides trust that they can use.

In this particular example one could make different kinds of tokens for different subscriptions which would give different permissions on the website. The subscriber signs in with his Ethereum account, this is made easy by using the Mist browser. If the Ethereum account owns a token for this website, this is recognized and it could be activated or used (depending on their purpose) when streaming from the site with the content.

Digital assets One can sell objects on a website. This can be different kinds of files, pictures, videoes etc. This can be managed by selling tokens as mentioned above and giving the users who own certain kinds of tokens the permission to download the various files.

4.8.3 Enforecment of agreements

This is the area where smart contracts really shows off their potential. Once the contract is on the blockchain, there is no possibility of changing it and once you agree to its terms by using it, you can not refuse to make a payment or some other agreement.

Insurance This usage can apply to bigger insurance companies, but also and maybe foremost unions and cooperatives. It can also be applied to a combination of these types of users. A smart contract can be programmed to make out certain payments when some criteria are being met. So let us say for instance that every month someone living in a cooperative makes a payment to a smart contract owned by the cooperative. Once a year an amount has to be paid to an insurance company and that can be handled as a sending of an amount of ether to another smart contract owned by said company.

Agreement between people who know each other Smart contracts can also be used by a group of people that knows each other, for instance neighbours, colleagues, band members, members of sports clubs or unions. Someone that has a project together involving some sort of joint payment on a regular, or irregular, basis. A smart contract can make out certain payments at certain times and these do not necessarily need a website or a front end to be up and running. There is only the need to program a smart contract and put it on the block chain and the people involved have to have an account on the block chain as well.

Digital Crowd funding Crowd funding using smart contracts reduces costs by not paying a third party a percentage of the fundings. Such a contract can have a goal of reaching a

certain amount of Ether in a certain amount of time. If the goal is not reached, the contract can make refunds to all addresses that donated funds. If the goal is reached, the contract will pay out all Ether to the person wishing to raise the funds. For an example of such a contract, see Appendix A.1.

Auction Think of known auction websites such as eBay. One can bid on different items and the one with the highest bid 'wins' the item. Auction contracts can also be made on the block chain. The contract can be programmed to send the item to the highest bidder, and refund all the others.

4.8.4 Proof of concept

In theory every web application can be made on top of block chains, but not every application benefits from this. There are a few applications made on top of Ethereum that demonstrates this, and serves best as proof of concept for what the block chain can do. We will look at one of them.

Social media Social media applications resembling the likes of Twitter, Facebook etc. do not demonstrate the best usage of block chains in my opinion. This is due to their big volumes of data and the fact that every 'tweet' or 'status update' would be expensive by having to pay gas for it.

Chapter 5

Alternative mining protocols

There are several other mining protocols proposed besides the PoW mining used in Bitcoin and Ethereum. Most of these protocols are designed to deal with the issue of high electricity use in Bitcoin's current PoW protocol. Recall that Ethereum tried to solve this problem by using a memory intensive PoW algorithm called Ethash, but the protocol the algorithm is built upon is still using too much electricity for it to be a long term, sustainable solution. PoS is the protocol that is currently being implemented in Ethereum with an algorithm called 'Casper' to resolve this. Because PoS is the choice of Ethereum, we will go into more details about this particular protocol in this chapter. The other proposed protocols will be briefly explained in this chapter as well, but we will not get into details about the various implementations of these since their blockchains are not as widely known and used as Ethereum.

5.1 Proof of Stake

The concept of PoS was first mentioned in a forum post [51] in 2011 and put to use in 2012 by the blockchain Peercoin. In PoS, the one who adds a new block to the chain is called a 'forger' and is chosen in a pseudo-random way [58]. As the name of the protocol suggests, to forge and add a new block to the chain you have to prove that you are staking something. What the participants are staking is a portion of their cryptocurrency and sometimes it is also taken into account how long they have been holding that currency.

Two main types of PoS designs exist, the first is a *chain-based* design that resembles PoW mining where one miner is chosen pseudorandomly amongst the stakeholders to mine the next block in the chain. The other type of design is based on *Byzantine fault tolerance* where some mathematical property can be proven given some assumptions and therefore conflicting blocks can not be finalized. One such assumption can for instance be that $\geq \frac{2}{3}$ of the participants are honest [18]. Honesty in this context means that the participants are

following the protocol as it is described. With this we achieve a network where only the correct data is broadcast. Byzantine fault tolerance is a way of achieving consensus. This means that nodes can communicate and trust that all of the nodes in the network has the same data. The expression 'Byzantine fault tolerant' has its origin in the known Byzantine Generals' Problem where one commander gives his lieutenants a message to either attack or retreat, and one (or more) of the lieutenants passes on a false message. The receiver of two different messages does not know which message to trust at this point [16]. A system that is Byzantine fault tolerant will act on the correct message, even if some of the participants are corrupt.

Ethereums PoS design follows the Byzantine fault tolerance idea, but with some modifications.

5.1.1 Casper

Ethereum has studied PoS since January 2014 and is implementing the PoS protocol with their algorithm called 'Casper'. Casper has two phases; the first is a hybrid between PoW and PoS and is called 'Casper: Friendly Finality Gadget' (FFG). The second is a pure PoS implementation called 'Casper: Correct-by-Construction' (CBC).

Casper: FFG

In this algorithm, all blocks will still be mined with PoW, but every 100^{th} block will in addition also be validated with PoS. Because of this, participants are in this algorithm called 'validators'. All mined 100^{th} blocks constitutes what is called a checkpoint tree and Casper is responsible for choosing blocks in one branch of this tree to either *justify* or *finalize*. The goal is to finalize blocks once they are justified. Casper is also called 'the son of the GHOST', or the Greedy Heaviest Observed Subtree, which is described in Chapter 4.5 and is Ethereum's way of handling stale blocks to battle centralization.

A validator who joins the set of validators has to deposit an amount of Ether, called 'stake', into a special Casper contract which is the way the algorithm is implemented. Checkpoint blocks are proposed to the validators and they cast votes with two different types of messages by interacting with the contract's vote function [19].

One of these vote messages is called a 'prepare' message which has both a source and a target. Both the source and target are checkpoint blocks, and the source has to be an ancestor of the vote target and should receive at least $\frac{2}{3}$ such prepare messages. When at least $\frac{2}{3}$ has made a prepare message for the target with the same source, the checkpoint is considered justified. In other words, a justified checkpoint has to be an ancestor of another justified

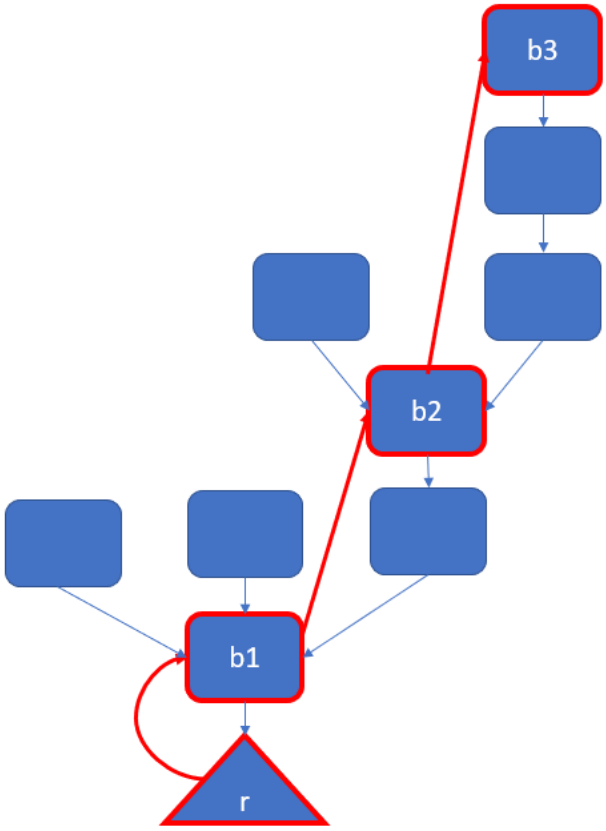


Fig. 5.1 A justified chain with four checkpoint blocks that are justified. In between each of these blocks there are 100 regular blocks. The root is denoted 'r'.

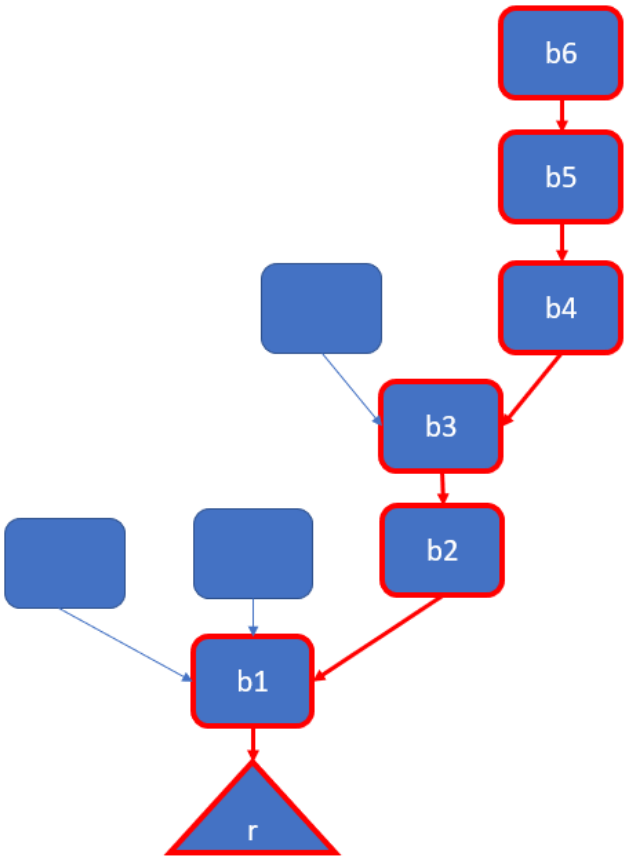


Fig. 5.2 A finalized chain with seven checkpoint blocks that are finalized. In between each of these blocks there are 100 regular blocks. The root is denoted 'r'.

checkpoint. Note that it does not have to be a direct child, only an ancestor so that they are in the same branch. Figure 5.1 illustrates this. Once a checkpoint is justified, validators send out the other type of message called a 'commit' message and once this checkpoint has received at least $\frac{2}{3}$ such messages it is considered finalized. A finalized checkpoint also has to have a direct child that is justified. This is illustrated in Figure 5.2.

These vote messages are added to the new block as transactions after the other transactions. The validator's deposit, or stake, rises and falls depending on received rewards and penalties. If a validator wants to withdraw, i.e leave the validator set, he/she has to send a 'withdraw' message to get a refund of the stake [18]. Once a validator withdraws from the set, the public key/account can not rejoin. The stake refund is delayed by four months, or 15 000 epochs which is 75 000 blocks. The reason for this delay is that there has to be enough time to discover if the leaving validator has violated any commands. These commands are called 'Slashing conditions' and penalizes the validators if they break the conditions of a valid vote. A valid vote has to come from a validator who is in the set of validators and also not violate the slashing conditions. These conditions are:

- A validator can not publish two distinct votes where the target height is the same. This means that one can not vote on different branches.
- A validator can not publish two distinct votes where the other vote is within the span of his first vote. This means that the distance of source and target of vote number two can not be within the source- target distance of vote number one.

If either of these conditions are violated, the validator's deposit gets slashed, meaning it gets deleted. The one who discovers this violation can add the evidence of this into a transaction and receive a small 'finder's fee'. Worth noticing is the fact that two conflicting checkpoints can not be finalized without at least $\frac{1}{3}$ of validators violating one of the Slashing conditions.

Casper:FFG will be implemented in what is called 'Ethereum 2.0' with other improvements to the protocol as well [30].

Casper: CBC

For now, the descriptions of Casper:CBC are abstract and its implementation is probably years into the future. It was supposed to be a full implementation of PoS for public blockchains where all blocks in the chain would be mined by the PoS protocol. The Ethereum Foundation has since made the description of it to be more general than it was originally, see [17]. It describes now how to reach consensus on blocks to make decisions on them, rather than

going into any details on what these decisions are, like for instance which node gets to forge the next block. However, it is worth knowing that this is their road to implementing PoS in the future.

5.2 Proof of Importance

Proof of Importance (PoI) is a protocol that New Economy Movement (NEM) designed. NEM is a blockchain that launched in March 2015. Behind this protocol lies the idea that other factors than a miner's wealth in coins is of importance to the network. These factors must of course also encourage him/her to not cheat the network. The mining process is here called 'harvesting' and the harvester is awarded the transactions fees belonging to their harvested block. Harvesters are chosen not only based on the stake they put up, like in PoS, it depends also on other factors like net transfer and how clustered the node in question is [48]. Net transfer is the number and size of transactions that the harvester has performed over the period of the last 30 days. There is also a requirement that the harvesters account has to hold at least 10 000 coins for a certain number of days.

What stands out about this protocol is the clustering factor which is implemented with Structural Clustering Algorithm for Networks (SCAN). NEMs' implementation defines nodes that are called 'cores' of which the clustering is centered around. Transactions with a total sum over 1 000 coins between nodes are seen as edges in this graph [49]. Core nodes are defined as having a certain number of neighbours, i.e it has to have sent and received transactions with many other participants and by this proving to be an active part of the network. To be clustered around such a node means that this node also is an active and hence important participant that contributes positively to the network and therefore can be trusted to a higher degree. In Figure 5.3 we see an example of a small graph with four clusters found by the SCAN algorithm [52]. Each of the clusters are presented in the illustration with different shapes and colors.

5.3 Proof of Burn

As the name suggests the miner has to provide a proof that something is 'burned'. This means that the miner has to prove that he or she has burned some cryptocurrency, which is the act of sending it to a special address called the 'Eater address'. This special address does not have a corresponding private key as it is generated randomly without the private key part being generated. When a user sends coins to this address the proof is made public because it is recorded in the blockchain that it is in fact sent to said address.

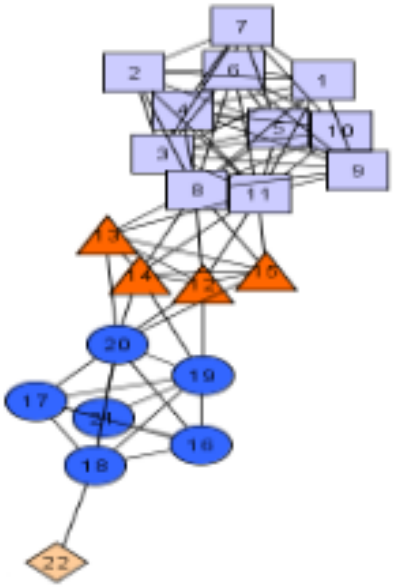


Fig. 5.3 An example of clustering with the SCAN algorithm [52].

The coins being sent is an amount of the transaction fees for mining, which results in the miner sharing the reward with the rest of the network since burning coins reduces the total amount of coins and thereby increases the value of the currency [45]. The more coins a user burns, the higher probability that the user gets to mine the next block. The blockchain Slimcoin uses this protocol and the miner of the next block is determined by a score called 'Effective Burnt Coins' [53].

What this protocol does is essentially going into the core of the idea that a miner has to do something expensive like investing in expensive equipment to do a lot of computation, to being exactly something expensive in its clearest form. In this way, the economy factor is not disguised or simulated as in other protocols like PoW, but it is out in the open that one has to invest in spending (burning) coins to earn more coins. With this stripped down version of investment, there is no need to use excessive electricity to prove that you have invested because this is obvious already.

5.4 Proof of Capacity

The concept of Proof of Capacity was first proposed in 2013 and was then referred to as 'Proof of Space' [50]. The idea behind this protocol is that the user has to give a proof that they have space, i.e capacity on their hard drive for solutions to the hashing algorithm.

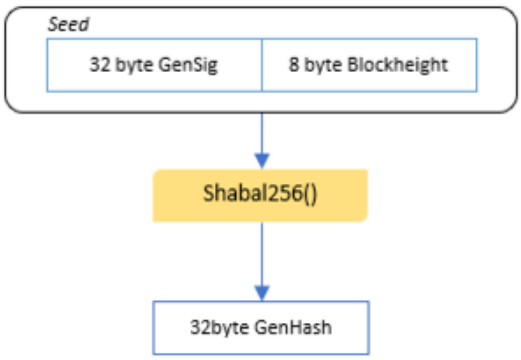


Fig. 5.4 The generation hash for a new block in Burstcoin [15], a Proof of Capacity blockchain.

The block time is relatively short, so everyone who wants to mine a block has to have pre-computed hash values stored on their hard drive. This means that the more solutions, called 'plots', a user has stored, the higher the probability that he or she has the correct solution and gets to mine the next block. The hash used is called Shabal256 and it can take days or even weeks to make the plot files [46]. The solutions one stores are the nonces and each nonce contains 8192 hashes stored in pairs called 'scoops'. There are 4096 scoops, and they are assigned a number ranging from 0 to 4095.

In the mining process you calculate a scoop number and uses the data in your own files of that scoop to calculate a deadline; an amount of time. This process is repeated for all the pre-computed nonces. The minimum of the deadlines are chosen and this represents the amount of time that must have passed since the last block was mined before the miner is allowed to add the current block to the chain. If no one else mines a new block within this deadline, the user can add the block to the chain and be rewarded for the mining. Burstcoin is the one blockchain that uses this protocol.

With the concept of pre-computing, how can you then ensure that the blockchain is in its correct order? Let us have a look at some of the details about the mining process [15]. Block mining is in this protocol called 'forging' and the participant (account) who forged it is called 'Block Generator'. To forge a block one needs a 32 byte long 'Generation Signature' which is made up of the previous generation signature and the previous block generator, in addition to the new block height and its base target. Base target adjusts the difficulty and is calculated from the previous 24 blocks in the chain and is adjusted so that one block is mined about every 4 minutes. The generation signature, block height and base target together is called a 'seed' and is hashed with Shabal256 to get a result called 'Generation hash'. This is pictured in Figure 5.4. The generation hash is a 32 byte long hash value that will be reduces

modulo 4096 to retrieve a scoop number. Now, all scoops from all stored nonces will be read individually and used as input together with the new generation signature in Shabal256 and its output is called a 'Target'. Target is divided by the base target to compute the deadline.

The wallet is acting as a timer and checks if the seconds defined by the deadline has passed since last block was forged. If another block is broadcast on the network before the deadline expires, the mining information the wallet has received is discarded. If not, the wallet will start forging a block and adding all unconfirmed transactions it has received from the network into it until the limit of 255 transactions is reached. Each transaction will also be validated before it is added to the new block. The new block will not contain the transactions in itself, but the ID of the transactions and a Shalbal256 hash value of all of them. Transactions are stored separately.

Proof of Capacity can be considered a PoW algorithm that is memory intensive. This is in contrast to Bitcoins' PoW algorithm Hashcash which is processor intensive. This means that it resembles Ethereums solution to avoid ASIC mining, but yet it differs by being more environmental due to the fact that the pre-computation is done in advance, and only once.

5.5 Proof of Activity (hybrid)

This protocol is a hybrid of PoW and PoS. This means that the mining of the block is done by PoW; miners compete to mine the next block by solving a puzzle. The blocks do not contain any transactions, but should be concidered a dummy block with only blockheaders and the address for the block reward. When it comes to choosing who's block is added to the chain and granted the reward, the protocol switches to PoS instead of the original PoW which chooses the block that is mined and broadcast first.

A group of random validators validate or sign the new block. These validators are chosen based on the blockheader of the new dummy block and the probability of being chosen is based on how many coins they have staked. When the new transaction block is signed by all of the validators, it is added to the chain and transactions can be added to it at this stage. If some of the validators are not available to sign the new block for some reason, the next mined dummy block is chosen and based on its blockheader a new set of validators is chosen. This then becomes the block that is added to the chain [44].

The reason it is called Proof of Activity is that the validators has to be available (active) to validate the new block, if they are not, someone else is chosen. The reward is split between the miner who mined the new block and the validators of it. One blockchain that uses this protocol is Decred which started in 2016.

5.6 Proof of Checkpoint (hybrid)

This is another hybrid of PoS and PoW. A certain number of blocks are mined by PoS and then a block is mined by PoW, called a 'checkpoint block'. This checkpoint block does not contain any transactions, but is needed to verify the blocks mined before it by PoS [47]. It is assumed that the PoW miner checks that all transactions are are valid in all blocks since the last checkpoint. The PoW miners may, hopefully, be different from the PoS miners, thus adding extra security to a PoS system. The checkpoints are linked both to the blocks that came before them and the blocks that come after, even though they are mined by different protocols.

5.7 Proof of Elapsed Time

Intel came up with this protocol. It is not yet in use in any blockchains, but the concept is still worth mentioning as it is an interesting idea. The mining process is similar to PoW, but the miner is chosen based on waiting time, resembling Proof of Capacity [42]. Each miner has a timer which expires and the one that expires first is either the winner and gets to mine the next block, or this miner's block is chosen of all the mined blocks if the implementation is more similar to PoW. The implementation can vary, but the basic idea can be compared to a lottery where every miner has the same chance at being picked and reaping the reward.

Chapter 6

Discussion

We will now have a look at different benefits, new possibilities, limitations and challenges with the new proposed mining protocols. This discussion will have its main focus on sustainability, but other issues such as centralization will also be pointed out. Some protocols also introduce other types of attacks than those mentioned in Chapter 2.3, these will also be presented. The goal is to find one or more protocols that are more sustainable than the current dominant Proof-of-work protocol, with wider, long-term use of the blockchain technology. We will start by looking at different advantages of the various protocols, and thereafter look at disadvantages before we make a conclusion of which protocol we believe is best suited in a long-term perspective.

6.1 Advantages of the different protocols

The main advantage we are looking for is sustainability, but there is also the possibility that these new proposed protocols have some other advantages as well. When it comes to sustainability, all the new proposed protocols except Proof of Activity is a big improvement upon the PoW protocol. In these protocols the miner is chosen in advance, and thereby minimizing the electricity use to only concern the miner.

PoI has another advantage as well, namely considering other factors than wealth to give a participant a score and the privilege to mine. These additional factors do have an implication that one has to be a 'wealthy' participant since it is based on transactions with a certain minimal sum, but it also means that one has to actually use the network and be an active part of it, which I think is a good idea. In my opinion, the idea of bringing in other factors to give a total Importance-score is a fairer way than only relying on the stake that is put up as is the case of the PoS protocol. If this protocol would expand its factors so that other, not as wealthy participants also could get mining privileges, this would be my recommended

protocol. The big question is what these factors should be since they have to give incentives that prevent people from cheating the network by making forks on the block chain.

Perhaps in the case of Ethereum one could have a factor of creating smart contracts, i.e the ones making decentralized applications on the platform would get some points in their total score for each contract they own, maybe with some point calculations on how complex the contracts are and how many people interact with it. PoI has a lot of potential of being both a sustainable *and* a truly decentralized protocol if the factors would extend to something else rather than just wealth.

Something that a new protocol should provide is decentralization and here Proof of Activity (in the case of a memory intensive algorithm) and Proof of Capacity stand out because mining is in the scope of the common user with these protocols. As mentioned, Proof of Importance also has the potential to become more decentralized, but based on its factors as of today, these two other protocols are better at this point.

Proof of Elapsed Time has a fairness that is its main advantage because of the timer that each participant has. This timer does not take into account how wealthy you are, neither in the sense of how many coins you own nor what kind of equipment you have.

6.2 Disadvantages of the different protocols

PoS is the preferred protocol to implement in the second largest public blockchain Ethereum. However, it does have a few challenges. One of them is the potential to make the blockchain become more centralized and another is that it makes three new kinds of attacks possible. The first of these attacks is called 'Nothing at stake', the second 'Long range division' and the third 'Catastrophic crashes'. The seriousness of these challenges all depend on the implementation of the protocol. Ethereum's Casper implementation has taken these new attacks into account and are dealing with most of them by using Slashing conditions and fork-choice rules. Some problems have not yet been solved by a specific algorithm in Casper:FFG, but will have a solution in Casper:CBC. Other PoS implementations also have to tackle these attacks which have proven to be a challenge

'Nothing at stake' is the notion that a miner can mine on several chains in case of forks, i.e behaving badly, and not lose anything as a result of this behaviour. Instead, the miner has everything to gain, because he can mine on every fork and thereby get a reward no matter what fork wins and becomes the true chain.

'Long range division' is the possibility that a coalition of validators withdraw and their deposit equals $\geq \frac{2}{3}$ of a *historical* deposit. They can now in theory finalize conflicting

checkpoints without getting slashed because they already have withdrawn so they have nothing to lose by violating the conditions.

'Catastrophic crashes' happens if $\geq \frac{1}{3}$ of validators crashes at the same time. This means that no future checkpoints can be finalized. This protocol gets criticized for its centralization issue where the rich are getting richer by forcing miners to stake coins of a certain sum, often a fairly high sum as well, and by this earning the privilege to mine and hereby becoming even more wealthy. This means that the network is run not by the common user but by the wealthy miner. This issue is essentially the same as for processor intensive PoW mining where the mining is centralized to those who can buy the expensive hardware that is required.

Proof of Importance has, as mentioned, the advantage that it also weighs in other factors than wealth in coins when giving participants mining privileges. Nevertheless, as of today this protocol still favors participants with wealth and the means to do a high number of expensive transactions, so the centralization is also an issue here. This protocol also has the attack issues of 'Nothing at stake', 'Long range division' and 'Catastrophic crashes' like the PoS protocol does since it is in a sense a PoS extension.

Proof of Burn requires miners to waste coins to increase their chances to mine the next block, which means that miners are selected in advance and do not have to race to finish first as in PoW. This is an advantage because it does not waste electricity in the mining process, but it raises another issue which is centralization of the mining process. Like PoW where miners invest in ASICs to mine faster, miners have to invest, or in this case burn, coins to get the privilege to mine. I would say this fact raises the issue about centralization, but the benefit is that it at least seems more sustainable than PoW mining. With that being said, as coins always gets burned in order to mine, this protocol would make the prices of the currency artificially high over time. This implies that the ones who have coins to burn in order to make even more coins, would definitely be getting even richer when they themselves are the ones pushing the price up.

Proof of Capacity has such short block time that miners must pre-compute hash values and store them on their computer. This protocol consumes disk space and not CPU power, but the hash values still has to be computed ahead of time which can take days or even weeks. When every miner does this, even only once, there will be a lot of electricity use combined. A wide use of this protocol would therefore still consume too much electricity to be sustainable in my opinion.

Proof of Capacity resembles the mindset of PoW mining where people are stocking up on ASIC CPUs, only in this case people would instead stock up on hard discs. It would still be a race to have the most and best hardware to mine so it would in this sense be about equally

expensive to mine. Also, if the computers are constantly searching through their data this would generate a high total amount of electricity use.

Proof of Activity can not be said to be either sustainable or decentralized as it has both PoWs energy waste and PoS centralization issues to deal with. In addition to being chosen as a validator by staking coins, one also has to be active at the time they are chosen to validate a block, this means that they also are forced to be online in case they are chosen. With this protocol a lot of blocks are still mined and just discarded if the nodes to validate them are not active leading to certain electricity waste. Also, a lot of validators have to be online even though they may not be picked for a while. In addition, the validators are chosen to be amongst those who hold a certain minimum amount of coins which is a centralization of the system. As with PoS and Proof of Importance, the issues with 'Nothing at stake', 'Long range division' and 'Catastrophic crashes' is also a disadvantage for this protocol.

Another hybrid is Proof of Checkpoint, but in this protocol most of the mining is done by the PoS protocol and only checkpoints are mined by PoW. This means that the energy waste is a lot less than with pure PoW, but it still suffers from the issue of centralization depending on the implementation of PoS. Also, one could argue that those who gets to mine the checkpoints would have to invest in expensive hardware in addition to putting up stakes to mine regular blocks. This would mean that the centralization would be even heavier in this protocol, but this would depend on whether the same miners mine both types of blocks or if the mining privileges are divided amongst participants. Since PoS is a major part of Proof of Checkpoint, 'Nothing at stake', 'Long range division' and 'Catastrophic crashes' also proposes a threat.

Proof of Elapsed Time is not yet implemented in any blockchains, so this protocol is somewhat difficult to assess. Nevertheless, the idea that everyone in the network can mine blocks based on a timer is very true to the decentralization idea of blockchains. The problem with centralization still arises because of the suggested use of Intels' own technology, Software Guard Extension (SGX), as a way of generating random waiting times. To use SGX for security and not the joint consensus rules that are backed up by cryptography does take away the point of using a blockchain in the first place, so this particular protocol would not be suited for a public blockchain.

A solution to this may be to implement a timer and generation of random waiting times in a smart contract, but the contract languages does not come with a random number function. The reason for this is that the system in itself is deterministic and should not include such randomness. If you were to implement such a function on your own, you could not use values like the blocks' timestamp or other 'random' values because these are values that miners

could in theory manipulate. So until a decentralized method for implementing this protocol is proposed, I would not recommend this as a new mining protocol for public blockchains.

As we can see, essentially all of these protocols suffer from the problem with centralization and hence the favoritism of wealthy participants of the network. Some protocols also suffer from other issues like energy waste, as in the case with Proof of Capacity and Proof of Activity. Also, three other types of attacks are made possible by all protocols that has something to do with PoS. These are the previously mentioned 'Nothing at Stake', 'Long range division' and 'Catastrophic crashes' scenarios.

6.3 Conclusion

Dismissing all other issues than sustainability, I would recommend all of these protocols over PoW, except for Proof of Activity. If we look at other issues and an overall assessment, I would recommend Proof of Importance as a new mining protocol for public blockchains. The reason for this is that not only does it resemble PoS which is already accepted of The Ethereum Foundation as their new protocol, but it also improves upon the concept of PoS by bringing in other factors than wealth. The protocol also has the potential of solving its issues like centralization by adding other types of factors into its calculations. By using Proof of Importance the electricity use would go substantially down in comparison to the current use of the PoW protocol and it would give the blockchain network the possibility to bring in other factors at a later stage by expanding it.

Ethereum is now switching to PoS by first implementing Casper:FFG and at a later stage by implementing Casper:CBC which is a full PoS implementation. PoS is the best alternative as of today when we think of the factors that Proof of Importance *does* factor in and do not take into account that other and better factors could be used. Despite this fact, I will still recommend PoI because of its ability to adapt and expand in the future. When making implementation choices in relatively new technology, one should, in my opinion, make room for improvements that are not yet thought of. PoI is the only new proposed mining protocol for block chains that does offer this particular possibility.

The concept of decentralization is difficult and it seems as if one has to accept the solution that has the lowest rate of centralization. PoI is also somewhat centralized, but because of the multiple factors, and the possibility for even more of them, this seems like the less centralized protocol. Maybe something to consider as a factor could be the number of transactions rather than the amount of the transactions. The new suggestions for additional factors should be voted upon among the participants of the network, and the new suggestion could be included in the Importance score if it had at least $\frac{2}{3}$ of the votes.

One last thing to take into consideration is the possibility of making a protocol that 'punishes' participants for centralization and rewards those who act decentralized. All mining protocols are proposed with the mindset that one has to be the best in something or have the most of something, which always pushes the system into a centralized direction. Would it be possible to come up with a protocol that gives the mining privileges to those who are not the best, nor has the most, but instead is the most average?

The pillar in blockchain mining is that miners has to do something that in some way is considered expensive, be it using assets or doing hard computations, to prevent them from cheating the system and make sure they act as honest participants. Can we somehow overcome this black and white notion that this is the only thing that would make participants act honest? It should be clear that one has to do *something* to get to mine the next block. It has to be a real participant of the system and not just someone who joins for the sole purpose of getting to decide which transactions are the 'truth' and not contributing to the blockchain network in any other way. Could it perhaps be enough to be an active node? One thought that comes to mind is that a miner would have to be on average active in the system over an extended period of time. If both time and participation would be the 'something expensive' thing to invest instead of assets or investment in hardware (which also comes down to assets), my guess is that attackers would not find it worthwhile to attack the system.

Appendix A

Contract oriented programming languages

One of Ethereum's contract-oriented, Turing complete language is called Solidity [54]. It is developed for the sole purpose of implementing smart contracts and is inspired by Javascript, C++ and Python. EVM is installed on every node in the network and compiles and runs the contract code. The specifications for EVM are described in the Ethereum Yellow Paper [31]. The code examples that will be discussed were programmed in the fall of 2017 as part of a programming project. It is my own work with guidance and help from www.ethereum.org and www.ethereum.stackexchange.com.

A.1 Solidity

There are several integrations available. Remix is the integration of choice since it is a part of the Ethereum browser Mist (extended developer version) so it is easily accessible by downloading Mist. Users that do not want to create their own contract can make good use of the same browser in a different version: Ethereum wallet. We will now look at a few examples to demonstrate the use of smart contracts and see how these are deployed and interacted with in practice.

Contract code

In the following section we will look at two different smart contracts; one called Kickstarter to raise funds for a project and one called WebsiteToken that creates tokens and interacts with the first contract. With these two examples we will cover the basics of the language and also show off some of the smart contracts' greatest potential.

```
pragma solidity ^0.4.6;
contract KickStarter {
  address owner;
  uint public fundingGoal;
  uint public fundsRaised;
  uint public deadline;
  bool fundingGoalReahced = false;
  bool kickStarterClosed = false;
  uint number;
  address[] _giver;
  mapping (address => uint256)  public givenFunds;
  event GoalReached(address recipient, uint toalAmountRaised);
  event FundTransfer(address backer, uint amount, bool isContributer)
      ;

 function KickStarter(uint fundingGoalInEther, uint durationInMinutes
     ) public {
 owner = msg.sender;
 fundingGoal = fundingGoalInEther * 1 ether;
  deadline = now + durationInMinutes * 1 minutes;
    }

 function donate() public payable {
  if (msg.value == 0) {
        revert();
        }

        require(!kickStarterClosed);
         number = msg.value;

         givenFunds[msg.sender] += number;
         fundsRaised += number;
         FundTransfer(msg.sender, number, true);

         _giver.push(msg.sender);
    }

    modifier afterDeadline(){
        if (now >= deadline) _;
    }

 function checkGoalReached() public afterDeadline{
        if(fundsRaised >= fundingGoal){
            fundingGoalReahced = true;
```

```
43              GoalReached(owner, fundsRaised);
44          }
45          kickStarterClosed = true;
46      }
47
48  function moveFund() public afterDeadline{
49          if(!fundingGoalReahced){
50              uint amount = givenFunds[msg.sender];
51              givenFunds[msg.sender] = 0;
52              if (amount > 0){
53                  msg.sender.transfer(amount);
54                 FundTransfer(msg.sender, amount, false);
55              }
56          }
57
58          if(fundingGoalReahced && owner == msg.sender){
59                    owner.transfer(fundsRaised);
60                     FundTransfer(owner, fundsRaised, false);
61
62          }
63      }
64
65  function getGiver(uint256 i) constant returns (address ){
66          return _giver[i];
67  }
68
69  function getNumberOfGivers() constant returns (uint ){
70    return _giver.length;
71  }
72
73  function() payable{
74  }
75 }
```

```
1  pragma solidity ^0.4.6;
2  import "./KickStarter.sol";
3
4  contract WebsiteToken {
5    KickStarter kick;
6    mapping (address => uint256) public balanceOf;
7    string public name;
8    string public symbol;
9    uint8 public decimals;
```

```
10   event Transfer(address indexed from, address indexed to, uint256
         value);
11         address owner;
12         uint public number;
13         modifier onlyowner {
14             if (msg.sender == owner) _;
15             }
16
17 function WebsiteToken()public {
18     balanceOf[msg.sender] = 1000;
19     name = "WebAccess";
20     symbol = "$";
21     decimals = 0;
22     owner = msg.sender;
23     kick = KickStarter(0x257e7a41f384c1823f53d2f304d32eb212cd7394);
24       for(uint256 i = 0; i < kick.getNumberOfGivers(); i++){
25           balanceOf[kick.getGiver(i)] = 1;
26       }
27       balanceOf[owner] -= kick.getNumberOfGivers();
28    }
29
30 function transfer(address _to, uint256 _value) public {
31 require(balanceOf[msg.sender] >= _value && balanceOf[_to] + _value
       >= balanceOf[_to]);
32
33   balanceOf[msg.sender] -= _value;
34   balanceOf[_to] += _value;
35
36      Transfer(msg.sender, _to, _value);
37        }
38
39 function buy() public payable{
40      if (msg.value == 0) {
41        revert();
42        }
43       number = uint256(msg.value);
44        require(balanceOf[owner] >= number/2000000000000000000 &&
              number >= 2000000000000000000);
45        balanceOf[owner] -= number/2000000000000000000;
46        balanceOf[msg.sender] += number/2000000000000000000;
47
48       Transfer(owner, msg.sender, number/2000000000000000000);
49   }
50
```

```
51  function moveFunds() public onlyowner{
52      owner.transfer(this.balance);
53   }
54
55  function() payable{
56      }
57  }
```

Code review

A few of Solidity's features include events, modifiers, message objects, mapping, heritage and suffixes.

Events are used to store arguments in the transaction log and Javascript can listen to these events. Two examples are found in the KickStarter contract; GoalReached and FundTransfer. Both of these events are triggered if certain conditions are fulfilled. The same is true for the contract WebsiteToken where an event Transfer is triggered if its conditions are true.

Modifiers can be added to functions such as the modifier 'payable' which states that the function can receive Ether. Every contract should have a function called a 'fallback function' that is nameless and payable, so that the contract is able to receive and hold Ether even though it might not be programmed to receive and do anything with its received Ether. Another modifier is 'onlyowner' which allows only the owner of the contract to use the function. This modification of the function is achieved by making sure that the address that sent the contract's constructor into the blockchain is the same that is interacting with the function.

Message objects hold information about the transaction. The function 'msg.sender' holds the address to the one who sent the transaction and the function 'msg.value' is the Ether value sent with the transaction.

Mapping can be viewed as a Hash table. In the contract KickStarter there is the mapping from addresses to uint256 where the addresses and their donation are stored.

Suffixes as 'seconds', 'minutes', 'hours', 'days', 'weeks' and 'years' are a part of the language in addition to 'now' which gives the blocks' timestamp. These suffixes are very helpful when programming a contract that has some sort of time limit. There are also suffixes for 'wei', 'finney', 'szabo', 'ether' and so on.

Contracts can interact with each other through heritage as with other known programming languages, but this works differently in Solidity where one needs the other contract's address. So the first contract needs to be on the blockchain and then an instance of this contract can be added as we see in the contract code for WebsiteToken.

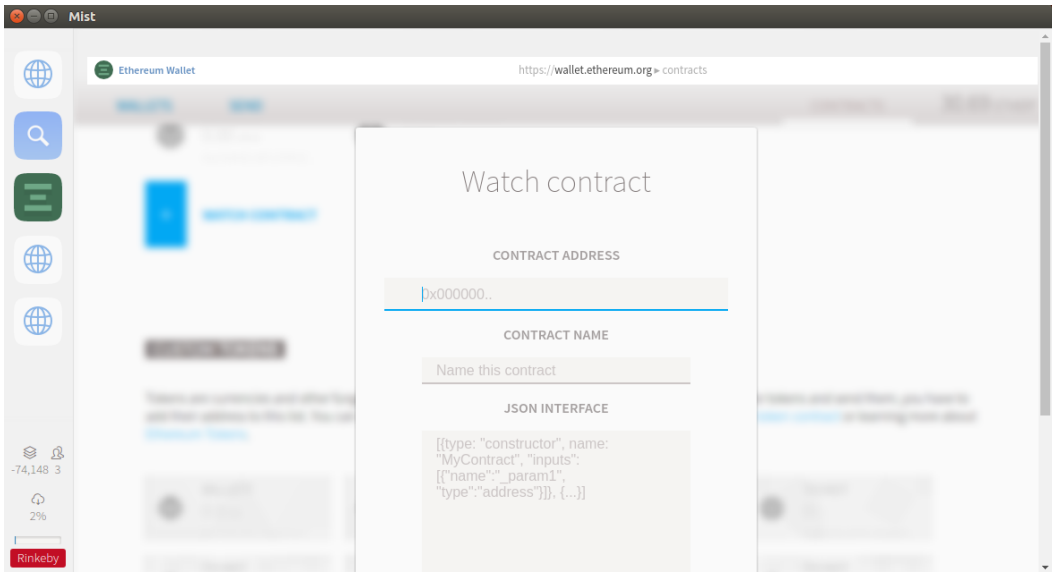


Fig. A.1 In your wallet you can watch contracts and interact with them as long as you have its address.

Transaction

Running the code in the contract creates a transaction in the blockchain and gives the contract an address. Once it is put on the blockchain it can not be changed. Every interaction with its functions are also transactions.

Interacting with and viewing a contract in Ethereum

Once you have a contract's address you can view it and interact with it in your account view in your wallet. It is not necessary to create a website/frontend, but this depends on the contract owner's needs.

Front end

Within a script tag for Javascript one can reach the contract code with HTML and interact with its functions. Someone wanting to make a front end/website for their smart contract has to include some specifications for their contract in their HTML code. These specifications are an address to the contract itself and some variables that one can find easily in Mist browser after the contract is sent as a transaction to the blockchain. A Web3 object also has to be included to reach the contract code; this object has several such useful functions. An example of this follows from part of the front-end code of the contracts Kickstarter and WebsiteToken.

```
<script type="text/javascript" src="web3.js"></script>
<script type="text/javascript">

 function donate () {
    var donation = contractspec.at('0
       x7c3d19891376553279b822aee6d75644942a29be')
    donation.donate({
        from: document.getElementById('fromGive').value,
        value: (document.getElementById('valueGive').value *
           1000000000000000000)},
        function (error, txHash) {
        tryTillResponse(txHash, function (error, receipt) {
            alert('done ' + txHash)
        })
    })
}

 var Web3 = require('web3')

  if (typeof web3 !== 'undefined') {
  web3 = new Web3(web3.currentProvider);
} else {
  web3 = new Web3(new Web3.providers.HttpProvider("http://
     localhost:8545"));
}
 var contractspec = web3.eth.contract([{"constant":false,"
    inputs":[],"name":"checkGoalReached","outputs":[],"
    payable":false,"stateMutability":"nonpayable","type":"
    function"},{"constant":true,"inputs":[],"name":"deadline"
    ,"outputs":[{"name":"","type":"uint256"}],"payable":false
    ,"stateMutability":"view","type":"function"},{"constant":
    false,"inputs":[],"name":"moveFund","outputs":[],"payable
    ":false,"stateMutability":"nonpayable","type":"function"
    },{"constant":true,"inputs":[],"name":"getNumberOfGivers"
    ,"outputs":[{"name":"","type":"uint256"}],"payable":false
    ,"stateMutability":"view","type":"function"},{"constant":
    true,"inputs":[],"name":"fundsRaised","outputs":[{"name":
    "","type":"uint256"}],"payable":false,"stateMutability":"
    view","type":"function"},{"constant":true,"inputs":[],"
```

```
    name":"fundingGoal","outputs":[{"name":"","type":"uint256
    "}],"payable":false,"stateMutability":"view","type":"
    function"},{"constant":true,"inputs":[{"name":"","type":"
    address"}],"name":"givenFunds","outputs":[{"name":"","
    type":"uint256"}],"payable":false,"stateMutability":"view
    ","type":"function"},{"constant":false,"inputs":[],"name"
    :"donate","outputs":[],"payable":true,"stateMutability":"
    payable","type":"function"},{"constant":true,"inputs":[{"
    name":"i","type":"uint256"}],"name":"getGiver","outputs"
    :[{"name":"","type":"address"}],"payable":false,"
    stateMutability":"view","type":"function"},{"inputs":[{"
    name":"fundingGoalInEther","type":"uint256"},{"name":"
    durationInMinutes","type":"uint256"}],"payable":false,"
    stateMutability":"nonpayable","type":"constructor"},{"
    payable":true,"stateMutability":"payable","type":"
    fallback"},{"anonymous":false,"inputs":[{"indexed":false,
    "name":"recipient","type":"address"},{"indexed":false,"
    name":"toalAmountRaised","type":"uint256"}],"name":"
    GoalReached","type":"event"},{"anonymous":false,"inputs"
    :[{"indexed":false,"name":"backer","type":"address"},{"
    indexed":false,"name":"amount","type":"uint256"},{"
    indexed":false,"name":"isContributer","type":"bool"}],"
    name":"FundTransfer","type":"event"}]);
</script>
```

In this code we have the function donate() from the Kickstarter contract with the contract's address as a variable and its specifications as another variable. Web3 is the Ethereum Javascript API that allows the interaction with a node using HTTP and as we can see it is used to fetch the contract's specifications.

References

[1] 51% attack. https://learncryptography.com/cryptocurrency/51-attack. Accessed: 2019-03-14.

[2] ARPANET. https://interestingengineering.com/arpanet-or-how-the-internet-was-born. Accessed: 2019-03-14.

[3] Bitcoin addresses. https://en.bitcoin.it/wiki/Technical_background_of_version_1_Bitcoin_addresses. Accessed: 2019-03-14.

[4] Bitcoin electricity. https://powercompare.co.uk/bitcoin/. Accessed: 2019-03-14.

[5] Bitcoin mining. https://en.bitcoin.it/wiki/Mining. Accessed: 2019-03-14.

[6] Bitcoin mining pools. https://www.buybitcoinworldwide.com/mining/hardware/. Accessed: 2019-03-14.

[7] Bitcoin price charts. https://coinmarketcap.com/currencies/bitcoin/. Accessed: 2019-03-14.

[8] Bitcoin target. https://en.bitcoin.it/wiki/Target. Accessed: 2019-03-14.

[9] Bitcoin transaction fee. https://support.earn.com/digital-currency/bitcoin-transactions-and-fees/how-do-i-calculate-my-transaction-fee. Accessed: 2019-03-14.

[10] Bitcoin Whitepaper. https://www.bitcoin.com/bitcoin.pdf. Accessed: 2019-03-14.

[11] Bitmain. https://en.wikipedia.org/wiki/Bitmain. Accessed: 2019-03-14.

[12] Blockchain info. https://blockchain.info/. Accessed: 2019-03-14.

[13] Bloom filter discussion. https://ethereum.stackexchange.com/questions/3418/how-does-ethereum-make-use-of-bloom-filters. Accessed: 2019-03-14.

[14] BTC units. https://en.bitcoin.it/wiki/Units. Accessed: 2019-03-14.

[15] Burstcoin details. https://burstwiki.org/wiki/Technical_information_about_mining_and_block_forging. Accessed: 2019-03-14.

[16] Byzantine General Problem. https://medium.com/all-things-ledger/the-byzantine-generals-problem-168553f31480. Accessed: 2019-03-14.

[17] Casper CBC. https://github.com/cbc-casper/cbc-casper-paper/blob/master/cbc-casper-paper-draft.pdf. Accessed: 2019-03-14.

[18] Casper FFG algorithm. https://github.com/ethereum/research/blob/master/papers/casper-basics/casper_basics.pdf. Accessed: 2019-03-14.

[19] Casper notes. https://vitalik.ca/files/casper_note.html. Accessed: 2019-03-14.

[20] Charts. https://www.blockchain.com/charts. Accessed: 2019-03-14.

[21] CPUGPU. https://en.bitcoin.it/wiki/Why_a_GPU_mines_faster_than_a_CPU. Accessed: 2019-03-14.

[22] Cryptographic primitives. https://en.wikipedia.org/wiki/Cryptographic_primitive. Accessed: 2019-03-14.

[23] ECDSA. https://en.bitcoin.it/wiki/Elliptic_Curve_Digital_Signature_Algorithm. Accessed: 2019-03-14.

[24] Electricity prices. https://www.globalpetrolprices.com/China/electricity_prices/. Accessed: 2019-03-14.

[25] Electricity production in China. https://www.eia.gov/todayinenergy/detail.php?id=33092. Accessed: 2019-03-14.

[26] Etereum prices. https://www.coinbase.com/charts. Accessed: 2019-03-14.

[27] ETH units. http://ethdocs.org/en/latest/ether.html. Accessed: 2019-03-14.

[28] Ethash algorithm. https://github.com/ethereum/wiki/wiki/Dagger-Hashimoto. Accessed: 2019-03-14.

[29] Ether. https://www.ethereum.org/ether. Accessed: 2019-03-14.

[30] Ethereum 2. https://www.coinspeaker.com/ethereum-roadmap-unveiled/. Accessed: 2019-03-14.

[31] Ethereum: A secure decentralised generalised transaction ledger, Ethereum Project Yellow Paper. https://ethereum.github.io/yellowpaper/paper.pdf. Accessed: 2019-03-14.

[32] Ethereum gas. https://hackernoon.com/ether-purchase-power-df40a38c5a2f. Accessed: 2019-03-14.

[33] Ethereum light clients. https://github.com/ethereum/wiki/wiki/Light-client-protocol. Accessed: 2019-03-14.

[34] Ethereum white paper. https://github.com/ethereum/wiki/wiki/White-Paper. Accessed: 2019-03-14.

[35] EVM. https://github.com/ethereum/wiki/wiki/Ethereum-Development-Tutorial. Accessed: 2019-03-14.

[36] Ghost protocol. https://delegatecall.com/questions/ethereum-ghost-protocol-9468d1f8-c656-4e64-8b83-80e1791c53d1. Accessed: 2019-03-14.

[37] Hashcash algorithm. https://en.wikipedia.org/wiki/Hashcash. Accessed: 2019-03-14.

[38] History of the internet. https://en.wikipedia.org/wiki/History_of_the_Internet. Accessed: 2019-03-14.

[39] Internet hosts. https://www.statista.com/statistics/264473/number-of-internet-hosts-in-the-domain-name-system/. Accessed: 2019-03-14.

[40] Krypton attack. https://bitcoinexchangeguide.com/krypton-mine-ico-kmt-token/. Accessed: 2019-03-14.

[41] Krypton Ddos. https://www.ccn.com/krypton-ethereum-bitcoin-proof-of-stake-blockchain-after-51-att
Accessed: 2019-03-14.

[42] Mining protocols. https://arxiv.org/pdf/1809.05613.pdf. Accessed: 2019-03-14.

[43] Patricia tree. https://github.com/ethereum/wiki/wiki/Patricia-Tree. Accessed: 2019-03-14.

[44] Proof of Activity basics. https://www.investopedia.com/terms/p/proof-activity-cryptocurrency.asp. Accessed: 2019-03-14.

[45] Proof of Burn basics. https://99bitcoins.com/what-is-proof-of-burn/. Accessed: 2019-03-14.

[46] Proof of Capacity basics. https://coincentral.com/what-is-proof-of-capacity/. Accessed: 2019-03-14.

[47] Proof of Checkpoint basics. https://steemit.com/cryptocurrency/@killjoy/the-different-proofs-of-crypto-currency. Accessed: 2019-03-14.

[48] Proof of Importance. https://www.smithandcrown.com/definition/proof-of-importance/. Accessed: 2019-03-14.

[49] Proof of Importance Whitepaper. https://nem.io/wp-content/themes/nem/files/NEM_techRef.pdf. Accessed: 2019-03-14.

[50] Proof of Space Whitepaper. https://eprint.iacr.org/2013/796.pdf. Accessed: 2019-03-14.

[51] Proof of Stake forum post. https://bitcointalk.org/index.php?topic=27787.0. Accessed: 2019-03-14.

[52] SCAN research paper. http://www1.se.cuhk.edu.hk/~hcheng/seg5010/slides/p824-xu.pdf. Accessed: 2019-03-14.

[53] Slimcoin. http://slimco.in/. Accessed: 2019-03-14.

[54] Solidity documentation. http://solidity.readthedocs.io/en/v0.4.24/. Accessed: 2019-03-14.

[55] State trie. https://ethereum.stackexchange.com/questions/268/ethereum-block-architecture. Accessed: 2019-03-14.

[56] TCP/IP Layers. http://microchipdeveloper.com/tcpip:tcp-ip-five-layer-model. Accessed: 2019-03-14.

[57] TLS. https://devcentral.f5.com/articles/ssl-profiles-part-8-client-authentication. Accessed: 2019-03-14.

[58] What is Proof of Stake. https://medium.com/nakamo-to/what-is-proof-of-stake-pos-479a04581f3a. Accessed: 2019-03-14.