Sommario

Introduzione

Nonostante la crescente consapevolezza di Bitcoin, la maggior parte delle persone ancora non conosce e non comprende le proprietà di questo nuovo sistema e come sia in grado di svilupparsi e sostenersi in modo completamente autonomo. Ad un primo approccio solitamente prevale lo scetticismo, non si riesce a concepire e a credere nell'esistenza di un nuovo modello monetario completamente governato da algoritmi e dalla crittografia, senza la necessità di essere controllato da terze parti. Questa tesi vuole spiegare in maniera chiara e comprensibile anche per chi non ha le competenze tecniche, in che modo tale sistema monetario elettronico stia funzionando da ormai 10 anni senza alcuna interruzione. I temi trattati sono incentrati principalmente sulle spiegazioni matematiche che permettono il funzionamento del protocollo Bitcoin, tentando di rispondere ad alcuni dei più riversati quesiti che sorgono ai novizi. Quali:

1. Come funziona Bitcoin
2. Come e perché Bitcoin è decentralizzato
3. Chi controlla e governa Bitcoin
4. Cos'è e come funziona il mining di Bitcoin
5. Cosa determina il valore di Bitcoin

Essendo un argomento relativamente nuovo, la prevalenza dei materiali utilizzati sono stati raccolti in rete da fonti che si sono dimostrate affidabili, quali Bitcoin Wiki e Blockchain.com, da uno dei libri più diffusi all'interno della community di Bitcoin: *Mastering Bitcoin* di Andreas Antonopoulos, e da alcuni videocorsi online quali *Blockchain Deep Fundamentals* di Ivan On Tech.

Capitolo 1 – Bitcoin e Blockchain

1.1 La nascita di Bitcoin

"*Ho lavorato su un nuovo sistema di contanti elettronici che è completamente peer-to-peer, senza terze parti di fiducia",* il 31 ottobre 2008, Satoshi Nakamoto ha annunciato così il design paper di Bitcoin in una mailing list per crittografi chiamata "*metzdowd"*[1]. Da questo momento in poi è nato il concetto di Bitcoin. Nel suddetto paper, noto come White Paper e denominato "*Bitcoin: A Peer-to-Peer Electronic Cash System*"[2], vi è la prima descrizione dettagliata del protocollo di Bitcoin. In esso si possono evidenziare alcune caratteristiche fondamentali per il corretto funzionamento del sistema:

1. Consente di inviare direttamente da una persona all'altra pagamenti online senza passare per un intermediario.
2. Impedisce la doppia spesa (*double-spending*) tramite una network peer-to-peer.
3. Genera un registro formato da una catena di transazioni crittografate tramite una funzione Hash e riunite in blocchi. Questi vengono validati da un meccanismo di prove di lavoro (*Proof-of-Work*).
4. La catena di transazioni più lunga è considerata quella valida da tutti i nodi del network.
5. I nodi del network sono liberi di sganciarsi e ricollegarsi a propria volontà in modo anonimo.

Questi cinque elementi sono gli elementi fondamentali per la costituzione e il corretto funzionamento di Bitcoin e sono alla base del concetto di Blockchain. Il termine Blockchain deriva dalla struttura del database costituito da catene di blocchi di

[1] metzdowd, November 2008 Archives - https://www.metzdowd.com/pipermail/cryptography/2008-November/date.html

[2] Satoshi Nakamoto: "Bitcoin: A Peer-to-Peer Electronic Cash System" - https://bitcoin.org/bitcoin.pdf

transazioni. Tramite la Blockchain si istituisce un database distribuito fra tutti i nodi all'interno della rete. Il database di transazioni viene reso pubblico e i dati già inseriti all'interno al suo interno saranno irreversibili e non modificabili senza il consenso della maggioranza dei nodi. Questo semplice, ma rivoluzionario meccanismo ha permesso la creazione della prima moneta digitale in grado di risolvere il problema della doppia spesa senza la necessità di un intermediario. La prima versione del software di Bitcoin venne rilasciato il 9 Gennaio 2009 sempre da Satoshi Nakamoto, Bitcoin v0.1[3]

1.2 Come Bitcoin impedisce la *double-spending*

Il problema della doppia spesa è tradizionalmente risolto da una Trusted Third Party (TTP). Una TTP è un certificatore dell'avvenuta transazione di denaro che permette alle controparti che fanno affidamento sulla TTP a non correre il rischio di pagare due volte. Un esempio di TTP è Paypal che ha il pieno controllo del database delle transazioni degli utenti e di fatto si sostituisce all'altra parte (il compratore paga Paypal che a sua volta paga il venditore). Quindi gli utenti perdono il controllo dei propri fondi a beneficio della TTP. La TTP, controllando il database, può decidere in ogni momento di cancellare una transazione o di bloccare i fondi di un determinato utente se lo ritiene necessario, circostanza sempre più ricorrente[4].

Con il sistema ideato da Satoshi Nakamoto, le persone sono in grado di spendere la propria moneta digitale in maniera irreversibile, e in modo tale che chiunque sia in grado di verificare la validità di ogni transazione. Ogni bitcoin, cioè l'unità di conto della blockchain di Bitcoin, viene trasferito da un utente ad un altro in maniera *peer-to-peer*. I trasferimenti sono effettuati tramite un sistema crittografico che permette di verificare la proprietà dei bitcoin. Per mezzo di questo meccanismo si forma una catena di

[3] Bitcoin v0.1 released - http://satoshinakamoto.me/2009/01/09/bitcoin-v0-1-released
[4] Google Search: "Paypal blocked my account" - https://www.google.com/search?q=paypal+blocked+my+account&oq=paypal+blo

transazioni che chiunque può andare a controllare o verificare e che, tramite un sistema di validazione delle transazioni diventa immutabile.

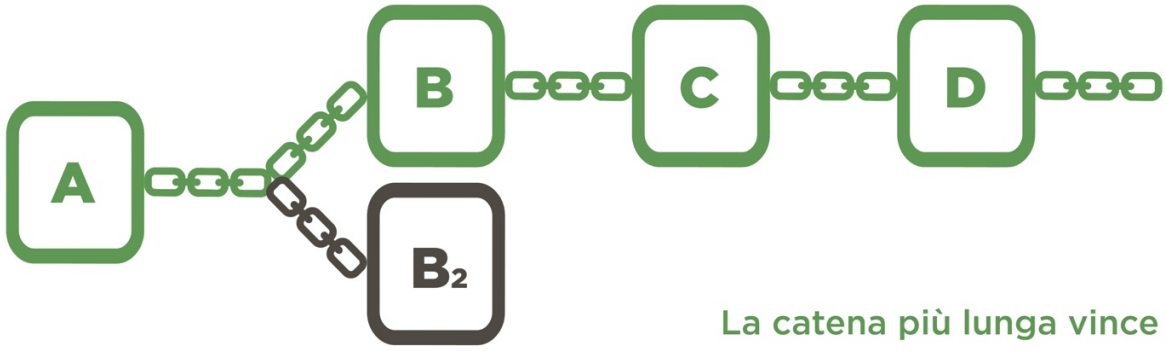
1.3 Blocchi di transazioni e *Proof-of-Work*

I blocchi della Blockchain sono dei contenitori di dati che raccolgono le informazioni sulle transazioni passate. Per generare un nuovo blocco è necessario risolvere un problema matematico (*Proof-of-Work*), utilizzando della potenza computazionale e consumando quindi energia elettrica. Chiunque riesca a risolvere per primo il problema riceve come ricompensa dei nuovi bitcoin appena generati dalla rete stessa. Questa ricompensa è anche nota come *coinbase***.** La difficoltà per risolvere il problema varia in base al numero di nodi che stanno tendando di risolverlo. Questa variazione è determinata in modo automatico dalla rete ed in modo tale che siano necessari circa 10 minuti per risolvere il problema. I nodi che si occupano della validazione dei blocchi sono meglio noti come *miners* (minatori). Questa denominazione deriva dal paragone spesso utilizzato fra la risoluzione del problema matematico e il lavoro svolto dai minatori d'oro. I miners di bitcoin consumano elettricità e potenza computazionale per essere ricompensati in bitcoin, mentre i minatori d'oro usano le proprie risorse e le proprie energie per trovare i giacimenti d'oro. L'obiettivo principale del processo di validazione di Bitcoin, o *Bitcoin mining*, è quello di permettere ai nodi della rete di raggiungere un consenso sulle transazioni completate nel passato, in maniera sicura e a prova di manomissione (*tamper*-resistant).

1.4 La catena più lunga è quella valida

La regola della catena più lunga è fondamentale affinché i nodi raggiungano il consenso sulle informazioni contenute all'interno della blockchain. Secondo questa regola, in caso di conflitti, i nodi devono sempre considerare legittima la blockchain che contiene più informazioni in quanto sarà quella che è stata validata più volte. Un

conflitto può nascere quando due miners riescono a risolvere il problema matematico quasi contemporaneamente. In questo caso si vanno a produrre due catene entrambe considerate valide dai nodi ma con diverse informazioni all'interno dell'ultimo blocco. Per risolvere il conflitto si applica la regola della catena più lunga: date queste due catene, la prima alla quale un miner riuscirà a generare un nuovo blocco verrà considerata quella valida da tutti i nodi, anche dai nodi che prima detenevano le informazioni della catena concorrente.



La catena più lunga vince

1.5 La pseudo-anonimità di Bitcoin

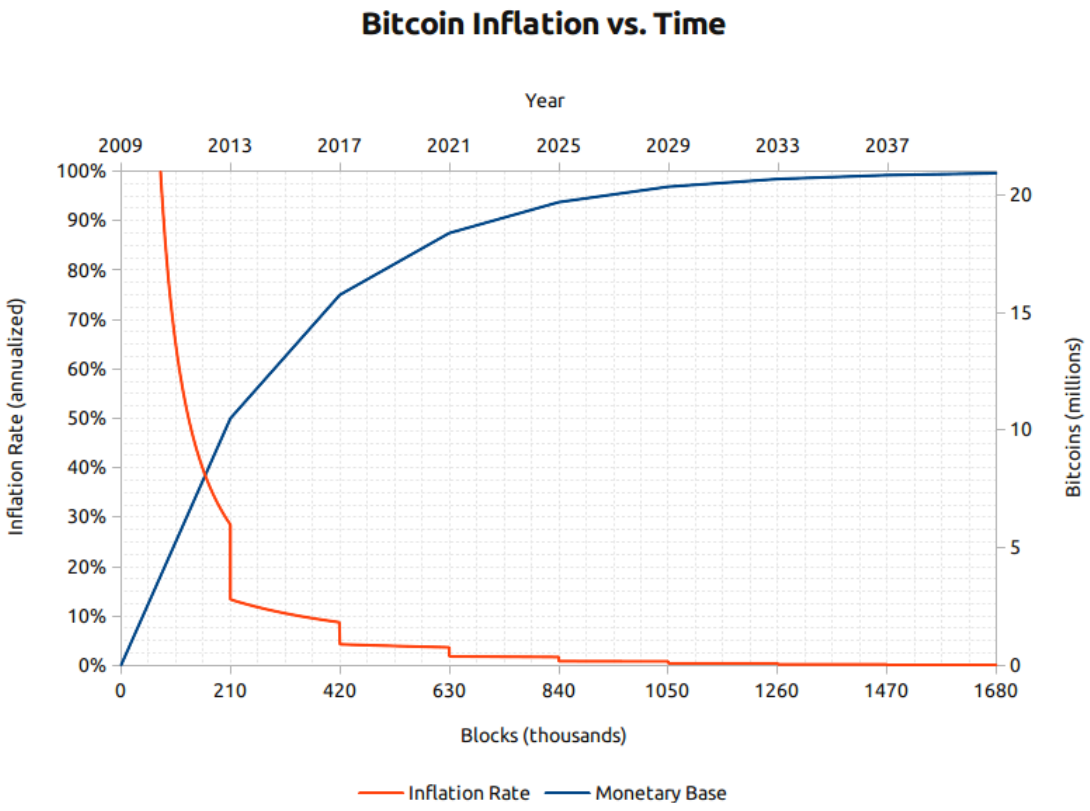
La natura decentralizzata di Bitcoin si basa su sistema *permissionless* dove non è necessario alcun permesso per far parte della rete ma chiunque può diventare un nodo e iniziare a validare le transazioni senza nessun processo identificativo. I nodi non sono altro che computer che eseguono il software di Bitcoin, e che detengono una copia della blockchain con tutte le sue informazioni. Il meccanismo di proprietà dei bitcoin si basa su un sistema di Public Key e Private Key (Cap 3.4 – Public Key Infrastructure). Tramite questo sistema è possibile verificare la proprietà dei propri fondi senza dover rilasciare alcuna identificazione, quindi in maniera anonima. Poiché

il database di transazioni è pubblico, chiunque può controllare quali transazioni siano state effettuate. In questo caso, però, sarà possibile vedere soltanto che un determinato indirizzo Bitcoin, composto da una stringa alfanumerica, ha inviato un importo di bitcoin ad un altro indirizzo. Non esiste alcun modo di collegare il proprietario persona fisica ad un indirizzo, se il primo non ne abbia fatto volontariamente pubblicità notizia. Per questo motivo il modello di proprietà di Bitcoin viene definito come *pseudo-anonimato*.

Capitolo 2 – Le basi del funzionamento di Bitcoin

2.1 Bitcoin halving

Come già spiegato, i bitcoin vengono generati circa ogni 10 minuti tramite il processo di mining, cioè di validazione delle transazioni. Il miner che riesce a validare il nuovo blocco, riceve come ricompensa per il suo lavoro la coinbase. Al momento della stesura di questa tesi, Giugno 2019, per ogni blocco verificato si riceve un compenso pari a 12,5 bitcoin (BTC). Ogni 4 anni il compenso viene dimezzato in un processo noto come *Bitcoin halving*, infatti, nei primi 4 anni di vita di Bitcoin, i miners ricevevano 50 BTC per ogni blocco validato. Il tasso di generazione dei nuovi bitcoin è quindi in continua decrescita, e continuerà a decrescere finché non si sarà raggiunto il limite massimo di bitcoin in circolazione. Questo limite, pari a 21 milioni di bitcoin, verrà raggiunto nel 2140 quando i miner non saranno più ricompensati con la coinbase. Già nel 2019, più della metà dei bitcoin sono stati immessi in circolazione.

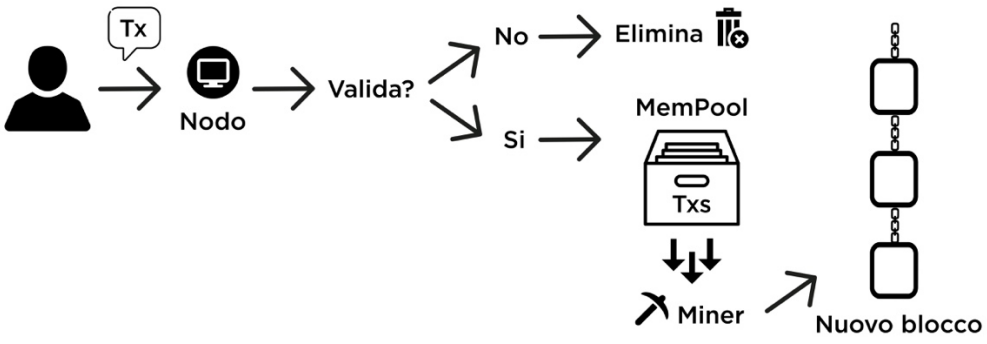


Bitcoin Inflation vs. Time - https://bitcointalk.org/index.php?topic=130619.0

2.2 Dimensioni dei blocchi, MemPool e commissioni

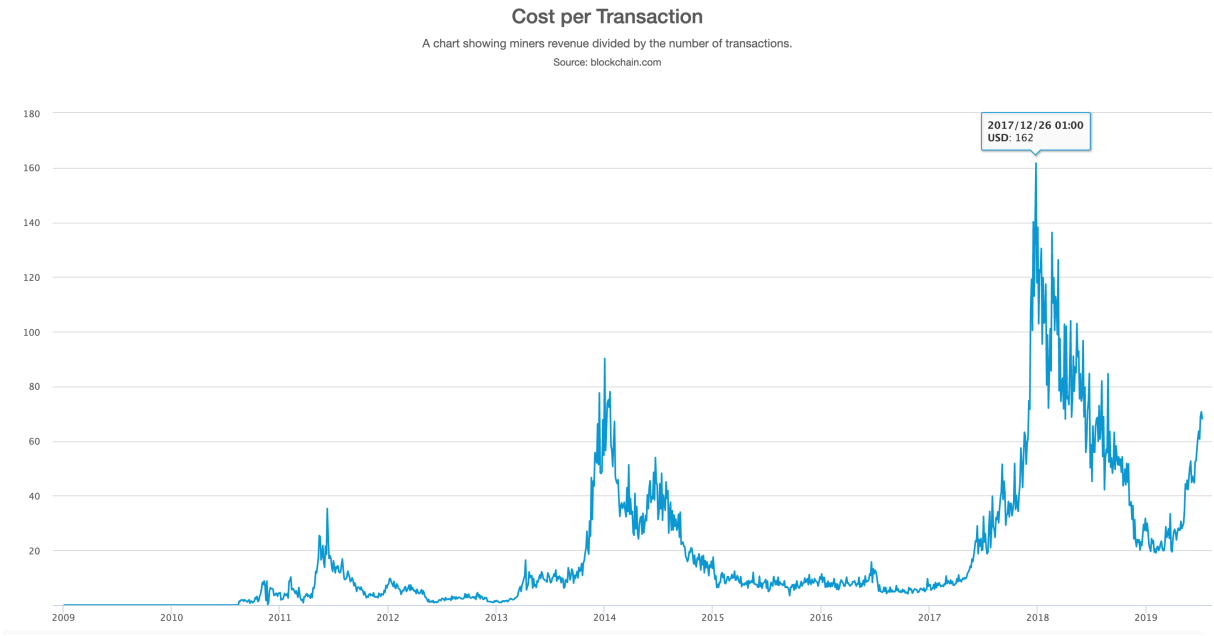
Ogni bitcoin è divisibile fino a 8 decimali, dove l'ultimo decimale è nominato *Satoshi*, in onore dell'ideatore del protocollo. Grazie alla loro natura digitale, i bitcoin sono facilmente scambiabili in mercati online, meglio noti come *Exchanges,* ad un valore determinato puramente dalla domanda e dall'offerta. Attualmente, 25 Giugno 2019, la capitalizzazione di bitcoin ha superato i 200 miliardi di dollari[5], e ogni bitcoin viene scambiato per un prezzo superiore a 11.000 dollari.

Oltre al compenso derivante dalla coinbase, i miners vengono ricompensati anche dalle commissioni pagate dagli utenti che vogliono confermare le proprie transazioni. Ogni blocco può contenere un numero limitato di transazioni, in quanto le dimensioni del blocco non devono superare 1 MB. A causa di questo limite, non tutte le transazioni che gli utenti hanno inviato ai nodi vengono immesse immediatamente dentro la Blockchain. Nel momento in cui un utente invia una transazione, questa viene innanzitutto depositata sul MemPool (*Memory Pool*), un contenitore di tutte le transazioni non ancora confermate. Dal MemPool i miners selezionano quelle che inseriranno all'interno del prossimo blocco da validare. Solitamente privilegiano le transazioni con le commissioni più elevate, a discapito delle altre anche se trasmesse in precedenza. Per questo motivo, alcuni utenti si troveranno ad aspettare più di 10 minuti prima di vedere la propria transazione confermata e aggiunta al database distribuito.



[5] Coinmarketcap - https://coinmarketcap.com/

L’importo delle commissioni viene stabilito direttamente dagli utenti, al pari di una mancia, e varia in base allo stato della rete. Per esempio, nel caso in cui la rete sia congestionata da un numeroso ammontare di transazioni, le commissioni tenderanno ad aumentare, e gli utenti che non vogliono pagare mance troppo elevate dovranno aspettare anche più di un’ora per vedere le loro transazioni confermate. Nel 2017 un utente ha pagato fino a $162 per una singola transazione.



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

Capitolo 3 – Funzione Hash, Merkle Tree e Public Key Infrastructure

Con la continua crescita in adozione e notorietà di Bitcoin, sempre più persone si stanno avvicinando a concetti che erano conosciuti solo dai più esperti. Concetti quali:

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

3.1 Funzioni Hash

Uno dei componenti principali della tecnologia Blockchain applicata al protocollo Bitcoin è la crittografia Hash. Una funzione hash è una funzione che permette di trasformare delle informazioni binarie in un'unica stringa alfanumerica di lunghezza prefissata. Possiamo individuare due principali proprietà che caratterizzano queste funzioni:

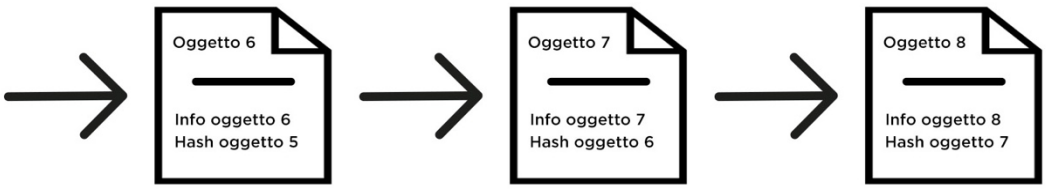
1. Sono unidirezionali, cioè una volta generato un hash non è possibile ricavare le informazioni che lo hanno costituito.
2. Sono unici, è praticamente impossibile generare due stringhe uguali da input di informazioni diverse. Modificando anche un solo valore all'interno dell'input, si va a generare un output completamente diverso. Possiamo quindi definire una funzione hash come un'impronta digitale dell'oggetto utilizzato come input, oggetto che può essere per esempio un documento, un film, musiche o qualsiasi altro file costituito da dati binari.

Vi sono varie funzioni hash con gradi di sicurezza diversa determinato dalla difficoltà di recuperare l'input dall'output. Nel protocollo Bitcoin viene utilizzato il SHA256 che permette di generare stringhe alfanumeriche dalla lunghezza di 64 caratteri. Per esempio, la frase: "*Amo Bitcoin"* viene trasformato nella stringa:

“e84e38b1b9d57f5e13867cc955bd146bc1704550195540ff71d6d4ebccd97bc4”. Nel momento in cui volessimo modificare anche solo una lettera della frase principale, otterremmo una stringa hash completamente diversa. Infatti, la frase: *“Am**a** Bitcoin”* viene trasformata in:
“f8c8d9dbd1633b764adfb02ed1ca1836a414021ab2335b41fbb969756e31a6bb”. Non c’è alcun modo di recuperare le due frasi iniziali dagli output finali, per questo motivo questi possono essere utilizzati come prove di integrità delle informazioni utilizzate come input. Infatti, chiunque in possesso di una stringa hash e del suo input, potrà verificare in qualunque momento che il secondo non sia stato modificato dopo che il primo è stato reso pubblico. Quest’ultimo concetto è fondamentale per generare una blockchain che sia temper-resistant, cioè resistente alla manomissione.

3.2 Funzione Hash applicata alla blockchain

Le funzioni hash possono essere utilizzate anche per creare delle strutture di dati annidate. Utilizzando una combinazione fra un oggetto e una stringa hash già esistente, è possibile creare una catena di informazioni rappresentate da un unico hash.

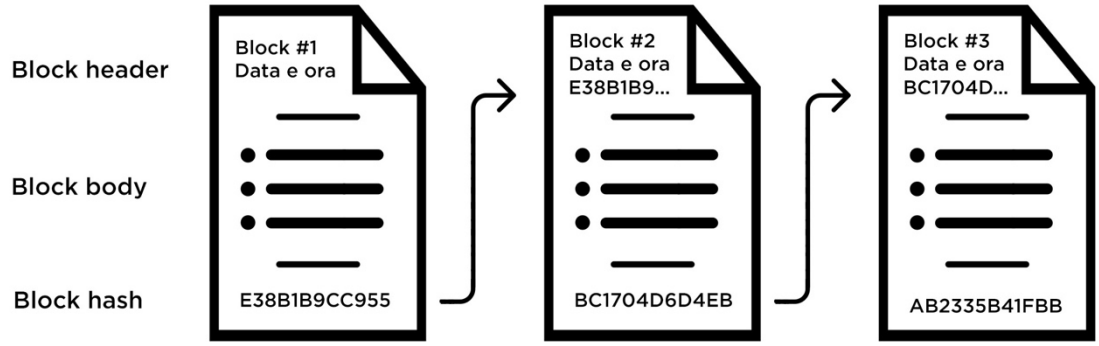


Riprendendo l’esempio mostrato nell’immagine, se si modificassero le informazioni contenute nell’oggetto 6, si andrá a modificare necessariamente anche l’hash dell’oggetto 7 e dell’oggetto 8. In questo modo, l’output dei nuovi oggetti é sempre dipendente dagli oggetti precedenti, inoltre è possibile verificare che le informazioni non siano state modificate semplicemente guardando se si è modificato l’ultimo hash.

La blockchain riprende la struttura di queste catene, dove gli oggetti sono costituiti dai blocchi di transazioni ed ogni blocco è composto da due parti distinte:

1. L'intestazione del blocco o *Block header*
2. Il corpo del blocco o *Block body*

Il block header contiene la stringa hash del blocco precedente e altre informazioni, quali la data e l'ora della creazione del blocco. Il block body contiene invece le transazioni raccolte dai miners selezionate dal MemPool. Il block header e il block body vengono utilizzati congiuntamente come input per generare una nuova stringa hash, nota come il *block hash*, che viene immessa all'interno del block header del blocco successivo. In questo modo è possibile riprodurre tutte le proprietà della catena di stringhe hash per creare un database di transazioni organizzate in blocchi temporali, persistenti e tamper-evident.



3.3 Riordinare le transazioni in un Merkle Tree

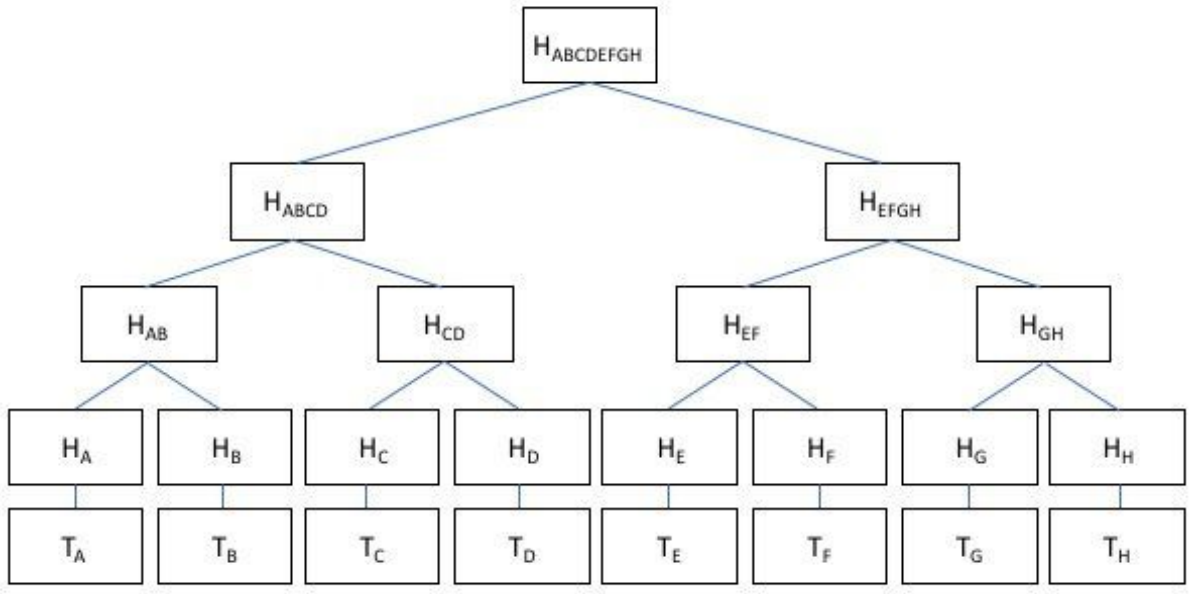
Il block body può contenere un gran numero di transazioni che dipende da 4 principali fattori:

1. Limite alle dimensioni del blocco, che può essere al massimo di 1MB
2. Dimensioni di ciascuna transazione, solitamente una transazione elementare pesa all'incirca 250 byte.
3. Dal numero di transazioni richieste negli ultimi 10 minuti
4. Dal numero di transazioni all'interno del MemPool

Per riuscire a gestire al meglio tutte le transazioni all'interno di ciascun blocco, queste vengono organizzate secondo una struttura detta Merkle Tree. Un Merkle Tree è una struttura "ad albero" composta da stringhe hash e che prevede un particolare procedimento.

1. Ogni transazione all'interno di un blocco viene trasformata in una stringa hash
2. Questi vengono riuniti in coppie da due, e la coppia stessa viene convertita in una nuova stringa
3. Si continuano a creare nuove stringhe in questo modo finché si genera un unico output hash, noto come *Merkle Root* o *Radice Merkle*.

Il Merkle Root rappresenta l'impronta digitale dell'insieme di tutte le transazioni contenute dentro al blocco, e presenterà tutte le caratteristiche degli hash e delle catene di hash. Quindi se si modificasse anche una singola transazione del blocco, la radice muterà completamente. Tramite questa struttura, si rende la gestione dell'insieme di transazioni molto meno dispendiosa, oltre a facilitare il controllo di eventuali manomissioni.



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

3.4 Public Key Infrastructure (PKI)

Il meccanismo di proprietà dei bitcoin si basa su un sistema di Public Key e Private Key. Questo sistema, noto come Public Key Infrastructure (PKI), è composto da:

1. Una funzione in grado di generare una coppia di chiavi (*Key pair*)
2. Un algoritmo per firmare utilizzando la coppia di chiavi
3. Una funzione in grado di verificare che la firma sia corretta

La coppia di chiavi è composta da una chiave privata (*Private Key*) e una chiave pubblica (*Public Key*). Come lo stesso termine suggerisce, la public key è la parte che viene resa pubblica, al pari del codice IBAN di un conto corrente, mentre la private key sarà il codice PIN che permette di gestire il conto corrente. Tutti i dati criptati tramite la public key possono essere decriptati solo tramite la corrispondete private e non vi è alcun modo di recuperare quest'ultima conoscendo la prima. Nel protocollo Bitcoin, la public key viene utilizzato per generare un indirizzo Bitcoin necessario per ricevere transazioni o richiedere pagamenti. La private key costituisce una prova di proprietà dei bitcoin contenuti all'interno di un indirizzo Bitcoin ed è l'unico mezzo di accesso ai fondi di un indirizzo Bitcoin. Se si perde la propria chiave privata, i bitcoin contenuti nell'indirizzo non potranno essere mai più utilizzati e vengono considerati "bruciati" (*burned*). A giugno 2017, si stimavano persi per sempre circa 4 milioni bitcoin[6].

[6] Martin YK Li: "How Much Bitcoin Has Been 'Lost' Forever?" - https://seekingalpha.com/article/4082979-much-bitcoin-lost-forever

Capitolo 4 –Key Pairs, Indirizzi Bitcoin e Firme digitali

4.1 Come vengono generate le Private keys e le Public keys

Una private key è un numero a 256 bits scelto in modo casuale. È possibile creare una nuova private key semplicemente lanciando una moneta 256 volte e trascrivendo i risultati dei lanci. In questo modo si avrà una serie di "testa" e "croce" che corrisponderanno ai valori binari della private key. Una private key corrisponde quindi ad un numero compreso fra 1 e 2^{256} e viene rappresentato secondo la formattazione *Wallet Import Format*, o *WIF*, basata sulla codifica *Base58*.

La public key viene generata a partire da una private key tramite una moltiplicazione a curva ellittica (Elliptic curve multiplication). Tale moltiplicazione si presenta nel seguente modo:

$$K = k * G$$

Dove *k* corrisponde alla private key, *G* ad una costante nota come *Punto di Generazione*, e *K* alla public key generata. La moltiplicazione a curva ellittica è un'operazione irreversibile, cioè da una public key *K* è impossibile ritrovare la private key *k* che la ha generata. Questa caratteristica è fondamentale per il corretto funzionamento del sistema di firme digitali di Bitcoin, in quanto è possibile condividere la propria chiave pubblica senza rivelare la corrispondente chiave privata.

4.2 Indirizzi Bitcoin

Da una chiave pubblica è possibile generare un indirizzo Bitcoin composto da una serie di lettere e numeri. Gli indirizzi Bitcoin sono utilizzati per inviare e ricevere le transazioni dagli utenti al pari del codice IBAN di un conto corrente. Come già introdotto nel primo capitolo (Cap 1.4 – La pseudo-anonimità di Bitcoin), non esiste alcun modo di collegare il proprietario persona fisica ad un indirizzo, se il primo non ne abbia fatto volontariamente pubblicità. Un utente può infatti generare un molteplice numero di indirizzi Bitcoin e utilizzarne uno nuovo per ogni transazione. In questo

modo è in grado di inviare e ricevere transazioni mantenendo la propria privacy e contrastando in parte la trasparenza della Blockchain.

Anche gli indirizzi Bitcoin vengono generati tramite una funzione Hash unidirezionale. In questo caso la chiave pubblica viene trasformata tramite la funzione SHA256 e inseguito tramite la funzione RIPEMD160, in modo da generare un numero a 160 bits. Avremo quindi che:

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

Dove *K* corrisponde alla chiave pubblica e *A* alla versione criptata a 160 bits della public key. Dalla public key criptata *A*, si calcola un *checksum*, per verificare che la nostra chiave non sia stata manomessa. Per calcolare il *checksum* si procede tramite una doppia hash della public key *A*, e in seguito prendiamo in considerazione solo i primi 8 caratteri esadecimali dell'hash risultante. Avremo quindi che:

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

Il *checksum* viene aggiunto alla stringa esadecimale di *A*, il valore risultante viene codificato secondo la codifica *Base58*, e otterremo così il nostro indirizzo Bitcoin.

Con il seguente codice Python è possibile generare una nuova private key, con la corrispondete public key e l'indirizzo Bitcoin. A causa di possibili problemi di sicurezza, non consiglio a nessuno di utilizzare le chiavi derivanti dal seguente codice, ma di utilizzarle esclusivamente a scopo informativo.

```
1.  import secrets
2.  import codecs
3.  import ecdsa
4.  import binascii
5.  import hashlib
6.  import base58
7.
8.  print('----------------')
9.
10. #Prendiamo un numero random a 256 bits
11. bits = secrets.randbits(256)
12. print('Numero casuale:')
13. print(bits)
14.
15. print('----------------')
16.
17. bits_hex = hex(bits)
18. 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 = '123456789ABCDEFGHJKLMNPQRSTUVWXYZabcdefghijkmnopqrstuvwxyz'
    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 Firme digitali su Bitcoin

Il sistema di firme digitali di Bitcoin è possibile tramite la Public Key Infrastructure. Tale sistema consiste in uno schema matematico che permette di firmare un messaggio tramite una chiave privata, e di verificarne la firma per mezzo della corrispondente chiave pubblica. Nel caso di Bitcoin, il messaggio da firmare corrisponde con la transazione autorizzata dal proprietario dell’indirizzo Bitcoin. La funzione che permette di generare la firma si presenta nel seguente modo:

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

Dove *dA* corrisponde alla private key, *m* è il messaggio (transazione), F_{hash} è la funzione che permette di generare l’hash del messaggio, F_{sig} è la funzione che generare la firma, e *Sig* è l’output risultante. Tale output è costituito da due numeri, noti come *R* e *S* che, oltre all’hash della transazione e alla public key, sono richiesti per verificare la firma.

Capitolo 5 – Introduzione alla Proof of Work (PoW)

Affinché una Blockchain sia considerata valida, è necessario che tutti i nodi della rete raggiungano il consenso sulle transazioni confermate nel passato. Soltanto con il consenso di tutti i nodi è possibile avere la certezza dell'integrità delle informazioni contenute all'interno della Blockchain.

5.1 Consenso Distribuito: Byzantine Generals' Problem

Satoshi Nakamoto, tramite il protocollo Bitcoin, è riuscito a risolvere uno dei problemi che avversava sui sistemi informatici distribuiti: il problema dei generali bizantini o *Byzantine Generals' Problem*. Questo problema è stato esposto per la prima volta su un paper pubblicato nel 1982[7] e viene spesso utilizzato come analogia per rappresentare il consenso distribuito dei nodi di una rete. Pur restando anonimi e senza conoscere le reciproche intenzioni, i nodi devono essere in grado di mettersi d'accordo e raggiungere il consenso. Gli ideatori del paper hanno proposto il problema nel seguente modo:

> "Immaginiamo che diverse divisioni dell'esercito bizantino siano accampate all'esterno di una città nemica, dove ogni divisione è comandata dal proprio generale. I generali possono comunicare l'uno con l'altro solo tramite un messaggero e dopo aver osservato il nemico, devono accordarsi su un piano d'azione comune. Tuttavia, alcuni dei generali possono essere dei traditori che cercano di impedire ai generali leali di raggiungere un accordo. Per risolvere il problema, I generali devono avere un algoritmo in grado di garantire che tutti i generali leali decidano sullo stesso piano d'azione e che un piccolo numero di traditori non può indurre i generali leali ad adottare un cattivo piano d'azione."[7]

[7] Leslie Lamport, Robert Shostak, and Marshall Pease - "The Byzantine Generals Problem": https://people.eecs.berkeley.edu/~luca/cs174/byzantine.pdf

Sostanzialmente ci si chiede come fare in modo che più soggetti, separati fisicamente e senza conoscersi, siano in grado di mettersi in assoluto accordo prima di intraprendere l'azione desiderata. Il 13 Novembre 2008, lo stesso Satoshi Nakamoto ha spiegato la propria soluzione nella stessa mailing list in cui ha annunciato l'ideazione di Bitcoin[8]. La soluzione di Nakamoto consiste nel provare l'esistenza di un accordo fra tutti i nodi tramite una prova di lavoro, o Proof-of-Work (PoW). I generali devono riuscire ad accordarsi sulla data e ora in cui dovranno attaccare, non è importante quando attaccheranno ma è importante che attacchino tutti insieme. Qualunque generale, tramite i suoi messaggeri, può proporre agli altri una data e un orario in cui attaccare. Solo la prima proposta ricevuta è considerata valida dal generale ricevente. Purtroppo, le informazioni vengono raggiunte in modo asincrono, cioè non è detto che tutti i generali ricevano la stessa informazione contemporaneamente. Questo problema viene risolto tramite una prova di lavoro basata sulla generazione di stringhe hash. Una volta che un generale riceve una proposta d'attacco, la utilizza per produrre un hash che deve presentare alcune caratteristiche predeterminate. Per riuscire a produrla con tali caratteristiche dovranno essere necessari all'incirca 10 minuti di lavoro congiunto fra tutti i generali. Una volta che un generale riesce a trovare l'hash con le caratteristiche predeterminate, la prova di lavoro si considera risolta e la soluzione viene trasmessa a tutti gli altri generali. I generali che ricevono la soluzione, la utilizzano per iniziare a generare un nuovo hash con le stesse caratteristiche predeterminate. Se qualcuno stava lavorando su un orario d'attacco diverso da quello condiviso tramite la soluzione, dovrà necessariamente iniziare a lavorare su un hash contenente la proposta ricevuta. Se in due ore i generali riescono a risolvere 12 Proof-of-Work generando 12 soluzioni diverse con lo stesso orario d'attacco, sarà dimostrato il consenso distribuito. Infatti, se per generare una

[8] metzdowd: "Bitcoin P2P e-cash paper", Satoshi Nakamoto Thu Nov 13 17:56:55 EST 2008 - https://www.metzdowd.com/pipermail/cryptography/2008-November/014849.html

soluzione erano necessari all'incirca 10 minuti utilizzando il lavoro complessivo di tutti i generali, in 2 ore di lavoro complessivo si dovrebbe aver risolto la PoW per 12 volte. A questo punto tutti i generali stanno lavorando sulla stessa proposta d'attacco, e la data e l'ora prestabilita sarà conosciuta da tutti. Si potrà quindi procedere all'azione, risolvendo il problema dei generali bizantini.

5.2 Consenso Distribuito fra i nodi di Bitcoin

Come nel caso dei generali bizantini, il consenso fra i nodi all'interno di Bitcoin viene raggiunto in modo asincrono, cioè i nodi non aspettano di sincronizzarsi fra di loro per stabilire quale blocco aggiungere al proprio database, ma scelgono in base alle informazioni che gli giungono disponibili singolarmente. La più importante fra queste è la soluzione della Proof-of-Work. Per aggiungere un nuovo blocco alla blockchain, i miners devono risolvere questo problema matematico che richiede potenza computazionale ed energia elettrica. Il primo miner che riesce a risolverlo, dovrà trasmettere la soluzione agli altri nodi, che una volta ricevuto, procederanno ad aggiornare il proprio database con le transazioni contenute all'interno del blocco validato. Una volta che il blocco è stato ricevuto da tutti i nodi, il database distribuito sarà stato aggiornato con le stesse informazioni e si sarà raggiunto il consenso distribuito.

Capitolo 6 – La matematica alla base della Proof-of-Work

6.1 Algoritmo Proof-of-Work

La Proof-of-Work consiste nel creare una stringa hash con le informazioni contenute all'interno del block header in modo tale che l'hash presenti determinate caratteristiche. Se si ha capito bene come operano le funzioni hash (Cap 3.1 – Funzioni Hash), si dovrebbe aver chiaro che uno stesso input genera sempre lo stesso output. Allora in che modo è possibile generare degli hash con determinate caratteristiche utilizzando le stesse informazioni contenute nel block header? Questo è possibile grazie ad un particolare oggetto: il *nonce*. Il nonce è un numero casuale che viene immesso all'interno del block header per generare nuove stringhe; soltanto modificando il nonce è possibile generare degli hash differenti utilizzando le stesse informazioni.

Un'altra importante caratteristica delle funzioni hash consiste nell'impossibilità di prevedere quale output si genera da un determinato input. Per questo motivo i miners, per generare una stringa con le caratteristiche desiderate, dovranno procedere solo a tentativi, rendendo la soluzione del problema matematico più simile alla vincita di una lotteria. Maggiore è la potenza computazionale utilizzata, più biglietti della lotteria si potranno acquistare e maggiore sarà la possibilità di vincere. Per esempio, la frase: "*Amo Bitcoin"* viene trasformato nella stringa: "e84e38b1b9d57f5e13867cc955bd146bc1704550195540ff71d6d4ebccd97bc4". Se volessimo invece produrre una stringa hash diversa che inizi con uno "**0**" e utilizzando la stessa frase, l'unica soluzione sarà aggiungere un *nonce.* In questo modo, l'input che verrà utilizzato per generare il nuovo hash sarà composto da: "Frase" + "nonce". Per esempio, la frase "Amo Bitcoin**0**", produce l'hash: "5c5f5a0c9c7c9b4b958d9cf79c0d140c2310e2ae4fbdff1a883d714a2b734bc1".

Continuando a generare nuove stringhe con altri nonce, avremo i seguenti risultati:

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

Ecco che dopo 4 tentativi, l'input "Amo Bitcoin**3**" genera un hash che inizia con uno zero, esattamente come avevamo richiesto.

6.2 Target e hash value

Nel caso di Bitcoin, le caratteristiche necessarie affinché un hash venga considerato valido sono determinate dal *target*. Il target è un valore che non deve essere superato dall'hash di un nuovo blocco affinché si consideri risolta la PoW. In sostanza, i miner utilizzano le informazioni contenute nel block header per generare l'equivalente stringa hash, se il valore della stringa è superiore al target, il miner cambierà nonce per generare un nuovo hash con un valore diverso, finché quest'ultimo risulti inferiore al target. Un output generato dalla funzione SHA256 genera un valore a 256-bit che viene espresso in un numero esadecimale. Il sistema dei numeri esadecimali è un sistema basato su 16 caratteri, a differenza di quello comunemente utilizzato basato su 10 caratteri (da 0 a 9). I caratteri utilizzati nel sistema esadecimale sono i numeri da 0 a 9 e lettere dalla A alla F. Tramite questo sistema numerico è possibile rappresentare valori elevati utilizzando un numero contenuto di caratteri, per esempio, è possibile esprimere valori fino al 255 con solo 2 caratteri.

Per mezzo di un linguaggio di programmazione come Python, si è in grado di convertire semplicemente numeri esadecimali in numeri decimali. Per esempio, riprendendo l'hash di "Amo Bitcoin3":

0187efb67ca570f679e123317aa10809654fd6bb960c45f407660ea5acfb882a

E utilizzando il codice:

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

Riusciamo a ricavare il valore decimale dell'hash, che in questo caso è equivalente a:

$$6.92E + 74$$

6.3 Rappresentazione del target e target value

Per quanto riguarda il valore target, questo è contenuto all'interno del block header di ogni blocco. La sua rappresentazione è conosciuta come *bits* o *target bits,* e consiste in un numero esadecimale a 8 caratteri. Affinché il blocco sia considerato valido e aggiunto alla blockchain, il valore del target deve risultare superiore rispetto al valore del blocco. Per esempio, l'ultimo blocco validato nel momento in cui sto scrivendo questa tesi, è il blocco numero #584362. Andando a controllare in un Block Explorer, cioè un sito internet connesso alla blockchain che permette di visualizzare i blocchi e le transazioni, è possibile conoscere in maniera semplice il *target bits* di questo blocco. In questo caso, secondo blockchain.com[9], il target bits del blocco #584362 corrisponde al valore decimale 388200748, che con il sistema esadecimale equivale a 1723792c. I primi due valori del target bits esadecimale, vengono chiamati *esponente*, mentre i successivi 6 caratteri sono noti come *coefficiente*. Una volta a conoscenza dell'esponente e del coefficiente, è possibile determinare il target del blocco. Il target sarà equivalente a:

[9] Blockchain.com, Bitcoin blocco #584362 - https://www.blockchain.com/btc/block-height/584362

$$target = coefficiente * 2\^(8 * (esponente - 3))$$

Nel caso del blocco #584362, il target corrisponderà a:

$$target = 0x23792 * 2\^(0x08 * (0x17 - 0x03))$$

In numeri decimali, avremo:

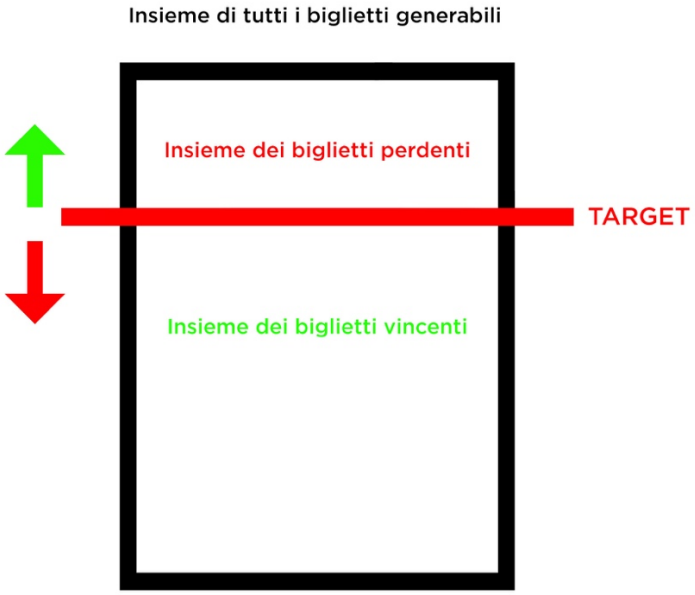
$$target = 2324780 * 2\^160$$

$$target = 3.40E + 54$$

Quindi, il primo miner che è riuscito a generare un blocco con un hash di valore inferiore a 3.40E+54, è riuscito a validare il blocco e a ricevere la ricompensa in bitcoin.

6.4 Chiarimenti sul target e sulla Difficoltà

Riassumendo, il target è il valore massimo che può avere un hash di un blocco, affinché questo venga considerato valido dalla rete. Tanto più elevato è il target, tanto più sarà semplice generare blocchi validi. Possiamo quindi affermare che minore è il target, maggiore sarà la difficoltà per i miner di validare un blocco, e maggiore sarà il consumo di risorse. Come già introdotto, poiché non è possibile prevedere le caratteristiche di una stringa hash date le informazioni utilizzate per generarla, tutto il processo di mining può essere rappresentato come una lotteria. I miner utilizzano le proprie risorse, per acquistare dei biglietti numerati di questa lotteria, finché non viene proclamato un vincitore. Il numero di ogni biglietto corrisponde con il valore dell'hash generato tramite le informazioni del blocco e il nonce. Il vincitore sarà colui che trova per primo un biglietto il cui numero è inferiore rispetto al numero del target.



La Difficoltà per minare un nuovo blocco viene rappresentata come il rapporto fra il valore del target del primo blocco della blockchain, o Genesis Block, e il target del blocco che si vuole confrontare. Quindi avremo che:

$$Difficoltà = \frac{Genesis\ Block\ Hash\ Value}{Comparison\ Block\ Hash\ Value}$$

Riprendendo il caso del blocco #584362, e sapendo che l'hash del blocco genesi corrisponde a 0x00000000ffffffffffffffffffffffffffffffffffffffffffffffffffffffff[10], avremo:

$$Difficoltà = \frac{0x00000000ffffffffffffffffffffffffffffffffffffffffffffffffffffffff}{0x000000000000000023792c00000000000000000000000000000000000000000}$$

$$Difficoltà = 7.93E + 12$$

La difficoltà indica quanto sia più dispendioso validare un nuovo blocco rispetto alle risorse utilizzate per validare il Blocco Genesi. Ad oggi è 7.93 trilioni di volte più dispendioso validare un nuovo blocco. La difficoltà è quindi una rappresentazione del target più comprensibile all'uomo. Per semplificare ulteriormente il concetto di target e capire se un blocco sia valido o meno, si fa spesso riferimento al numero di 0 iniziali all'interno della rappresentazione binaria dell'hash del blocco da validare. Sostanzialmente, sia il target sia l'hash del blocco sono dei numeri a 256 bits, quindi

[10] Blockchain.com, Bitcoin, blocco #1 - https://www.blockchain.com/btc/block-height/1

sono composti da una serie di 256 valori binari, cioé composti da degli zero e degli uno. Tanti più zero iniziali contiene la rappresentazione binaria, tanto minore sarà il valore del target o dell'hash e per questo motivo se l'hash contiene un numero di zero iniziali maggiore rispetto a quelli contenuti dal target, possiamo essere sicuri che il valore il suo valore sia minore al valore del target e che il blocco potrà essere considerato valido.

Con il seguente codice Python è possibile calcolare il valore del target di un blocco e la sua difficoltà inserendo soltanto il corrispondente target bits espresso secondo il sistema numerico decimale:

```
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 = '0x00000000FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF
    F'
30.
31. original_value = int(original_target, 16)
32.
33. print('Original value:', '%.2E' % Decimal(original_value))
34.
35. difficulty = int(original_target, 16) / int(hex_target,16)
36.
37. print('Difficulty:', '%.2E' % Decimal(difficulty))
```

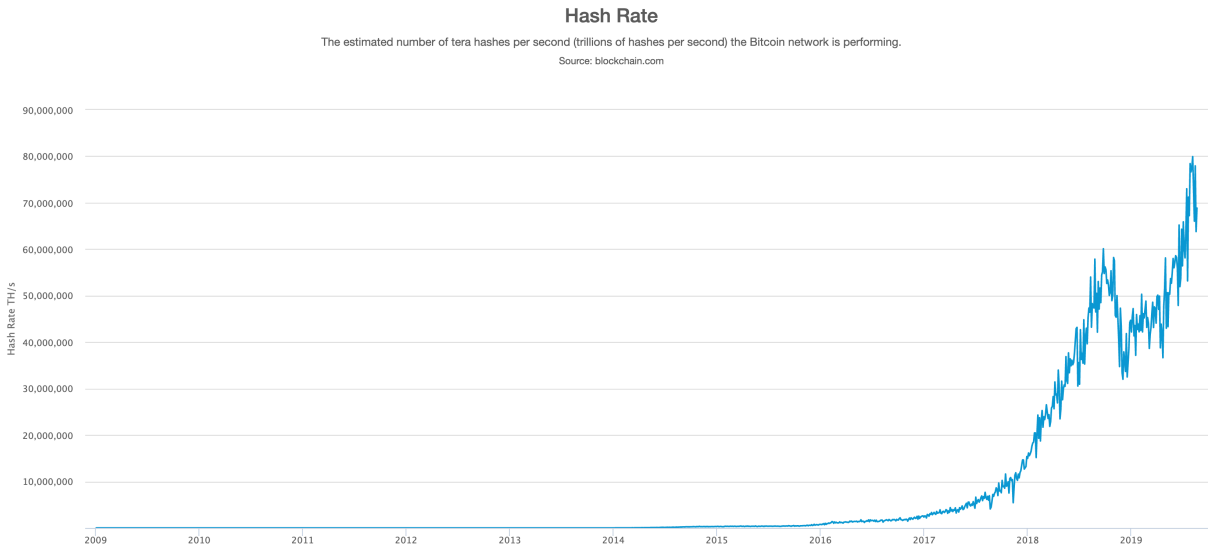
6.5 Ricalibrazione del target

Il target stabilisce la difficoltà per un miner di validare un nuovo blocco. Riprendendo l'analogia sulla lotteria, stabilisce la percentuale del numero di biglietti vincenti sul numero totale di biglietti acquistabili. Un miner può verificare un solo hash alla volta, che equivale ad acquistare un biglietto della lotteria alla volta. Però tanto maggiore è la potenza computazionale del miner, tanto più velocemente sarà in grado di generare nuovi hash, quindi di acquistare nuovi biglietti. Per questo motivo un miner con una potenza computazionale maggiore avrà più possibilità di generare un blocco valido e di vincere la lotteria, quindi di ricevere la ricompensa in bitcoin. Un miner con una potenza computazionale minore, potrà comunque essere in grado di vincere la lotteria, se con un po' di fortuna riesce a trovare per primo un biglietto vincente. Più il target è alto, più biglietti vincenti saranno disponibili e conseguentemente la difficoltà sarà bassa. Dato un valore del target costante, al crescere della potenza computazionale, il tempo per vincere la lotteria si riduce. In questo caso infatti, il numero di biglietti vincenti rimane uguale, ma il numero di biglietti acquistati nello stesso periodo di tempo aumenta, aumentando la probabilità di trovare un biglietto vincente in minor tempo. Come già spiegato nei capitoli precedenti, nel protocollo Bitcoin un blocco viene validato all'incirca ogni 10 minuti, portando alla generazione di nuovi bitcoin. Se il target dovesse rimanere costante, all'aumentare della potenza computazionale dei miner, il tempo di validazione di un blocco e il tempo di generazioni di nuovi bitcoin si ridurrebbe. Per fare in modo di mantenere stabile questo periodo, pur avendo variazioni della potenza computazionale totale dei miner, il protocollo Bitcoin prevede una ricalibrazione del target. Questa ricalibrazione viene effettuata automaticamente ogni 2016 blocchi validati, quindi ogni 20160 minuti circa, che equivalgono a 2 settimane. Il target viene ricalibrato in modo tale che i miners tramite la loro attuale potenza computazionale, riescano a validare un blocco ogni 10 minuti. Sostanzialmente, se nelle ultime due settimane in media sono stati richiesti più di 10

minuti per validare un blocco, il target verrà alzato per abbassare la difficoltà. Viceversa, se sono stati necessari in media meno di 10 minuti, il target verrà abbassato per aumentare la difficoltà e ristabilire l'equilibrio. Avremo quindi che:

$$Nuovo\ target = Vecchio\ target * (\frac{Tempo\ richiesto\ per\ validare\ gli\ ultimi\ 2016\ blocchi}{20160\ minuti})$$

La potenza computazionale viene misurata in base al numero di hash che i computer da mining sono in grado di generare e verificare ogni secondo, questa misura è nota come hashrate/s. Negli ultimi 10 anni, abbiamo assistito ad una crescita esponenziale dell'hashrate, che ad oggi raggiunge i 70 milioni terahash (TH/s) al secondo, dove 1TH/s equivale a 1,000,000,000,000 (un trilione) di hash generati al secondo.



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

Conclusione

L'obiettivo di questa tesi è quello di rispondere ad alcuni dei quesiti più diffusi di chi si approccia per la prima volta a Bitcoin. Gli argomenti sono stati volutamente incentrati sulle spiegazioni analitiche del funzionamento di questo nuovo sistema monetario, cercando di evitare gli eventuali effetti economici. Una volta provata l'efficacia dei meccanismi matematici su cui si basa questa tesi, ci si potrebbe concentrare sulle conseguenze che tutta questa struttura potrebbe apportare nell'economia, soprattutto in termini macroeconomici. Lo scopo di Bitcoin non è quello di essere un mezzo di investimento, ma un prodotto rivoluzionario e innovativo, al pari del motore a vapore, delle ferrovie, dell'acciaio o delle e-mail. La sua qualità di essere composto interamente da software, basato sulla matematica e sulla crittografia lo rende il primo strumento esistente che garantisce una via d'uscita dal controllo monetario dei governi senza la necessità di alcun permesso. Anche coloro che prima non potevano partecipare all'economia globale perché privi di qualsiasi accesso ai servizi finanziari, i cosiddetti "*unbanked*", possono ora farne parte senza alcuna restrizione. Anche se questi effetti non sono ancora evidenti nel mondo occidentale, si fanno già sentire in alcuni paesi, come il Venezuela e lo Zimbabwe, entrambi segnati da una forte iperinflazione. I cittadini di questi due paesi hanno compreso immediatamente il potenziale di Bitcoin, e lo utilizzano all'oscuro dal proprio governo per proteggersi dalla svalutazione della moneta locale e come riserva di valore. Probabilmente la caratteristica più importante del sistema monetario digitale che Bitcoin garantisce è la sua libera disponibilità a chiunque. Si tratta di un sistema completamente aperto, trasparente, neutrale, verificabile, peer-to-peer e soprattutto libero da limiti geografici. Può essere trasferito ovunque nel mondo senza alcun vincolo aggiuntivo, a differenza di altri sistemi utilizzati da millenni, come l'oro e il contante. Come il *Times,* testata giornalistica inglese, ha fatto notare, questo porta a forti implicazioni nei confronti delle politiche monetarie che i governi potranno adottare.

“Se un futuro presidente della Fed cercasse di ripetere la politica di quantitative easing di Ben Bernanke (stampando effettivamente denaro), gli investitori preoccupati potrebbero iniziare a sganciare i loro risparmi dal dollaro e trasferirli in streaming nel cloud così velocemente che la Fed sarebbe costretta a cambiare rotta. […] Una volta che le valute alternative saranno disponibili su Internet senza attrito, ogni computer portatile diventerà la propria isola Cayman.”[11]

[11] TIME, Michael Sivy: "The Real Significance of the Bitcoin Boom (and Bust)" - http://business.time.com/2013/04/12/the-real-significance-of-the-bitcoin-boom-and-bust/

Bibliografia

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). *Tutto su Blockchain, Capire la tecnologia e le nuove opportunità.* Hoepli.

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

Sitografia

- 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 a Bitcoin: https://learnmeabitcoin.com/
- Seeking Alpha: https://seekingalpha.com
- The Next Web: https://thenextweb.com
- Satoshi Nakamoto: http://satoshinakamoto.me