Contents

Introduction

Despite Bitcoin's growing awareness, most people still do not know and understand the properties of this new system and how it is able to develop and sustain itself completely independently. At a first approach scepticism usually prevails, one cannot conceive and believe in the existence of a new monetary model completely governed by algorithms and encryption, without the need to be controlled by third parties. This thesis wants to explain in a clear and comprehensible way, even for those who do not have the technical skills, how this electronic monetary system has been working for 10 years without interruption. The topics covered are mainly focused on the mathematical explanations that allow the Bitcoin protocol to work, trying to answer some of the most poured questions that arise to the novices. Such as:

1. How Bitcoin works
2. How and why Bitcoin is decentralized
3. Who controls and governs Bitcoin
4. What Bitcoin mining is and how it works
5. What determines the value of Bitcoin

Being a relatively new topic, the prevalence of the materials used have been collected online from sources that have proved reliable, such as Bitcoin Wiki and Blockchain.com, from one of the most popular books within the Bitcoin community: *Mastering Bitcoin* by Andreas Antonopoulos, and from some online video courses such as *Blockchain Deep Fundamentals* by Ivan On Tech.

1 | Bitcoin and Blockchain

1.1 The birth of Bitcoin

"*I've been working on a new electronic cash system that is completely peer-to-peer, with no trusted third parties*", on October 31, 2008, Satoshi Nakamoto announced the Bitcoin design paper in a mailing list for cryptographers called "*metzdowd*". In the above mentioned paper, known as White Paper and called "*Bitcoin: A Peer-to-Peer Electronic Cash System*", there is the first detailed description of the Bitcoin protocol. In it we can highlight some fundamental characteristics for the correct functioning of the system:

1. Allows you to send payments directly from one person to another online without going through an intermediary.
2. Prevents *double-spending* via a peer-to-peer network.
3. Generates a registry consisting of a chain of transactions encrypted by a Hash function and grouped into blocks. These are validated by a *Proof-of-Work* mechanism.
4. The longest transaction chain is considered valid by all nodes in the network.
5. The nodes of the network are free to unhook and reconnect at will anonymously.

These five elements are fundamental for the constitution and proper functioning of Bitcoin and are the basis of the Blockchain concept. The term Blockchain comes from the structure of the database consisting of chains of blocks of transactions. Through the Blockchain you establish a database distributed among all nodes within the network. The transaction database is made public and the data already entered in it will be irreversible and not modifiable without the consent of the majority of nodes. This simple but revolutionary mechanism allowed the creation of the first digital currency

able to solve the problem of double spending without the need for an intermediary. The first version of Bitcoin software was released on January 9, 2009 also by Satoshi Nakamoto, Bitcoin v0.1.

1.2 How Bitcoin prevents *double-spending*

The problem of double spending is traditionally solved by a Trusted Third Party (TTP). A TTP is a certifier of a successful money transaction that allows counterparties relying on the TTP to avoid the risk of paying twice. An example of TTP is Paypal which has full control of the user's transaction database and actually replaces the other party (the buyer pays Paypal which in turn pays the seller). So users lose control of their funds to the TTP. The TTP, by checking the database, can decide at any time to cancel a transaction or to block the funds of a specific user if it deems it necessary, a circumstance that is increasingly recurrent.

With the system designed by Satoshi Nakamoto, people are able to spend their digital currency irreversibly, and in such a way that anyone can verify the validity of each transaction. Each bitcoin, the Bitcoin blockchain's unit of account, is transferred from one user to another in a *peer-to-peer* manner. Transfers are made through a cryptographic system that allows users to verify the ownership of bitcoins. Through this mechanism a chain of transactions that anyone can go to check or verify and which, through a transaction validation system, becomes immutable.

Blockchain blocks are data containers that collect information about past transactions. To generate a new block it is necessary to solve a mathematical problem (*Proof-of-Work*), using computational power and thus consuming electricity. Whoever succeeds in solving the problem first receives as a reward new bitcoins just generated by the network itself. This reward is also known as *coinbase*. The difficulty to solve the problem varies according to the number of nodes that are tending to solve it. This change is determined automatically by the network and takes about 10 minutes to resolve the problem. Nodes that deal with block validation are better known as *miners*. This name derives from the comparison often used between solving the mathematical problem and the work done by gold miners. Bitcoin miners consume electricity and computational power to be rewarded in bitcoin, while gold miners use their resources and energy to find the gold deposits. The main objective of the Bitcoin validation process, or *Bitcoin mining*, is to enable network nodes to reach a consensus on past transactions completed in a secure and *tamper-resistant* manner.

1.3 The longest chain is the valid one

The rule of the longest chain is fundamental for the nodes to reach consensus on the information contained within the blockchain. According to this rule, in case of conflicts, nodes must always consider legitimate the blockchain that contains the most information as it will be the one that has been validated more times. A conflict can arise when two miners manage to solve the mathematical problem almost simultaneously. In this case there will be two chains both considered valid by the nodes but with different information inside the last block. To resolve the conflict the rule of the longest chain applies: given these two chains, the first one to which a miner will succeed in generating a new block will be considered valid by all the nodes, even by the nodes that previously held the information of the competing chain.

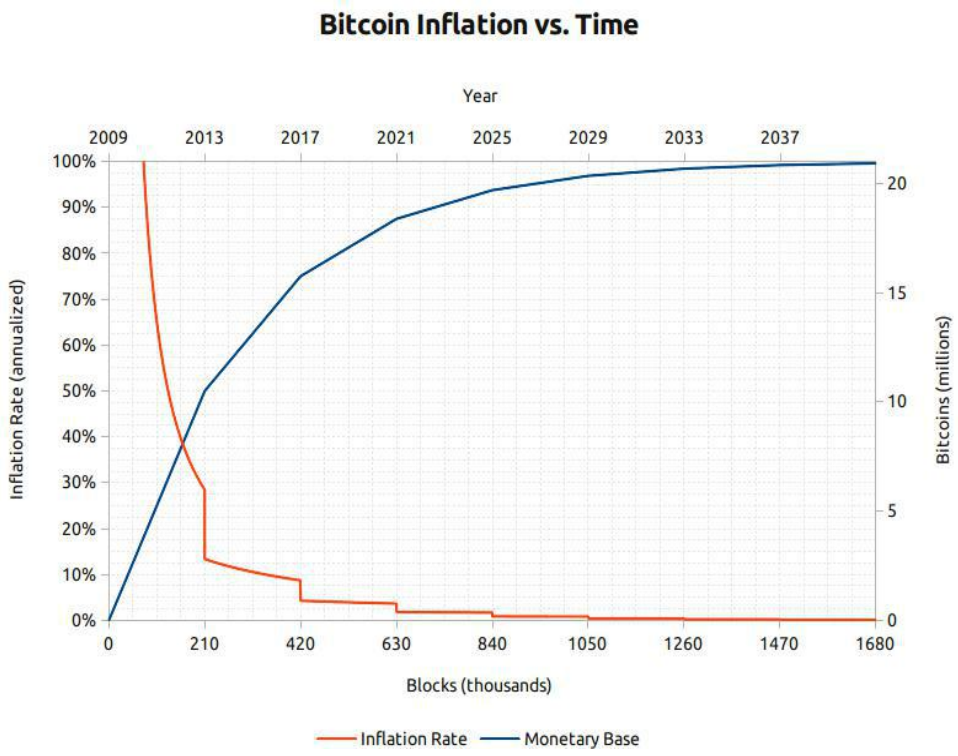
1.4 Bitcoin's pseudo-anonymity

The decentralized nature of Bitcoin is based on *permissionless* system where no permission is needed to be part of the network but anyone can become a node and start validating transactions without any identification process. The nodes are nothing more than computers running Bitcoin software, and holding a copy of the blockchain with all its information. The bitcoin ownership mechanism is based on a Public Key and Private Key system (Cap 3.4 - Public Key Infrastructure). Through this system you can verify the ownership of your funds without having to issue any identification, therefore anonymously. Since the transaction database is public, anyone can check what transactions have been made. In this case, however, you will only see that a certain Bitcoin address, consisting of an alphanumeric string, has sent a bitcoin amount to another address. There is no way to link the owner person to an address if the first one has not voluntarily advertised it. For this reason the Bitcoin property model is defined as *pseudo-anonymized*.

2 | The Basics of Bitcoin Operation

2.1 Bitcoin halving

As already explained, bitcoins are generated approximately every 10 minutes through the mining process, i.e. transaction validation. The miner who manages to validate the new block receives the coinbase as a reward for his work. At the time of writing this thesis, June 2019, a fee of 12.5 bitcoins (BTC) was received for each block tested. Every 4 years the fee is halved in a process known as *Bitcoin halving*, in fact, in the first 4 years of Bitcoin's life, miners received 50 BTC for each validated block. The rate of generation of new bitcoins is therefore constantly decreasing, and will continue to decrease until the maximum limit of bitcoins in circulation is reached. This limit of 21 million bitcoins will be reached in 2140 when the miners will no longer be rewarded with coinbase. As early as 2019, more than half of the bitcoins were put into circulation.



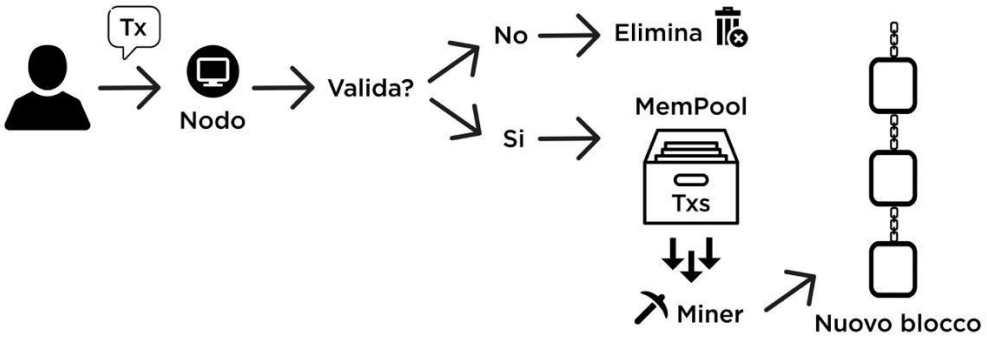
Bitcoin Inflation vs. Time - https://bitcointalk.org/index.php?topic=130619.0

2.2 Block sizes, MemPool and commissions

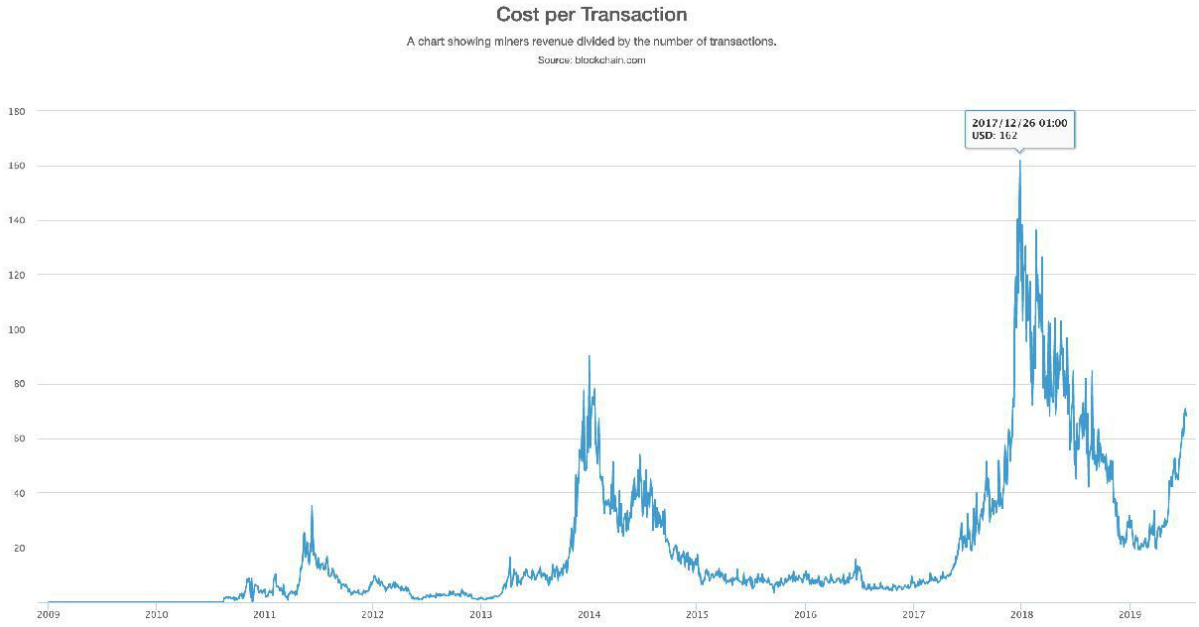
Each bitcoin is divisible up to 8 decimal places, where the last decimal place is named *Satoshi*, in honor of the creator of the protocol. Due to their digital nature, bitcoins are easily tradable in online markets, better known as *Exchanges,* at a value determined purely by supply and demand. Currently, June 25, 2019, the capitalization of bitcoins has exceeded 200 billion, and each bitcoin is traded for more than 11,000.

In addition to the remuneration from the coinbase, miners are also rewarded by the commissions paid by users who want to confirm their transactions. Each block may contain a limited number of transactions, as the size of the block must not exceed 1 MB. Because of this limitation, not all transactions that users have sent to nodes are immediately entered into the Blockchain. When a user sends a transaction, it is first deposited into the MemPool (*Memory Pool*), a container of all transactions not yet confirmed. From MemPool, the miners select the ones they will insert into the next block to be validated. They usually

favour transactions with the highest commissions, to the detriment of the others even if they have been transmitted previously. For this reason, some users will find themselves waiting more than 10 minutes before seeing their transaction confirmed and added to the distributed database.



The amount of the commission is determined directly by users and varies according to the state of the network. For example, if the network is congested with a large amount of transactions, commissions will tend to increase, and users who do not want to pay high tips will have to wait even more than an hour to see their transactions confirmed. In 2017 a user paid up to 162 for a single transaction.



blockchain.com: "Cost per Transaction" - https://www.blockchain.com/charts/cost-per-transaction?timespan=all

3 | Hash, Merkle Tree and Public Key Infrastructure Function

With Bitcoin's continued growth in adoption and notoriety, more and more people are approaching concepts that were known only to the most experienced. Concepts such as:

- Hash functions
- Merkle Tree
- Public Key Infrastructure (PKI)
- Proof of Work (PoW)

3.1 Hash functions

One of the main components of Blockchain technology applied to the Bitcoin protocol is Hash encryption. A hash function is a function that allows you to transform binary information into a single alphanumeric string of a fixed length. We can identify two main properties that characterize these functions:

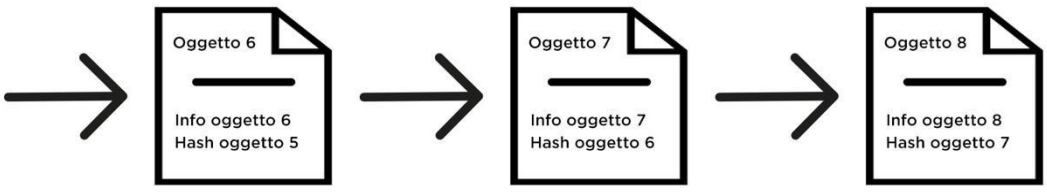
1. They are unidirectional, i.e. once a hash has been generated it is not possible to obtain the information that constituted it.
2. They are unique, it is practically impossible to generate two equal strings from different information inputs. If you change even one value within the input, you will generate a completely

different output. We can then define a hash function as a fingerprint of the object used as input, an object that can be for example a document, a movie, music or any other file consisting of binary data.

There are various hash functions with different degrees of security determined by the difficulty of recovering input from output. In the Bitcoin protocol, SHA256 is used to generate alphanumeric strings with a length of 64 characters. For example, the phrase "*I love Bitcoin*" is transformed into the string: "e84e38b1b9d57f5e13867cc955bd146bc- 1704550195540ff71d6d4ebccd97bc4". If we wanted to change even one letter of the main sentence, we would get a completely different hash string. In fact, the phrase "***Love Bitcoin*** is transformed into: "f8c8d9dbd1633b764adfb02ed1ca1836a414021ab-2335b41fbb969756e31a6bb". There is no way to retrieve the two initial sentences from the final outputs, so they can be used as evidence of the integrity of the information used as input. In fact, anyone in possession of a hash string and its input can check at any time that the latter has not been modified after the first has been made public. This last concept is fundamental to generate a blockchain that is temper-resistant.

3.2 Hash function applied to the blockchain

The hash functions can also be used to create nested data structures. Using a combination of an object and an existing hash string, you can create a chain of information represented by a single hash.

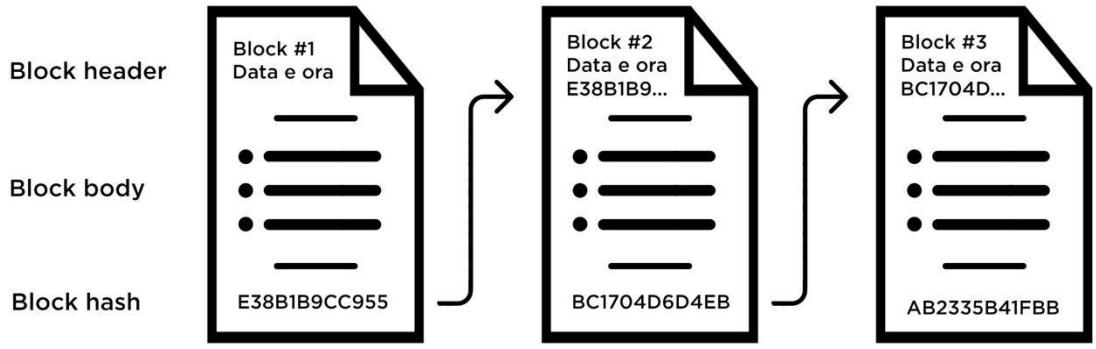


Taking the example shown in the image, if you modify the information contained in object 6, you will necessarily modify the hash of object 7 and object 8. In this way, the output of new objects is always dependent on the previous objects, and you can also verify that the information has not been changed by simply looking at whether you changed the last hash. The blockchain takes up

the structure of these chains, where the objects are made up of transaction blocks and each block consists of two distinct parts:

1. The *block header*
2. The *Block body*

The block header contains the hash string of the previous block and other information, such as the date and time of block creation. The block body contains the transactions collected by the miners selected by MemPool. The header block and the body block are used together as input to generate a new hash string, known as the *hash block*, which is entered into the header block of the next block. In this way you can reproduce all the properties of the hash string chain to create a database of transactions organized in time blocks, persistent and tamper-evident.



3.3 Reordering transactions in a Merkle Tree

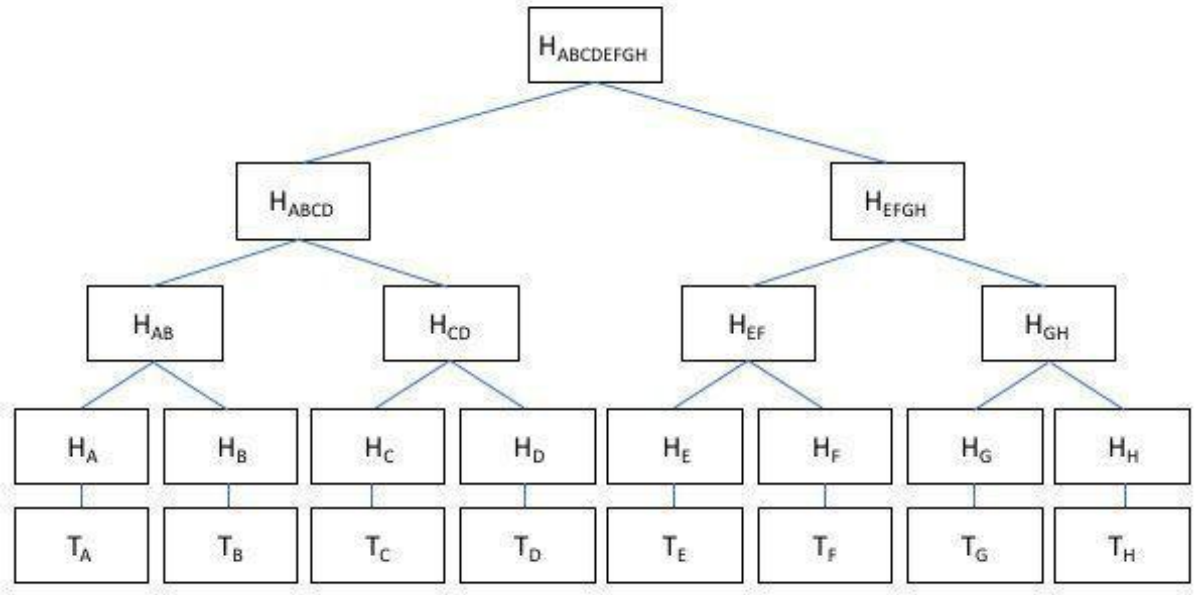
The block body can contain a large number of transactions that depends on 4 main factors:

1. Limit to block size, which can be a maximum of 1MB
2. Size of each transaction, usually an elementary transaction weighs about 250 bytes.
3. Number of transactions requested in the last 10 minutes
4. Number of transactions within MemPool

In order to better manage all transactions within each block, they are organized according to a structure called Merkle Tree. A Merkle Tree is a "tree" structure composed of hash strings with a particular procedure.

1. Each transaction within a block is transformed into a hash string
2. These are put together in pairs by two, and the pair itself is converted into a new string
3. Continue to create new strings in this way until it generates a single hash output, known as *Merkle Root*.

The Merkle Root represents the fingerprint of all the transactions contained within the block, and will have all the features of hashes and hash chains. So if you change even a single block transaction, the root will change completely. Through this structure, the management of the set of transactions is made much less expensive, as well as facilitating the control of any tampering.



Investopedia: "Merkle Tree" - https://www.investopedia.com/terms/m/merkle-tree.asp

3.4 Public Key Infrastructure (PKI)

The bitcoin ownership mechanism is based on a Public Key and Private Key system. This system, known as Public Key Infrastructure (PKI), consists of:

1. A function capable of generating a *key pair*
2. An algorithm to sign using the key pair
3. A function that can verify that the signature is correct

The key pair consists of a private key (*Private Key*) and a public key (*Public Key*). As the same term suggests, the public key is the part that is made public, like the IBAN code of a current account, while the private key will be the PIN code that allows you to manage the current account. All data encrypted via the public key can only be decrypted via the private correspondent and there is no way to retrieve the latter by knowing the first one. In the Bitcoin protocol, the public key is used to generate a Bitcoin address needed to receive transactions or request payments. The private key constitutes proof of ownership of the bitcoins contained within a Bitcoin address and is the only means of accessing the funds of a Bitcoin address. If you lose your private key, the bitcoins contained in the address can never be used again and are considered "*burned*". By June 2017, an estimated 4 million bitcoins were lost forever6.

4 | Key Pairs, Bitcoin Addresses and Digital Signatures

4.1 How Private keys and Public keys are generated

A private key is a 256 bits number chosen randomly. You can create a new private key by simply flipping a coin 256 times and transcribing the results of the flips. In this way you will have a series of "heads" and "tails" that match the binary values of the private key. A private key therefore corresponds to a number between 1 and ^{2256} and is represented according to the *Wallet Import Format*, or *WIF*, based on *Base58* encoding.

The public key is generated from a private key by elliptic curve multiplication. This multiplication is as follows:

$$K = k * G$$

Where k corresponds to the private key, G to a constant known as the *Generation Point*, and K to the generated public key. The elliptical curve multiplication is an irreversible operation, i.e. from a public key K it is impossible to find the private key k that generated it. This feature is essential for the proper functioning of Bitcoin's digital signature system, as you can share your public key without revealing the corresponding private key.

4.2 Bitcoin addresses

From a public key you can generate a Bitcoin address consisting of a series of letters and

numbers. Bitcoin addresses are used to send and receive transactions from users in the same way as the IBAN of a bank account. As already introduced in the first chapter (Chapter 1.4 - Bitcoin's Pseudo-anonymity), there is no way to link the natural person owner to an address if the first one has not voluntarily advertised it. A user can generate a multiple number of Bitcoin addresses and use a new one for each transaction. In this way is able to send and receive transactions while maintaining its privacy and partly counteracting the transparency of Blockchain.

Bitcoin addresses are also generated using a one-wayash function. In this case the public key is transformed through the SHA256 function and tracked through the RIPEMD160 function, in order to generate a 160 bits number. So we'll have that:

$$A \quad = \quad RIPEMD160(SHA256(K))$$

Where K corresponds to the public key and A to the 160 bits encrypted version of the public key. From the public key encrypted A, *a checksum is* calculated to verify that our key hasn't been tampered with. To calculate the *checksum* we proceed with a double hash of the public key A, and then we take into account only the first 8 hexadecimal characters of the resulting hash. So we'll have that:

$$checksum \quad = \quad SHA256(SHA256(A))[: \quad 8]$$

The *checksum* is added to the hexadecimal string of A, the resulting value is encoded according to the *Base58* encoding, and we get our Bitcoin address. With the following Python code you can generate a new private key, with the corresponding public key and the Bitcoin address. Due to possible security problems, I do not advise anyone to use keys derived from the following code, but to use them for information purposes only.

Code Listing 4.1: Python - Generate Bitcoin Private Key

```
import secrets
import codecs
import ecdsa
import binascii
import hashlib
import base58

print('----------------')

#Prendiamo un numero random a 256 bits
bits = secrets.randbits(256)
print('Numero casuale:')
print(bits)

print('----------------')

bits_hex = hex(bits)
private_key = bits_hex[2:]
print('Private Key:')
print(private_key)

print('----------------')

binary_private_key = bin(int('1'+private_key, 16))[3:]
print('Valori binari della Private Key:')
print(binary_private_key)
#Potrebbe essere equivalente a 256 lanci di una moneta

print('----------------')

private_key_WIF = base58.b58encode(codecs.decode(private_key, '
    hex')).decode("utf-8")
print('Private Key formato WIF:')
print(private_key_WIF)

print('----------------')

#Misuriamo la public key tramite la moltiplicazione a curva
    ellittica
signign_key = ecdsa.SigningKey.from_string(codecs.decode(
    private_key, 'hex'), curve=ecdsa.SECP256k1)
verifying_key = signign_key.verifying_key
public_key = (b'\04' + signign_key.verifying_key.to_string()).
    hex()
print('Public Key:')
print(public_key)

print('----------------')

public_key_bytes = codecs.decode(public_key, 'hex')
# Funzione SHA-256 per la public key
sha256_bpk = hashlib.sha256(public_key_bytes)
sha256_bpk_digest = sha256_bpk.digest()
# Funzione RIPEMD-160 per la stringa SHA-256
```

```
ripemd160_bpk = hashlib.new('ripemd160')
ripemd160_bpk.update(sha256_bpk_digest)
ripemd160_bpk_digest = ripemd160_bpk.digest()
ripemd160_bpk_hex = codecs.encode(ripemd160_bpk_digest, 'hex')

RIPEMD_result = ripemd160_bpk_hex
print('Stringa RIPEMD-160 --> A:')
print(RIPEMD_result.decode("utf-8"))

#Aggiungi bytes per il mainnet di Bitcoin
RIPEMD_result_mainnet = b'00' + RIPEMD_result

print('----------------')

#Misurare il checksum
sha256_nbpk = hashlib.sha256(RIPEMD_result_mainnet)
sha256_nbpk_digest = sha256_nbpk.digest()
sha256_2_nbpk = hashlib.sha256(sha256_nbpk_digest)
sha256_2_nbpk_digest = sha256_2_nbpk.digest()
sha256_2_hex = codecs.encode(sha256_2_nbpk_digest, 'hex')
checksum = sha256_2_hex[:8]
print('Checksum:')
print(checksum.decode("utf-8"))

print('----------------')

#Funzione presa da: https://www.freecodecamp.org/news/how-to-
    create-a-bitcoin-wallet-address-from-a-private-key-
    eca3ddd9c05f/
def base58(address_hex):
    alphabet = '123456789
       ABCDEFGHJKLMNPQRSTUVWXYZabcdefghijkmnopqrstuvwxyz'
    b58_string = ''
    # Get the number of leading zeros
    leading_zeros = len(address_hex) - len(address_hex.lstrip('
       0'))
    # Convert hex to decimal
    address_int = int(address_hex, 16)
    # Append digits to the start of string
    while address_int > 0:
        digit = address_int % 58
        digit_char = alphabet[digit]
        b58_string = digit_char + b58_string
        address_int //= 58
    # Add    1    for each 2 leading zeros
    ones = leading_zeros // 2
    for one in range(ones):
        b58_string = '1' + b58_string
    return b58_string

address_hex = RIPEMD_result_mainnet.decode("utf-8") + checksum.
    decode("utf-8")
address = base58(address_hex)
print('BTC address:')
print(address)
```

4.3 Digital signatures on Bitcoin

The Bitcoin digital signature system is possible through the Public Key Infrastructure. This system consists of a mathematical scheme that allows you to sign a message using a private key, and to verify the signature by means of the corresponding public key. In the case of Bitcoin, the message to be signed corresponds to the transaction authorized by the owner of the Bitcoin address. The function to generate the signature is the following:

$$Sig \quad = \quad F_{sig}(F_{hash}(m), dA)$$

Where dA corresponds to the private key, m is the message (transaction), $Fhash$ is the function that generates the message hash, $Fsig$ is the function that generates the signature, and Sig is the resulting output. This output consists of two numbers, known as R and S, which, in addition to the transaction hash and the public key, are required to verify the signature.

5 | Introduction to Proof of Work (PoW)

For a Blockchain to be considered valid, all nodes in the network must reach consensus on transactions confirmed in the past. Only with the consent of all nodes is it possible to be sure of the integrity of the information contained within the Blockchain.

5.1 Distributed Consent: Byzantine Generals' Problem

Satoshi Nakamoto, through the Bitcoin protocol, was able to solve one of the problems that was adverse on distributed computer systems: the *Byzantine Generals' Problem* or *Byzantine Generals' Problem*. This problem was first exposed in a paper published in 19827 and is often used as an analogy to represent the distributed consensus of network nodes. While remaining anonymous and without knowing each other's intentions, the knots must be able to agree and reach consensus. The creators of the paper proposed the problem in the following way:

> "Let us imagine that several divisions of the Byzantine army are camped outside an enemy city, where each division is commanded by its own general. Generals can communicate with each other only through a messenger and after observing the enemy, they must agree on a common action plan. However, some of the generals may be traitors who try to prevent loyal generals from reaching an agreement. To solve the problem, generals must have an algorithm that ensures that all loyal generals decide on the same plan of action and that a small number of traitors cannot induce loyal generals to adopt a bad plan of action".

Basically one wonders how to ensure that several subjects, physically separated and without knowing each other, are able to come to an absolute agreement before taking the desired

action. On November 13, 2008, Satoshi Nakamoto himself explained his solution in the same mailing list where he announced the creation of Bitcoin8. Nakamoto's solution consists in proving the existence of an agreement between all nodes by means of a proof of work, or Proof-of-Work (PoW). The generals must be able to agree on the date and time when they will attack, it is not important when they attack but it is important that they attack all together. Any general, through his messengers, can propose to others a date and time to attack. Only the first proposal received shall be considered valid by the receiving general. Unfortunately, information is reached asynchronously, i.e. it is not necessarily the case that all generals receive the same information at the same time. This problem is solved by a work test based on hash string generation. Once a general receives a proposal for an attack, he uses it to produce a hash that must have certain predetermined characteristics. To be able to produce it with these characteristics, it will take approximately 10 minutes of joint work between all the generals. Once a general manages to find the hash with the predetermined characteristics, the work test is considered resolved and the solution is passed on to all other generals. The generals who receive the solution, use it to start generating a new hash with the same predetermined characteristics. If someone was working on a different attack time than the one shared through the solution, they will necessarily have to start working on a hash containing the proposal received. If in two hours the generals manage to solve 12 Proof-of-Work generating 12 different solutions with the same attack time, the distributed consensus will be demonstrated. In fact, if to generate a solution were needed about 10 minutes using the overall work of all the generals, in 2 hours of total work you should have solved the PoW 12 times. At this point all the generals are working on the same attack proposal, and the date and time will be known by all. We can then proceed to action, solving the problem of the Byzantine generals.

5.2 Consent Distributed among Bitcoin nodes

As in the case of Byzantine generals, the consensus between nodes within Bitcoin is reached asynchronously, i.e. nodes do not wait to synchronize with each other to determine which block to add to their database, but choose based on the information that comes to them individually available. The most important of these is the Proof-of-Work solution. To add a new blockchain to the blockchain, miners must solve this mathematical problem that requires computational power and electricity. The first miner that manages to solve it, will have to transmit the solution to the other nodes, which once received, will proceed to update their database with the transactions contained within the validated block. Once the block has been received by all nodes, the distributed database will have been updated with the same information and distributed consent will have been reached.

6 | The mathematics behind Proof-of-Work

6.1 Proof-of-Work algorithm

Proof-of-Work consists of creating a hash string with the information contained within the header block so that the hash has certain characteristics. If you understand how hash functions work (Cap 3.1 - Hash functions), you should be clear that the same input always generates the same output. Then how can I generate hashes with certain characteristics using the same information in the header block? This is possible thanks to a particular object: the *nonce*. The nonce is a random number that is entered into the header block to generate new strings; only by modifying the nonce you can generate different hashes using the same information.

Another important feature of hash functions is that it is impossible to predict which output is generated from a given input. For this reason the miners, in order to generate a string with the desired characteristics, will only have to proceed with attempts, making the solution of the mathematical problem more similar to winning a lottery. The more computational power you use, the more lottery tickets you can buy and the better your chances of winning. For example, the phrase " *"Amo Bitcoin"* is transformed into the string:

"e84e38b1b9d57f5e13867cc955bd146bc1704550195540ff71d6d4ebccd97bc4"

If instead we wanted to produce a different hash string starting with a "**0**" and using the same phrase, the only solution would be to add a *nonce*. In this way, the input that will be used to generate the new hash will be: "Phrase" + "nonce". For example, the phrase "Amo Bitcoin**0**",

produces the hash:

"5c5f5a0c9c7c9b4b958d9cf79cf79c0d140c2310e2ae4fbdff1a883d714a2b734bc1"

By continuing to generate new strings with other nonce, we will have the following results:

Phrase + Nonce	String Hash
Amo Bitcoin	e84e38b1b9d57f5e13867cc955bd146bc1704550195540ff71d6d4ebccd97bc4
Amo Bitcoin0	5c5f5a0c9c7c9b4b958d9cf79c0d140c2310e2ae4fbdff1a883d714a2b734bc1
Amo Bitcoin1	9e19a2b61a173c922f98b5d317da2011d393c835628d83a4a9f617528c6d92b3
Amo Bitcoin2	ec98d0c6573dbdda60ce6d90270a9bc2719fff8a21a644b35341781618a1b98c
Amo Bitcoin3	**0**187efb67ca570f679e123317aa10809654fd6bb960c45f407660ea5acfb882a

So after 4 attempts, the "I love Bitcoin3" input generates a hash that starts with a zero, exactly as we requested.

6.2 Target and hash value

In the case of Bitcoin, the characteristics necessary for a hash to be considered valid are determined by the *target*. The target is a value that must not be exceeded by the hash of a new block for PoW to be considered resolved. Basically, miners use the information contained in the block header to generate the equivalent hash string, if the string value is higher than the target, the miner will change nonce to generate a new hash with a different value, until the latter is lower than the target. An output generated by the SHA256 function generates a 256-bit value which is expressed in a hexadecimal number. The hexadecimal number system is a system based on 16 characters, unlike the commonly used system based on 10 characters (0 to 9). The characters used in the hexadecimal system are numbers from 0 to 9 and letters from A to F. Using this numerical system it is possible to represent high values using a small number of characters, for example, it is possible to express values up

to 255 with only 2 characters. Using a programming language like Python, you are able to simply convert hexadecimal numbers to decimal numbers. For example, taking the hash from "Amo Bitcoin3":

0187efb67ca570f679e123317aa10809654fd6bb960c45f407660ea5acfb882a

And using the code:

Code Listing 6.1: Python - Get Hash decimal value

```
1 block_hash = '0187
    efb67ca570f679e123317aa10809654fd6bb960c45f407660ea5acfb882a
    '
2 hash_value = int(block_hash, 16)
3 print(hash_value)
```

We can derive the decimal value of the hash, which in this case is equivalent to:

$$6.92E + 74$$

6.3 Target representation and target value

As for the target value, this is contained within the header block of each block. Its representation is known as *bits* or *target bits*, and consists of an 8-character hexadecimal number. For the block to be considered valid and added to the blockchain, the target value must be higher than the block value. For example, the last block validated at the time I'm writing this thesis is block number # 584362. Going to check in a Block Explorer, i.e. a website connected to the blockchain that allows you to view blocks and transactions, you can easily know the *target bits* of this block. In this case, according to blockchain.com9, the target bits of block # 584362 corresponds to the decimal value 388200748, which with the hexadecimal system is 1723792c. The first two values of the hexadecimal target bits are called *exponent*, while the next 6 characters are known as *coefficient*. Once you know the exponent and the coefficient, you can determine the target of the block. The target will be equivalent to:

$$target = coefficient * 2^{8*(esponent-3}$$

In the case of block 584362, the target will correspond to:

$$target = 0x23792 * 2^{0x08*(0x17-0x03}$$

In decimal numbers, we'll have:

$$target = 2324780 * 2^{160}$$

$$target = 3.40E + 54$$

So, the first miner who managed to generate a block with a hash value of less than 3.40E+54, was able to validate the block and receive the bitcoin reward.

6.4 Clarification of target and difficulty

In summary, the target is the maximum value that a block hash can have for it to be considered valid by the network. The higher the target, the easier it will be to generate valid blocks. We can therefore say that the smaller the target, the more difficult it will be for the miners to validate a blockade, and the greater the consumption of resources. As already introduced, since it is not possible to predict the characteristics of a hash string given the information used to generate it, the whole mining process can be represented as a lottery. The miners use their resources to buy numbered tickets for this lottery until a winner is declared. The number of each ticket corresponds to the value of the hash generated by the block and nonce information. The winner will be the one who first finds a ticket whose number is lower than the target number.

The Difficulty to undermine a new block is represented as the ratio between the target value of the first block of the blockchain, or Genesis Block, and the target of the block you want to compare. So we'll have that:

$$Difficulty = \frac{GenesisBlockHashValue}{ComparisonBlockHashValue}$$

Resuming the case of block # 584362, and knowing that the hash of the genesis block corresponds to 0x000000ffffffffffffffffffffffffffffffffffffffffffffffffffffffffff, we will have:

$$Difficulty = \frac{0x000000ffffffffffffffffffffffffffffffffffffffffffffffffffffffffff}{0x0000000000000000023792600000000000000000000000000000000000000000}$$

$$Difficulty = 7.93E + 12$$

The difficulty indicates how much more expensive it is to validate a new block than the resources used to validate the Genesis Block. To date it is 7.93 trillion times more expensive to validate a new block. The difficulty is therefore a representation of the target more understandable to man. To further simplify the concept of target and understand whether a block is valid or not, reference is often made to the number of 0 initials within the binary hash representation of the block to be validated. Basically, both the target and the hash of the block are 256 bits numbers, so are composed of a series of 256 binary values, i.e. consisting of zeros and ones. The more initial zeros the binary representation contains, the lower will be the value of the target or hash and for this reason if the hash contains more initial zeros than those contained by the target, we can be sure that its value is lower than the value of the target and that the block can be considered valid.

With the following Python code you can calculate the target value of a block and its difficulty by entering only the corresponding target bits expressed according to the decimal number system:

Code Listing 6.2: Python - Get Block Target value

```
1  from decimal import Decimal
2
3  bits = 388200748 #inserire qui il valore decimale del target
       bits
4  target_bits = hex(bits)
5
6  print('Target bits:', target_bits)
7
8  exponent = '0x'+target_bits[2:4]
9  coefficient = '0x'+target_bits[4:]
10
11 target = int(coefficient,16) * 2**(8 * (int(exponent,16)-3))
12
13 hex_target = hex(target)
14
15 if len(hex_target) < 65:
16   number_of_zeros = 64 - len(hex_target)
17   hex_target = str(hex_target)[2:]
18   for n in range(0, number_of_zeros):
19     hex_target = '0'+hex_target
20   hex_target = '0x'+hex_target
21
22 print('Hex Target:', hex_target)
23
24 target_value = int(hex_target, 16)
25 print('Target value:', '%.2E' % Decimal(target_value))
26
27 print('Lenght:', len(hex_target))
28
29 original_target = '0
```

```
    x00000000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
    '

original_value = int(original_target, 16)

print('Original value:', '%.2E' % Decimal(original_value))

difficulty = int(original_target, 16) / int(hex_target,16)

print('Difficulty:', '%.2E' % Decimal(difficulty))
```

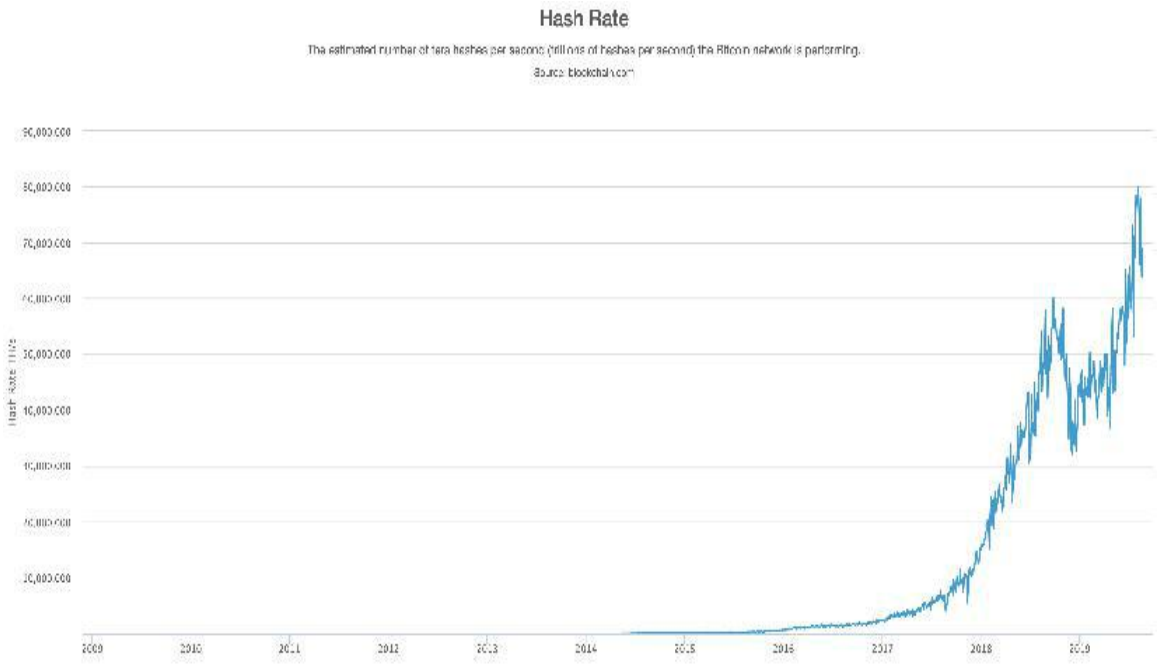
6.5 Target Recalibration

The target establishes the difficulty for a miner to validate a new block. Taking up the lottery analogy, it establishes the percentage of the number of winning tickets out of the total number of tickets that can be purchased. A miner can only check one hash at a time, which is equivalent to buying one lottery ticket at a time. However, the greater the miner's computational power, the faster it will be able to generate new hashes and buy new tickets. For this reason, a miner with greater computational power will have a better chance to generate a valid block and win the lottery, then receive the reward in bitcoin. A miner with less computational power will still be able to win the lottery if with a little luck he can find a winning ticket first. The higher the target, the more winning tickets will be available and consequently the difficulty will be low. Given a constant target value, as the computational power increases, the time to win the lottery is reduced. In this case, the number of winning tickets remains the same, but the number of tickets purchased in the same period of time increases, increasing the probability of finding a winning ticket in less time. As already explained in previous chapters, in the Bitcoin protocol a block is validated approximately every 10 minutes, leading to the generation of new bitcoins. If the target were to remain constant, as the computational power of the miners increases, the validation time of a block and the time of generations of new bitcoins would be reduced. In order to keep this period stable, while having variations in the total computational power of the miners, the Bitcoin protocol provides for a recalibration of the target. This recalibration is performed automatically every 2016 validated blocks, so every 20160 minutes or so, which

is equivalent to 2 weeks. The target is recalibrated so that the miners, using their current computational power, can validate a block every 10 minutes. Basically, if in the last two weeks on average more than 10 minutes to validate a block, the target will be raised to lower the difficulty. Conversely, if it took on average less than 10 minutes, the target will be lowered to increase the difficulty and restore balance. So we'll have that:

$$NewTarget = OldTarget * \frac{Timerequiredtovalidatethelast2016blocks}{20160minutes}$$

The computational power is measured by the number of hashes that mining computers are able to generate and verify every second, this measurement is known as hashrate/s. In the last 10 years, we have witnessed an exponential growth of hashrate, which to date reaches 70 million terahash (TH/s) per second, where 1TH/s is equivalent to 1,000,000,000 (one trillion) hash generated per second.



blockchain.com. "Hash Rate" - https://www.blockchain.com/charts/hash-rate?timespan=all

Conclusion

The objective of this thesis is to answer some of the most common questions of those who approach Bitcoin for the first time. The arguments were deliberately focused on analytical explanations of the functioning of this new monetary system, trying to avoid possible economic effects. Once we have proven the effectiveness of the mathematical mechanisms on which this thesis is based, we could focus on the consequences that this whole structure could bring to the economy, especially in macroeconomic terms. The purpose of Bitcoin is not to be an investment vehicle, but a revolutionary and innovative product, just like the steam engine, railways, steel or e-mail. Its quality of being composed entirely of software, based on mathematics and cryptography makes it the first existing instrument that guarantees a way out of the monetary control of governments without the need for any permission. Even those who previously could not participate in the global economy because they lacked any access to financial services, the so-called "*unbanked*", can now participate without any restrictions. Although these effects are not yet evident in the Western world, they are already being felt in some countries, such as Venezuela and Zimbabwe, both of which are marked by strong hyperinflation. The citizens of these two countries immediately understood the potential of Bitcoin, and used it in the dark by their own government to protect themselves against the devaluation of the local currency and as a store of value. Probably the most important feature of the digital monetary system that Bitcoin guarantees is its free availability to anyone. It is a system that is completely open, transparent, neutral, verifiable, peer-to-peer and above all free of geographical limits. It can be transferred anywhere in the world without any additional constraints, unlike other systems used for thousands of years, such as gold and cash. As the *Times, a* British newspaper, has pointed out, this has strong implications for the monetary policies that governments may adopt.

"If a future Fed president were to try to repeat Ben Bernanke's policy of quantitative easing

(actually printing money), worried investors could start to unburden their savings from the dollar and stream them to the cloud so quickly that the Fed would be forced to change course. [...] Once alternative currencies are available on the Internet without friction, each laptop will become its own Cayman island.

Bibliography

Satoshi, Nakamoto (2008). *Bitcoin: A Peer-to-Peer Electronic Cash System.*

Andreas M., Antonopoulos (2017). *Mastering Bitcoin: Programming the Open Blockchain.* O' Reilly Media, Inc., CA.

Roberto, Garavaglia (2018). *All about Blockchain, understanding technology and new opportunities.* Hoepli.

Leslie Lamport, Robert Shostak, and Marshall Pease (1982). *The Byzantine Generals Problem.* TOPLAS.

Sitography

Blockchain.com: https://www.blockchain.com/

Bitcoin Talk: https://bitcointalk.org

Ivan On Tech Academy: https://www.ivanontech.com/

Bitcoin.com: https://www.bitcoin.com

Bitcoin Wiki: https://en.bitcoin.it/wiki/

Metzdowd: https://www.metzdowd.com

Mycryptopedia: https://www.mycryptopedia.com

Investopedia: https://www.investopedia.com/

Free Code Camp: https://www.freecodecamp.org

An Integrated World: https://www.anintegratedworld.com

Learn Me at Bitcoin: https://learnmeabitcoin.com/

Seeking Alpha: https://seekingalpha.com

The Next Web: https://thenextweb.com

Satoshi Nakamoto: http://satoshinakamoto.me