



**UNIVERSITÀ DEGLI STUDI DI ROMA  
TOR VERGATA**

MACROAREA DI SCIENZE MATEMATICHE FISICHE  
NATURALI

CORSO DI LAUREA IN INFORMATICA

A.A. 2018/2019

**Tesi di Laurea**

La dinamica della rete Bitcoin: analisi empirica del vicinato di  
un full-node

**RELATORE**

Dott. Francesco Pasquale

**CANDIDATO**

Marcello Politi

*"C'è vero progresso solo quando i vantaggi di una nuova tecnologia diventano per tutti."*

*Henry Ford*

# Indice

<b>1</b>	<b>Bitcoin</b>	<b>5</b>
1.1	Il mondo Bitcoin . . . . .	5
1.2	Cenni storici . . . . .	7
1.2.1	La storia del mining . . . . .	8
1.3	Acquistare bitcoin . . . . .	10
1.4	I wallet Bitcoin . . . . .	11
1.5	Utilizzo nella vita quotidiana . . . . .	13
<b>2</b>	<b>Come funziona il sistema Bitcoin?</b>	<b>15</b>
2.1	Crittografia . . . . .	15
2.1.1	Introduzione . . . . .	15
2.1.2	Crittografia su curva ellittica . . . . .	16
2.1.3	Indirizzi Bitcoin . . . . .	18
2.2	Le transazioni . . . . .	20
2.2.1	UTXO: output di transazione non spesi . . . . .	20
2.2.2	I fees di transazione . . . . .	21
2.2.3	Linguaggio di script delle transazioni . . . . .	22
2.3	La rete Bitcoin . . . . .	24
2.3.1	Architettura di rete peer-to-peer . . . . .	24

2.3.2	I ruoli dei partecipanti della rete Bitcoin . . . . .	24
2.3.3	Come fa un nuovo partecipante a collegarsi alla rete Bitcoin? .	26
2.3.4	Ricostruire la blockchain grazie al comando "Inventory" . . .	29
2.4	La Blockchain . . . . .	30
2.4.1	Il contenuto di un blocco della blockchain . . . . .	31
2.4.2	Identificazione dei blocchi attraverso header e height . . . . .	31
2.4.3	Collegare i Blocchi nella Blockchain . . . . .	33
2.4.4	Merkle Trees . . . . .	33
2.5	Mining per il consenso della rete . . . . .	35
2.5.1	Introduzione . . . . .	35
2.5.2	Consenso Decentralizzato . . . . .	36
2.5.3	Aggregare Transazioni nei Blocchi . . . . .	37
2.5.4	La Transazione Coinbase . . . . .	38
2.5.5	Minare il Blocco . . . . .	38
2.5.6	Algoritmo Proof-of-Work . . . . .	39
<b>3</b>	<b>Analisi delle connessioni di un singolo nodo</b>	<b>42</b>
3.1	Motivazioni . . . . .	42
3.2	Descrizione dell'applicativo . . . . .	44
3.3	Risultati ottenuti . . . . .	50
<b>4</b>	<b>Conclusioni e sviluppi futuri</b>	<b>53</b>
	<b>Elenco delle figure</b>	<b>55</b>

# Ringraziamenti

Desidero ringraziare innanzitutto il relatore di questa tesi, il dott. Francesco Pasquale per la disponibilità, l'attenzione e la gentilezza dimostrate durante la stesura del lavoro, ma soprattutto per il forte interesse che mi ha suscitato negli argomenti trattati durante le sue lezioni.

Inoltre vorrei ringraziare la mia famiglia, il mio punto di riferimento. In particolare mio padre che ha sempre creduto in me e con i suoi sacrifici mi ha permesso di raggiungere questo traguardo. Mia madre, che mi ha spronato e sostenuto in ogni momento, aiutandomi a trovare la grinta anche nelle circostanze più difficili. Mia sorella che, forse inconsapevolmente, mi trasmette continuamente la gioia e la spensieratezza e mi dà la certezza di essermi sempre vicino.

Ai miei compagni di studio e non, e a chi mi è stato sempre a fianco.

Un sentito grazie a tutti!

# Introduzione

Sempre un numero più alto di servizi internet basano la loro infrastruttura di rete su un'architettura peer-to-peer (P2P), la quale permette una notevole resilienza, flessibilità e rapidità di adattamento. Una delle sfide maggiori in questo tipo di reti è di garantire robustezza a prescindere da malfunzionamenti o attacchi esterni. La conoscenza della topologia della rete può essere la chiave sia di un miglioramento della performance del servizio che si poggia su di essa, sia di un attacco più mirato alla rete stessa, come ad esempio un *denial of service*. Inoltre l'anonimia potrebbe essere un obiettivo (e.g. *TOR*) che può venire compromessa se un hacker è in grado di ricostruire i *path* di comunicazione tra i nodi.

Il sistema di moneta digitale Bitcoin utilizza una rete P2P per trasmettere informazioni riguardanti le transazioni attraverso i partecipanti della stessa. Lo scambio di informazione segue il *gossip protocol*: se un *peer* riceve una nuova transazione, controlla che questa sia valida e la inoltra ai suoi vicini. A partire dalla sua creazione nel 2009, Bitcoin ha conquistato un'attenzione sempre maggiore, complice anche il *boom* osservato nel 2017, anno nel quale la valuta ha raggiunto una quotazione pari a circa 3.000. L'obiettivo principale di Bitcoin è quello di garantire trasferimenti di denaro sicuro che non si basino sulla fiducia in terze parti (come ad esempio una banca), e questo obiettivo viene raggiunto distribuendo una copia del database delle transazioni su ogni nodo del network, divenendo così la prima forma di pagamento *trustless*.

Il database, che prende il nome di *blockchain*, tiene traccia di tutte le transazioni avvenute nella rete dal 1 gennaio 2009.

La validità delle transazioni viene garantita da un meccanismo a firma digitale tramite chiave privata e pubblica, permettendo che solamente i reali possessori di una determinata somma possano spenderla. Le transazioni vengono verificate, prima di essere immesse nella blockchain, da nodi speciali, denominati *miners*. I *miners* svolgono il duplice ruolo di verifica della transazioni e di emissione dei bitcoin. La *blockchain* non viene aggiornata in *real time*, ma viene aggiunto un gruppo di transazioni, chiamato blocco, in media ogni 10 minuti . Ogni volta che tale blocco viene aggiunto, il sistema emette una quantità prestabilita di bitcoin , destinata al minatore che per primo è riuscito a verificarne la validità. Queste operazioni richiedono un notevole potere computazionale, perciò i minatori sono molti meno dei nodi totali della rete, ma il loro numero è comunque sufficiente a garantire la decentralizzazione della stessa. Un'ulteriore caratteristica di Bitcoin è che l'emissione dell'omonima valuta, denominata di seguito BTC, è limitata, e cresce nel tempo asintoticamente fino a raggiungere il limite prestabilito di 21 milioni di unità. In questo modo il valore non può essere manipolato tramite l'inflazione, come invece può accadere con le valute tradizionali. L'introduzione fin qui fatta è volutamente generica, poichè i dettagli tecnici verranno analizzati in seguito. Si è scelto di suddividere il lavoro di questa tesi in tre parti. Nel primo capitolo si vedrà una spiegazione generale di quello che è il mondo Bitcoin, come tale tecnologia è nata e si è diffusa, e come un utente comune può usufruirne. Nella secondo capitolo invece verrà fatta un'analisi più tecnica sulle principali innovazioni che hanno permesso di creare tutta l'infrastruttura che sta alla base del Bitcoin, mentre negli ultimi due capitoli mostreremo come con un applicativo, creato appositamente per questo fine, siamo riusciti a ricavare informazioni utili

da dati reali riguardanti le connessioni di un *full node* con i suoi *peer*.

## Scopo del lavoro

In questo lavoro mi concentrerò a studiare la topologia della rete Bitcoin, o almeno come questa viene osservata da parte di un singolo *full node*. Attraverso degli applicativi sviluppati appositamente per questo scopo, andremo a raccogliere dati che in seguito verranno elaborati allo scopo di estrarre informazioni utili riguardo le connessioni di un nodo verso altri *peer* del network Bitcoin. Il risultato di questo lavoro dovrà essere visto comunque come un passo intermedio per il raggiungimento di un obiettivo più ambizioso che punta ad inferire e a studiare le connessioni dell'intero network Bitcoin.



# Capitolo 1

## Bitcoin

### 1.1 Il mondo Bitcoin

Bitcoin è una collezione di concetti e tecnologie che costituiscono le basi per un ecosistema di una moneta digitale. L'unità di conto di questa valuta è il "bitcoin", scritto solitamente con la prima lettera minuscola o identificato dalla sigla **BTC** o **XBT**, e' usato per salvare o trasmettere la valuta tra i partecipanti della rete.

Per convenzione quindi se il termine Bitcoin è utilizzato con l'iniziale maiuscola si riferisce alla tecnologia e alla rete, mentre se minuscola (bitcoin) si riferisce alla valuta in sé. Gli utenti possono fare con Bitcoin quello che è possibile fare con una qualsiasi moneta convenzionale, l'unica differenza è che il bitcoin è interamente virtuale, non esistono monete fisiche ma solamente transazioni che possono trasferire valuta da un mittente verso un destinatario.

Gli utenti di Bitcoin possiedono delle *keys* (chiavi) <sup>1</sup> con le quali possono provare la loro proprietà di una somma di bitcoin all'interno del network. Con queste chiavi gli utenti possono firmare le varie transazioni per trasferire della valuta ad un nuovo

---

<sup>1</sup>chiavi private e chiavi pubbliche: le transazioni di bitcoin, come verrà analizzato nel secondo capitolo, si basano sulla tecnologia della crittografia asimmetrica (o crittografia a chiave pubblica/privata), in particolare sul meccanismo delle firme digitali.

proprietario.

Le chiavi sono solitamente memorizzate all'interno di un *wallet* che è un'applicazione per computer o smartphone che permette all'utente di effettuare operazioni complicate con molta facilità, offrendo a quest'ultimo un'interfaccia *user-friendly* <sup>2</sup>. L'unico requisito richiesto per poter spendere bitcoin è la chiave con la quale l'utente può firmare le varie transazioni.

Bitcoin è un sistema distribuito peer-to-peer (**P2P**) <sup>3</sup>, come tale non esiste un server "centrale" o un punto di controllo.

I bitcoin vengono creati attraverso un processo chiamato *mining*, un termine che ricalca metaforicamente l'attività di estrazione dell'oro da una miniera, il quale consiste in una competizione tra i partecipanti (i *full-node* della rete) che attraverso il loro potere computazionale cercano di trovare una soluzione ad un problema matematico mentre processano le transazioni. In media ogni 10 minuti un *miner* è in grado di trovare una soluzione a tale problema e a validare così delle transazioni, questo *miner* verrà ricompensato con dei nuovi bitcoin. Il processo di *mining* essenzialmente decentralizza il processo di emissione di moneta, e quindi elimina il bisogno di una banca centrale che si occupi di tale emissione.

La difficoltà del problema matematico che i *miner* dovranno affrontare può essere modificato dinamicamente in modo da mantenere una media di un successo ogni 10 minuti, in questo modo anche se il numero di *miner* dovesse mutare, la media dei successi resterebbe sempre invariata.

Il protocollo stabilisce inoltre che ogni 4 anni la percentuale di bitcoin che viene creata si dimezzi, e il numero totale di bitcoin che possono essere creati è fissato a 21 milio-

---

<sup>2</sup>In informatica, di facile utilizzo anche per chi non è esperto.

<sup>3</sup>Architettura di rete informatica in cui i nodi sono tra loro paritetici, potendosi comportare sia da client che da server.

ni, una volta raggiunta questa soglia non verranno più creati nuovi bitcoin. Possiamo quindi prevedere che il numero di bitcoin in circolazione raggiungerà l'apice massimo (21 milioni) nel 2040. Grazie a questo i bitcoin non potranno essere inflazionati poichè non può essere "stampata" nuova moneta.

La moneta Bitcoin è la prima vera applicazione di questa invenzione <sup>4</sup>, rappresenta il culmine di anni di ricerche nell'ambito della crittografia, dei sistemi distribuiti, e molteplici altre discipline, tuttavia "Non c'è nulla a cui lo si possa paragonare", dice il suo stesso ideatore Satoshi Nakamoto <sup>5</sup>

## 1.2 Cenni storici

Bitcoin è una *criptovaluta* <sup>6</sup> e un sistema di pagamento mondiale creato nel 2008 da un anonimo inventore noto con lo pseudonimo di Satoshi Nakamoto. Questa innovativa tecnologia fu presentata al mondo attraverso la pubblicazione del *paper* (documento) [4]. Nakamoto ha così creato un sistema di denaro elettronico completamente decentralizzato [6], e l'innovazione principale che ha reso possibile ciò è stata l'elaborazione dell'algoritmo chiamato *Proof-of-Work* che permette alla rete di arrivare ad un consenso generale sullo stato delle transazioni, grazie a tale algoritmo si è anche risolto il problema del *double spend*, che consiste nel fatto di poter spendere la stessa valuta più di una sola volta.

Il network di bitcoin è iniziato nel 2009, grazie all'implementazione rilasciata da Nakamoto stesso, la quale poi negli anni ha visto più modifiche da parte di altri programmatori che con il loro contributo hanno reso sempre più efficiente il programma

---

<sup>4</sup>Bitcoin è solamente la prima di tante criptovalute, successivamente a Bitcoin furono create ad esempio *Ethereum*, *Ripple*, *Bitcoin Cash*...

<sup>5</sup>Pseudonimo dell'inventore della criptovaluta Bitcoin

<sup>6</sup>Il vocabolo criptovaluta è l'italianizzazione dell'inglese *cryptocurrency* e si riferisce ad una rappresentazione digitale di valore basata sulla crittografia.

di Nakamoto. Nell'agosto 2010, una grave falla per la sicurezza venne trovata nel protocollo, a seguito della quale le transazioni non venivano correttamente verificate prima di essere immesse nella blockchain, il che avrebbe potuto consentire ad un eventuale aggressore di generare una quantità arbitraria di BTC. Tale falla venne sfruttata nel giro di pochi giorni, prima che venisse predisposta una patch correttiva, ma poi il "buco" venne chiuso e la situazione stabilizzata grazie a specifici interventi. Negli anni successivi, il valore dei BTC è cresciuto sensibilmente, passando da semplici frazioni di dollaro a un ammontare di 10.092,20 dollari <sup>7</sup> Questa tecnologia ha aperto le strade a nuovi studi nel campo del calcolo distribuito, dell'economia e dell'econometria.

### 1.2.1 La storia del mining

Il cosiddetto *mining* è il processo con il quale si registrano le transazioni nel libro mastro di Bitcoin, un registro che ogni nodo della rete Bitcoin possiede in cui sono salvate tutte le transazioni effettuate e validate fino a quel momento. Il libro delle passate transazioni è chiamato *blockchain* ed è appunto una catena di blocchi. La *blockchain* serve essenzialmente per avere una conferma all'interno della rete di avvenute transazioni. I nodi che si occupano di minare blocchi (*miners*) devono inserire all'interno del blocco minato la *proof-of-work* per far sì che il blocco venga considerato valido. Questa *proof-of-work* verrà verificata dagli altri nodi della rete non appena riceveranno il nuovo blocco minato. Intuitivamente la *proof-of-work* ci dà la sicurezza che il nodo che ha minato il blocco ha realmente fatto del "lavoro", ha impiegato cioè il suo potere computazionale nella risoluzione di un problema matematico, e per questo viene ricompensato. Proprio come un comune lavoratore riceve lo stipendio per il lavoro svolto. Minare un blocco è molto difficile (vedremo nei capitoli successivi

---

<sup>7</sup>Data di rilevazione del prezzo: domenica 25 agosto 2019. (Fonte: [blockchain.info](https://blockchain.info))

i vari tecnicismi) quindi richiede un potere computazionale elevato. La difficoltà del problema che i *miners* devono risolvere può essere però modificata dinamicamente. Nei primi anni di vita di Bitcoin, il mining era solamente un hobby per gli appassionati di criptovalute. Un semplice computer era l'unico hardware richiesto, ma nell'arco di 10 anni le cose sono cambiate drasticamente.

Nel 2009 i primi minatori utilizzavano CPU multi-core standard per produrre BTC a un ritmo di 50 btc per blocco. Un altro avvenimento fondamentale per la storia del mining accadde nell'ottobre 2010, un utente della community rilasciò il codice per il *mining* bitcoin con GPU. Con l'aumentare delle difficoltà data dalle GPU che hanno aumentato l' *hashrate* <sup>8</sup>, aumentò anche l'esigenza di hardware migliore e dedicato.

Il prezzo della criptovaluta insieme alla difficoltà del *mining* continuavano a crescere, e con essi, i requisiti hardware necessari per guadagnare abbastanza fecero divenire le schede video obsolete per consentire agli utenti medi di fare soldi. Nel 2011 comparirono i primi dispositivi (*FPGA*) facendo aumentare la possibilità di guadagno dei *miners*. Il maggior vantaggio che comportò l'uso di questa tipologia di hardware era il minor consumo di energia. Consumava un terzo di energia rispetto alle *GPU* per svolgere in modo efficace lo stesso compito. Lo sviluppo degli *FPGA* ha presto dato il via all'ingegnerizzazione dei sistemi *ASIC*, che hanno il miglior rendimento tra potenza di calcolo e consumo di energia. Questo passaggio ha segnato il passaggio da un hobby ad una vera propria industria del *mining*.

---

<sup>8</sup>Per hashrate si intende l'unità di misura della potenza di elaborazione della rete Bitcoin. Per fini di sicurezza la rete Bitcoin deve eseguire delle operazioni matematiche intensive. Quando la rete raggiunge un hash rate di 10 Th/s, significa che può realizzare un trilitone di calcoli al secondo. Fonte: <https://bitcoin.org/it/glossario#hash-rate>

## 1.3 Acquistare bitcoin

Siamo ormai abituati ad effettuare pagamenti elettronici attraverso carte di credito, PayPal e conti bancari. Tutti questi meccanismi di pagamento appena citati hanno in comune il fatto di essere reversibili, con questo intendiamo che dopo che un utente ha inviato una certa somma di denaro, teoricamente può riappropriarsi del denaro spedito annullando la transazione. In Bitcoin questo è impossibile, le transazioni sono irreversibili, una volta che una transazione viene registrata all'interno della blockchain non c'è modo di tornare indietro. Come può o allora un nuovo utente della piattaforma Bitcoin entrare in possesso di bitcoin?

Poichè chi vende Bitcoin generalmente si fa pagare attraverso carta di credito (o simili), come fa allora questo venditore di bitcoin ad essere sicuro che l'acquirente non ritorni in possesso del denaro che ha usato per acquistare gli stessi (poichè come abbiamo detto un pagamento con carta di credito è reversibile)? Esistono alcuni metodi sicuri per i nuovi utenti che vogliono acquistare bitcoin:

- Trovare un amico fidato che è già in possesso di bitcoin e comprarli direttamente da lui.
- Usare un servizio come *localbitcoins.com* per trovare un venditore di bitcoin nella tua area (solitamente questi sono venditori fidati).
- Usare un bitcoin ATM della propria città. Questa non è nient'altro che una macchina che accetta contanti e invia direttamente bitcoin sul *wallet* del tuo smartphone.
- Utilizzare un sistema di cambio di valuta che accetti bitcoin associato alla propria banca.

## 1.4 I wallet Bitcoin

Un *wallet bitcoin* è un servizio che consente di interfacciarsi con il sistema Bitcoin. Esistono varie implementazioni di questi *wallet* che si differenziano l'un l'altro in termini di qualità, performance, sicurezza, privacy e affidabilità. Esiste anche un *wallet* conosciuto come "Satoshi Client" o "Bitcoin Core" che deriva dall'implementazione originale scritta da Satoshi Nakamoto. I bitcoin non sono di fatto contenuti all'interno dei *wallet* stessi, ma sono memorizzati in un registro aperto al pubblico, la *blockchain*, sotto degli specifici indirizzi appartenenti ai diversi utenti. Gli indirizzi sono punti di ricezione e invio, e si presentano sotto forma di codici alfanumerici di 33 o 34 caratteri, generalmente iniziati per 1, come ad esempio *1G1vTdCYjqb5gucmhNQH7yTBy9uPHC5Aht*, in modo da non contenere alcun riferimento dell'utente utilizzatore, facendo di Bitcoin un sistema di pagamento pseudonimo. I *wallet* custodiscono solamente le chiavi dell'utente, che gli permettono di spendere i bitcoin associati al preciso indirizzo che deriva dalla chiave pubblica che a sua volta deriva dalla chiave privata in oggetto. Quando si crea un nuovo *wallet*, automaticamente vengono generate cento coppie di chiavi private e pubbliche (key-pool), per cui l'utente può usufruire di più indirizzi diversi, per godere di maggiori livelli di privacy. Esistono diversi tipi di portafogli tra cui scegliere, a seconda dei livelli di praticità, sicurezza e complessità desiderata:

- **Desktop Wallet:** software *wallet* da installare sul proprio computer che permette di memorizzare e custodire le chiavi private all'interno dell'hard disk. L'installazione di questi software, ce ne sono di vario tipo per diversi sistemi operativi, richiede generalmente il download dell'intera *blockchain*. E' stato il primo tipo di *wallet* creato.

- **Mobile Wallet:** Questo è il più comune tipo di *wallet* bitcoin. Può essere usato su smarthpone con sistemi operativi come Apple iOS e Android, questi *wallet* sono spesso un'ottima scelta per i nuovi utenti perchè sono semplici e intuitivi da utilizzare.
- **Web Wallet:** I *web wallet* sono acceduti tramite un web browser e salvano il *wallet* di ogni utente su un server di un proprietario terzo. Tipicamente i *wallet* online sono offerti in via accessoria dagli exchange, piattaforme di compravendita di bitcoin in cambio di valute tradizionali. Considerando gli spiacevoli inconvenienti capitati ad alcuni di questi exchange in passato, probabilmente questo tipo di *wallet* non è il più sicuro della lista, ma è sicuramente il più semplice e veloce da utilizzare, in quanto è possibile accedervi da qualsiasi dispositivo connesso a internet.
- **Hardware Wallet:** Dispositivi creati appositamente per custodire le chiavi private degli indirizzi bitcoin e di altre criptovalute. Sono solitamente dei mini computer aventi un'unica funzione, quella di firmare digitalmente le transazioni con le chiavi private dell'utente. Questi si collegano al computer generalmente via USB, e interagiscono con i software *wallet* in tutta sicurezza anche se il computer risultasse compromesso.
- **Paper Wallet:** Le chiavi che controllano i bitcoin possono essere stampati su materiali come carta, metallo ecc... per avere un "salvataggio" a lungo termine. Ovviamente si tratta di una tecnologia non elevata ma permettono sicurezza. I salvataggi offline vengono spesso chiamati con il termine *cold storage*.



## 1.5 Utilizzo nella vita quotidiana

Facciamo un esempio qui di come potrebbero essere utilizzati i bitcoin nella vita di tutti i giorni. Alice che possiede nel suo *wallet* 0.10 BTC vuole effettuare una transazione per comperare una tazza di caffè nel negozio di Bob a Palo Alto, California. In questo caso il costo della tazza di caffè è di un dollaro e 15 oppure di 15 millibitcoin. Bob possiede nel suo negozio un sistema di pagamento automatico che genera uno speciale QR code contenente una richiesta di pagamento. A differenza di altri QR code che contengono solamente l'indirizzo di destinazione dei bitcoin, questo contiene in oltre, anche l'ammontare del pagamento, e una generica descrizione del tipo "Caffè di Bob". Questo rende più facile generare la transazione poichè Alice vedrà solamente un oggetto come quello mostrato in figura che dovrà scannerizzare con il proprio smartphone per generare una corretta transazione. Una volta generata la transazione



Figura 1.1: QR code per la richiesta di pagamento in BTC

di 0.0150 BTC verso il negozio di Bob, Alice sempre attraverso il suo smartphone autorizza la transazione. Nel giro di pochi secondi Bob vedrà la transazione sul registro, e la transazione sarà quindi completata. Possiamo esaminare la transazione effettuata

una volta che è stata inclusa all'interno della blockchain usando un sito di *explorer* (si veda [1] pp. 16-17).

## Capitolo 2

# Come funziona il sistema Bitcoin?

### 2.1 Crittografia

#### 2.1.1 Introduzione

Molte delle proprietà dei Bitcoin sono possibili grazie alla crittografia, con la quale possono essere generate chiavi digitali, indirizzi Bitcoin e firme digitali. Bitcoin fa uso di una crittografia *asimmetrica o a chiave pubblica*. In questo tipo di crittografia ogni utente possiede una coppia di chiavi, una chiave privata e una chiave pubblica, utilizzate per criptare e decriptare delle informazioni che gli stessi si scambiano in un messaggio. Questo tipo di crittografia risulta molto più efficace rispetto alla crittografia simmetrica, in cui i corrispondenti cifrano e decifrano i dati scambiati utilizzando un'unica chiave segreta (ma in questo caso sorge il problema di come scambiarsi in maniera sicura la chiave stessa). La chiave privata è, appunto, tenuta segreta dal suo possessore, mentre la chiave pubblica può essere resa nota. La chiave pubblica è generata a partire dalla privata attraverso una funzione non iniettiva, ovvero dalla chiave pubblica è impossibile risalire alla chiave privata da cui deriva. Infine chiave privata e chiave pubblica hanno una specifica e diversa funzione: la chiave pubblica

serve cifrare le informazioni da inviare, informazioni che possono essere decifrate soltanto con la relativa chiave privata. Bitcoin si basa su una particolare crittografia asimmetrica chiamata *moltiplicazione su curva ellittica* che offre un meccanismo per la creazione di una coppia di chiavi, una chiave pubblica che è usata per ricevere denaro, e una chiave privata usata invece per firmare le transazioni e spendere fondi. La **chiave privata (k)** è semplicemente un numero casuale, poi da questa chiave usando appunto l' *elliptic curve multiplication* si genera la chiave pubblica, dalla quale infine facendo uso di una *funzione di hash* <sup>1</sup> è possibile generare l'*address* (indirizzo Bitcoin) dello user.

Per creare la chiave privata si prendono 256 bit in modo casuale e si controlla che il numero risultante sia minore di un certo valore  $n$  <sup>2</sup>, in caso negativo si ripete l'esperimento rigenerando un altro numero casuale nello stesso modo.

La **chiave pubblica (K)** è calcolata dalla privata (k) usando la moltiplicazione su curva ellittica:

$$K = k \times G \quad (2.1.1)$$

ove G è la costante di generazione di punti sulla curva.

### 2.1.2 Crittografia su curva ellittica

La crittografia usata da Bitcoin si basa una particolare curva ellittica chiamata *secp256k1*, definita dalla funzione:

$$y^2 \mod p = (x^3 + 7) \mod p \quad (2.1.2)$$

---

<sup>1</sup>Una funzione di hash h, è una funzione che mappa un messaggio arbitrariamente lungo in una stringa di lunghezza prefissata, cercando di far in modo che da questa stringa non si possa risalire al messaggio che l'ha generata.

<sup>2</sup>In Bitcoin  $n=1,1,158 \times 10^{77}$ .

Notiamo che ci troviamo in un caso discreto, la curva sarà quindi descritta da solamente alcuni punti sugli assi cartesiani.

(Per semplificare il ragionamento possiamo pensare al caso reale, *figura 2.1*)

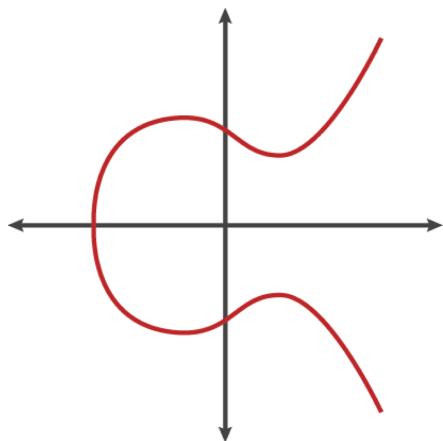


Figura 2.1: Crittografia su curva ellittica: visualizzazione della curva ellittica con  $p=17$ , caso continuo

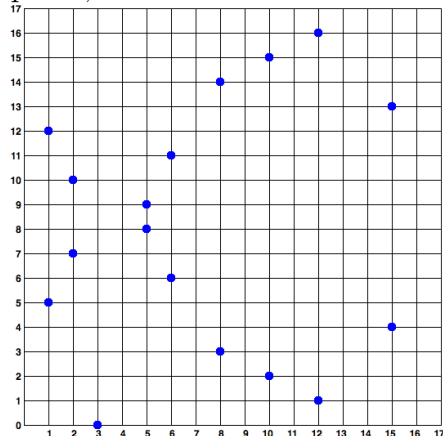


Figura 2.2: Crittografia su curva ellittica: visualizzazione della curva ellittica con  $p=17$ , caso discreto

Una proprietà fondamentale di questa curva è che dati due punti  $P_1$  e  $P_2 \in$  ad essa, allora anche  $P_3 = P_1 + P_2 \in$  alla curva. Possiamo pensare alla moltiplicazione tra due punti della curva nel seguente modo: se voglio moltiplicare un punto per un dato numero  $K$ , basterà sommare il punto a se stesso  $K$  volte, in questo modo risulta molto

semplice ricavare la *public key* dalla *private key* attraverso la 2.1.1

### 2.1.3 Indirizzi Bitcoin

Un indirizzo Bitcoin viene prodotto da una chiave pubblica, e consiste in una stringa che inizia con "1". Un indirizzo può rappresentare il possessore di una coppia di chiavi (*public/private key*) oppure qualcos'altro come uno script di pagamento.

Partendo dalla chiave pubblica ,vengono applicate due funzioni di hash, uno *SHA256* e poi *RIPEMD160* sul risultato, ottenendo un numero a 160 bit (20 byte) che presenterà l' *address* dell'utente:

$$A_{address} = RIPEMD160(SHA256(K))$$

Gli indirizzi Bitcoin sono codificati in *Base58Check* per proteggersi dagli errori umani. Questo formato è utilizzato perchè elimina i caratteri tra loro fraintendibili per l'occhio umano, inoltre sempre per prevenire un errore che può venire commesso da un utente la codifica *Base58Check* crea un meccanismo di *check* (controllo). Il *checksum*<sup>3</sup> sono 4 byte che vengono aggiunti alla fine dell'indirizzo, e derivano dalla funzione *hash* di tutto quello che li precede, così possiamo sempre controllare la correttezza di un indirizzo.

Per convertire dei numeri (dati) in *Base58Check* , per prima cosa creiamo un prefisso chiamato *version-byte* che indica il tipo di dato, ad esempio per un indirizzo Bitcoin è 0 ( 0x00 in hex).

---

<sup>3</sup>In telecomunicazioni e informatica il checksum è una sequenza di bit che viene utilizzata per verificare l'integrità di un dato o di un messaggio che può subire alterazioni durante la trasmissione sul canale di comunicazione.

Poi si crea un checksum con un *double SHA*:

$$checksum = SHA256(SHA256(prefix + data))$$

Dai 32 byte che risulteranno da questo procedimento prendo solo i primi 4.

Il risultato finale è quindi composto da: *prefix, data, checksum*. Il prefisso cambierà a seconda di cosa voglio codificare.

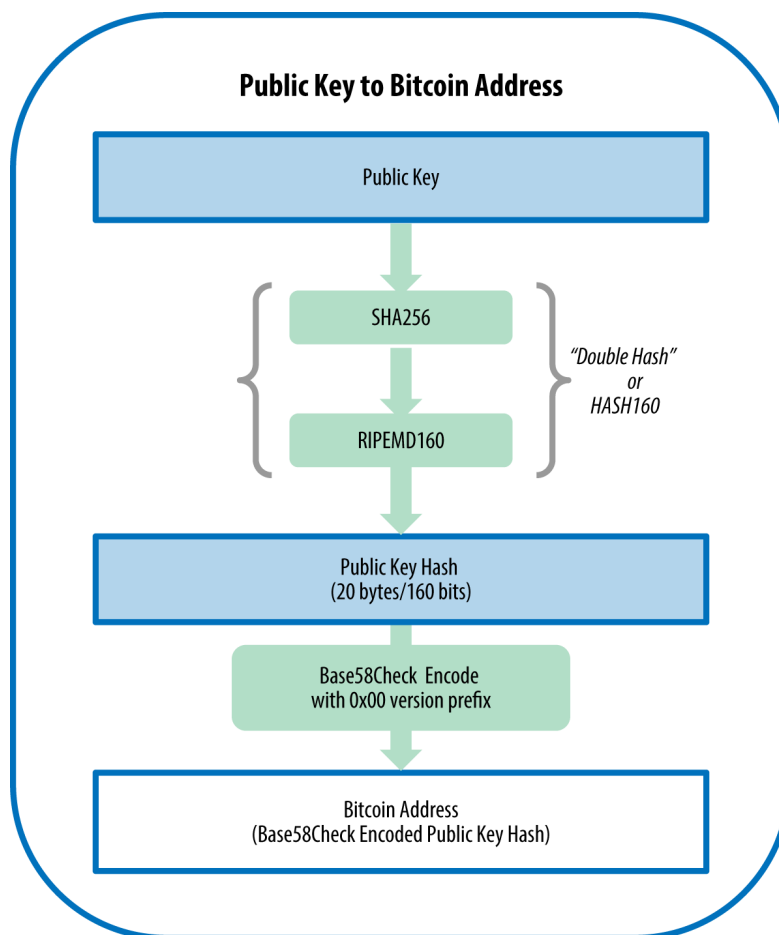


Figura 2.3: Conversione da una chiave pubblica ad un indirizzo Bitcoin

## 2.2 Le transazioni

Le transazioni sono delle strutture dati che codificano il trasferimento di valori tra partecipanti del sistema Bitcoin, forniscono quindi agli utenti la possibilità di spendere *satoshis una sottounità di bitcoin, che può essere paragonata ai centesimi di Euro*. Ogni transazione è composta da diversi elementi, che danno la possibilità di poter svolgere a parimenti transazioni semplici e più complesse, a seconda delle esigenze. Ogni nodo nel network Bitcoin dovrebbe essere in grado di trasmettere le transazioni nel network e farle minare in un blocco. Inoltre, ogni nodo dovrebbe essere in grado di controllare se:

- Il denaro speso nella transazione esiste realmente.
- Il denaro è già stato utilizzato (problema del *double spending*).
- Chi spende il denaro ha effettivamente i diritti per poterlo fare.

L'idea generale delle transazioni nel network Bitcoin è quella di inviare degli Input che vanno a spendere degli Output. Ogni transazione ha quindi almeno un Input ed un Output. Ogni Input spende un Output precedente. Ogni Output è quindi in una specie di sala d'attesa (e prende il nome di Output Non Speso, **UTXO**), finchè un successivo Input non lo "spende". Ogni transazione ha come prefisso il numero della versione, che ha lo scopo di indicare agli altri *peer* ed ai *miner* il corretto insieme di regole da usare per validarla. Questo dà la possibilità agli sviluppatori di poter creare nuove regole senza invalidare le transazioni precedenti.

### 2.2.1 UTXO: output di transazione non spesi

Gli output nelle transazioni sono pezzi indivisibili di bitcoin registrati nella *blockchain*. Un *full node* della rete traccia tutti gli output spendibili, conosciuti come **unspent**



**transaction outputs UTXO.** Ogni transazione rappresenta un cambiamento nell'insieme di UTXO. Quindi quando diciamo che il *wallet* di uno user ha "ricevuto" bitcoin, intendiamo dire che il *wallet* ha trovato un UTXO che può essere speso usando le proprie chiavi. Lo *user's bitcoin balance* è la somma degli UTXO che lo user può spendere.

Come un centesimo non può essere suddiviso, un Bitcoin non può essere suddiviso sotto l'ottavo decimale del *satoshis*. Sebbene un output può avere un valore qualunque, una volta creato questo è indivisibile, gli output sono appunto **discreti** e **indivisibili**. Ad esempio, se si possiede uno UTXO di 20 bitcoin e si vuole pagare 1 bitcoin, la transazione generata deve consumare tutti e 20 i bitcoin producendo 2 output, 1 BTC all'address desiderato, e gli altri 19 BTC al tuo proprio *wallet* (come se fosse il resto). L'applicazione *wallet* di uno user tipicamente seleziona tutti gli UTXO dello stesso per arrivare alla quantità desiderata, per poter così effettuare il pagamento.

C'è però un'eccezione a questa catena di input e output, una speciale transazione chiamata **coinbase transaction**, che è la prima transazione contenuta in ogni blocco della blockchain. Questa transazione è posta lì dal miner "vincente" e crea dei bitcoin spendibili dal miner stesso. La *coinbase transaction* non consuma UTXO, invece ha un input speciale chiamato *coinbase*. Questo è come i Bitcoin vengono creati durante il processo di *mining*.

### 2.2.2 I fees di transazione

La maggior parte delle transazioni includono *fees*, delle ricompense in bitcoin per il lavoro svolto dai *miner*. Analizziamo più in dettaglio come i fees vengono inclusi nelle transazioni. La maggior parte dei *wallet* calcola e include *fees* automaticamente.

Le *transaction fees* (fees di transazione) servono come incentivo per includere (minare)

una transazione nel prossimo blocco, possiamo vederle come una sorta di commissione.

Le *transaction fees* sono calcolate in base alla dimensione della transazione in kilobytes, non in base al valore in BTC di essa, come si potrebbe pensare. I miner danno priorità alle transazioni da includere nel blocco che sono in procinto di minare in base a vari criteri, tra questi c'è proprio la quantità di *fees* che ricaverebbero includendo una data transazione in tale blocco. Una transazione con sufficienti *fees* è perciò più favorevole ad essere inclusa nel prossimo blocco, se invece i *fees* sono troppo pochi la transazione può essere ritardata o addirittura neanche processata. Una transazione con *fees* prossimi allo zero rischia di non essere neanche propagata nel network.

Nel *Bitcoin Core* (implementazione di riferimento di bitcoin) la politica dei *fees* è settata dal ***minerlayfee option***. Attualmente il *minerlayfee* è 0.00001 BTC per kilobyte.

Un algoritmo chiamato *Fees Estimation Algorithm* calcola un ammontare di *fees* appropriato affinché i *fees* offerti rendano la transazione competitiva all'interno del network. Si stima tramite questo algoritmo la quantità necessaria di *fees* che dia alla transazione un'alta probabilità di essere inclusa da qui ad un certo numero di blocchi a venire. I servizi offrono di solito una alta, media, bassa probabilità a seconda di della quantità di *fees* che si è disposti a pagare.

### 2.2.3 Linguaggio di script delle transazioni

L'*engine* di validazione delle transazioni Bitcoin si basa su due tipologie di script, *locking script* e *unlocking script*.

Il *locking script* è posto su un output: specifica la condizione da soddisfare affinché si

possa spendere questo output in futuro. Il *locking script* è chiamato anche *scriptPubKey*, *witness script* o *cryptographic puzzle*.

L' *unlocking script* risolve le condizioni poste su un output attraverso il *locking script*, questa è parte dell'input di ogni transazione. Molto spesso gli *unlocking script* contengono la firma digitale del *wallet* dello user, prodotta dalla sua chiave privata.

L'algoritmo di firma digitale usato nei Bitcoin è l' *Elliptic curve digital signature algorithm ECDSA*. La firma digitale ha tre scopi in Bitcoin:

1. La firma digitale dà prova della proprietà di una chiave privata, cioè del possessore dei fondi.
2. La prova dell' autorizzazione (a spendere) è innegabile.
3. La firma prova che la transazione non può e non deve essere modificata da nessuno dopo che è stata firmata.

Ogni nodo di validazione dei Bitcoin, validerà le transazioni eseguendo il *locking* e l' *unlocking script* insieme. Il software di validazione copierà l' *unlocking script*, prenderà l' *UTXO* referenziato dall'input, e copierà il *locking script* di questo *UTXO*. Ora i due script saranno eseguiti in sequenza. Ci sarà la convalida se l' *unlocking script* soddisferà le condizioni del *locking script*. Tutti gli input sono convalidati indipendentemente. Il linguaggio di script della transazione Bitcoin è chiamato **Script**. Questo contiene molti operatori, ma non contiene loop, solamente controlli condizionali. Il linguaggio quindi **non è Turing completo**. Queste limitazioni permettono di non creare loop infiniti che possono essere usati come *denial of service attack* contro la rete Bitcoin (Ricorda ogni transazione è validata da ogni full node).

## 2.3 La rete Bitcoin

### 2.3.1 Architettura di rete peer-to-peer

Il sistema Bitcoin è strutturato su un'architettura di rete *peer-to-peer* (**P2P**). Con questo termine si indica che i computer che fanno parte della rete sono appunto "alla pari", non ci sono nodi "speciali", e perciò ogni nodo si assume la responsabilità del funzionamento dei vari servizi di rete. Non c'è un server, non ci sono servizi centralizzati, e alcun tipo di gerarchia all'interno della rete. I nodi nella rete *P2P* forniscono e allo stesso tempo consumano i servizi offerti dalla stessa. La caratteristica principale quindi di tale rete è il fatto di essere decentralizzata e aperta a chiunque. All'infuori di Bitcoin le applicazioni che hanno avuto più successo su questa architettura di rete sono *Napster* e *BitTorrent*.

Bitcoin è nato per essere un sistema di moneta digitale su rete *P2P*, e la decentralizzazione del controllo è il cuore dei principi di Bitcoin, e questa può essere raggiunta solamente sfruttando le caratteristiche di una tale rete.

Il termine *Bitcoin Network* si riferisce all'insieme di nodi che stanno eseguendo il protocollo *Bitcoin P2P*.

### 2.3.2 I ruoli dei partecipanti della rete Bitcoin

Sebbene i nodi in una rete *P2P* siano alla pari, questi possono avere compiti differenti all'interno di essa. Un nodo Bitcoin implementa 4 diverse funzionalità fondamentali:

- routing
- blockchain database
- mining

- servizi wallet

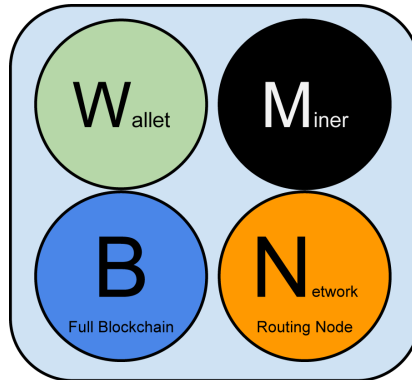


Figura 2.4: Un nodo della rete Bitcoin con tutte e quattro le funzionalità: *wallet*, *miner*, *full blockchain database* e *network routing*

Tutti i nodi includono la funzioni di *routing* per partecipare alla rete. Ogni nodo valida e propaga le transazioni e i blocchi sulla rete, inoltre scopre e mantiene connessioni verso altri *peer*.

Alcuni nodi, chiamati *full nodes*, mantengono anche una copia aggiornata della *blockchain*. I *full nodes* posso quindi verificare in autonomia le transazioni senza aver bisogno di referenze esterne. Alcuni nodi mantengono invece solamente una parte di tutta la *blockchain*, e verificano le transazioni usando un metodo chiamato *simplified payment verification*, *SPV*. Questi nodi sono chiamati nodi *SPV* o nodi *lightweight*. I nodi *miner* competono per creare nuovi blocchi, utilizzando hardware dedicato alla risoluzione dell' algoritmo di *Proof-of-work*. Alcuni nodi *miner* sono anche *full node*, cioè hanno una copia dell' intera *blockchain*, mentre altri sono nodi *lightweight* che partecipano a *pool* di *mining* dove soltanto il *pool server* ha la funzionalità di *full node*.

### 2.3.3 Come fa un nuovo partecipante a collegarsi alla rete Bitcoin?

Quando un nuovo nodo si avvia, questo deve scoprire altri nodi all'interno della rete in modo da poter entrare a far parte dell'ecosistema Bitcoin. Per iniziare questo processo un nodo deve scoprire almeno un altro nodo all'interno della rete e connettersi con esso. La posizione geografica dei nodi è irrilevante, inoltre la topologia della rete Bitcoin non è geograficamente definita. Per questo, si può scegliere a caso qualsiasi nodo Bitcoin della rete.

Per connettersi a un *peer* conosciuto, si stabilisce una connessione TCP, solitamente sulla porta 8333. Dopo aver stabilito la connessione, il nodo inizierà la fase di *handshake* trasmettendo un messaggio denominato ***version*** che contiene le seguenti informazioni:

- *nVersion*: La versione del protocollo Bitcoin P2P che il client "parla"
- *nLocalServices*: Una lista di servizi locali supportati dal nodo
- *nTime*: Il tempo corrente
- *addrYou*: L'indirizzo IP del nodo remoto visualizzato da questo nodo
- *addrMe*: L'indirizzo IP del nodo locale, come è stato scoperto dal nodo locale
- *subver*: Mostra il tipo di software che sta girando su questo nodo
- *BestHeight*: La lunghezza della blockchain

Il messaggio di *version* è sempre il primo messaggio inviato da ogni *peer* agli altri *peer*. Il *peer* che riceve questo messaggio andrà a vedere se il nodo che sta provando a stabilire una connessione con lui è compatibile, facendo un check del campo *nVersion*,

in caso positivo risponderà con un messaggio di *verack*.

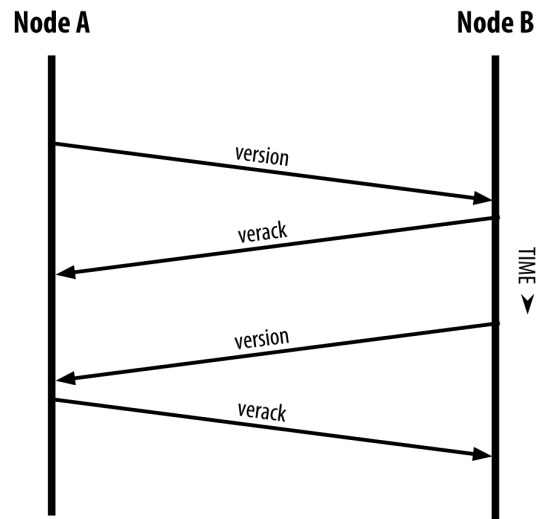


Figura 2.5: Handshake tra due peer della rete Bitcoin

Come fa un nodo a trovare dei *peer*?

Il primo metodo è facendo delle *query DNS* a dei *DNS seeds* che sono server che forniscono indirizzi IP di nodi Bitcoin. Alcuni di questi *DNS seeds* forniscono una lista statica di indirizzi di nodi Bitcoin stabili, altri invece ritornano un sottoinsieme random della lista degli indirizzi dei nodi Bitcoin collezionati da un *crawler* della rete. Il *Bitcoin Core* contiene al suo interno il nome di 5 differenti *DNS seed*. Questi offrono un alto livello di affidabilità per il processo iniziale di *bootstrapping*.

Come alternativa, ad un nodo in fase di *bootstrap* che non conosce nulla circa la rete, può essere fornito l'indirizzo IP di almeno un nodo che si trova nella rete, con il quale questo può stabilire una connessione e dal quale può ricevere maggiori informazioni riguardo la rete stessa. Una volta che una o più connessioni sono state stabilite, il

nuovo nodo invierà un messaggio *addr* ai suoi vicini contenente il proprio indirizzo IP. I vicini inoltreranno il messaggio *addr* a loro volta ai loro vicini assicurandosi così che il nuovo nodo sia ben connesso alla rete <sup>4</sup>. Il nuovo nodo in connessione può in aggiunta anche inviare un messaggio *getaddr* ai suoi vicini, chiedendo loro di ritornare una lista di indirizzi IP di altri vicini. Un nodo deve essere connesso ad alcuni *peer*

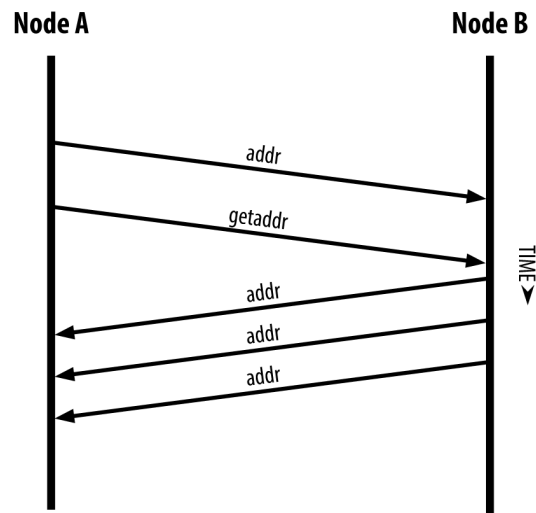


Figura 2.6: Propagazione di indirizzi Bitcoin

differenti in modo da poter stabilire dei path all'interno della rete Bitcoin. Questi path non sono del tutto affidabili, poichè i vicini di ogni nodo possono cambiare dinamicamente. Nella fase di *bootstrap* c'è bisogno solamente di una connessione perchè questa fornirà informazioni su un altro nodo della rete che a sua volta farà in modo di collegarti a tutto il network. Dopo la fase di *bootstrap*, un nodo ricorderà un *peer* con il quale è stato connesso, così ogni volta che verrà ravviato potrà collegarsi con questo senza passare nuovamente per la fase di *bootstrap*.

Se non c'è traffico su una connessione, i nodi periodicamente si scambieranno dei mes-

---

<sup>4</sup>Questo meccanismo di propagazione di messaggi all'interno di una rete viene chiamato *gossip protocol*



saggi per mantenere attiva la connessione, ma se questi smetteranno di comunicare per più di 90 minuti la connessione vierrà chiusa di default. Per questo la rete è dinamica, può ingrandirsi o rimpicciolirsi senza un controllo centrale.

### 2.3.4 Ricostruire la blockchain grazie al comando "Inventory"

La prima cosa che deve fare un *full node* che è appena entrato nella rete, è di provare a ricostruire tutta la *blockchain*. Ogni nuovo nodo conosce solo il primo blocco (*genesis block* <sup>5</sup>), il quale è incorporato all'interno del codice *Bitcoin Core*. Il nuovo nodo deve quindi scaricare centinaia di migliaia di blocchi in modo da sincronizzarsi con tutta la rete.

Un nodo può vedere il messaggio *version* mandatogli dai suoi *peer*, conoscere così la quantità di blocchi che loro possiedono, e compararli con la quantità di blocchi che lui ha nella sua *blockchain*.

Un *peer* che ha una *blockchain* più lunga di un altro, può identificare quali sono i blocchi che a quest'ultimo mancano per poterlo "raggiungere". Quindi identificherà i primi 500 blocchi da condividere e trasmetterà l'*hash* di questi blocchi in un messaggio *inv* (Inventory).

Il nodo che non possiede questi blocchi, cercherà di recuperarli usando una serie di messaggi *getdata* richiedendo così i dati che compongono gli stessi.

Assumiamo per esempio, che un nodo abbia solamente il blocco iniziale. Questo riceverà un messaggio *inv* dai *peer* contenente l'*hash* dei prossimi 500 blocchi della catena. Il nodo inizierà a richiedere i blocchi a tutti i suoi vicini, mantenendo traccia di quanti blocchi sono "in transit" per ogni connessione *peer*, stando attento a non superare il limite di una data costante: *MAX\_BLOCKS\_IN\_TRANSIT\_PER\_PEER*.

---

<sup>5</sup>Il genesis block può essere visualizzato al link : <https://www.blockchain.com/it/btc/block/000000000019d6689c085>

In questo modo, se esso necessita di molti blocchi, chiederà soltanto nuovi blocchi senza richiedere per due volte uno stesso blocco.

## 2.4 La Blockchain

La struttura dati della *blockchain*, è una lista di blocchi ordinata, ognuno dei quali contiene delle transazioni. La *blockchain* può essere registrata in un semplice database. I blocchi sono collegati l'un l'altro, ogni blocco fa riferimento a quello che lo precede. Spesso la *blockchain* viene visualizzata come uno *stack*, con questa visualizzazione viene spontaneo parlare di altezza della *blockchain*, *height*, termine con il quale si intende la distanza tra il primo e l'ultimo blocco dello *stack*.

Ogni blocco è identificato da un *hash*, generato usando l'algoritmo *SHA256* sull' *header* del blocco stesso. Ogni blocco quindi contiene all' interno del proprio *header*, il campo *previous block hash*, che non è altro che un puntatore al blocco genitore (blocco precedente). In altre parole ogni blocco contiene l' *hash* del genitore all' interno del proprio *header*.

Sebbene ogni blocco abbia un solo genitore, potrebbero avere però temporaneamente più figli. Un blocco può avere più di un figlio quando si verifica una *fork* nella *blockchain*, cioè una situazione momentanea in cui due diversi blocchi sono stati scoperti quasi simultaneamente da *miner* differenti. Ad ogni modo le situazioni di *fork* vanno risolte facendo rimanere solamente uno di questi figli all'interno della catena.

Il campo *previous block hash* è all' interno dell' *header* dei blocchi, quindi ovviamente influenza l' *hash* del blocco corrente. L'identità del figlio cambia se cambia quella del genitore. Se in qualunque modo il genitore venisse modificato, verrebbe modificato anche il suo *hash*. Il cambiamento dell' *hash* del genitore necessita un cambiamento nel campo *previous block hash* del figlio. Il cambiamento di questo campo a sua volta

necessita che l' *hash* del figlio cambi, così come l' *hash* del nipote, e così via...

Questo effetto a cascata assicura che quando un blocco ne ha molti altri che lo succedono, questo non potrà essere modificato senza ricalcolare tutti gli *hash* dei blocchi che lo succedono (cosa che richiederebbe una forza computazionale enorme!).

Per via della grande forza computazionale richiesta possiamo essere sicuri che una lunga catena di blocchi rende la storia della *blockchain* immutabile, e questa è una delle caratteristiche principali della sicurezza della stessa.

### 2.4.1 Il contenuto di un blocco della blockchain

Un blocco è semplicemente una struttura dati che aggrega le transazioni al fine di includerle nel *public ledger*, la *blockchain*. Un blocco è composto da un *header*, che contiene i metadati, seguito da una lunga lista di transazioni. L' *header* dei blocchi è di 80 byte, in media ogni transazione è di 400 byte e sempre in media ogni blocco contiene circa 1900 transazioni. L' *header* dei blocchi consiste in tre insiemi di metadati. C'è un riferimento al blocco genitore che crea una connessione tra due blocchi successivi. Il secondo insieme di metadati consiste nei campi *difficulty*, *timestamp*, e *nonce* che sono relative alla competizione dei *miner* che vedremo più avanti. Infine il terzo insieme di metadati consiste nel *merkle tree root*, una struttura dati utilizzata per recapitolare tutte le transazioni all' interno del blocco.

### 2.4.2 Identificazione dei blocchi attraverso header e height

L'identificatore principale di un un blocco è il suo *hash* crittografico, una firma digitale, creata applicando per due volte l'algoritmo *SHA256* sull' *header* del blocco stesso, i 32 byte risultanti sono chiamati *block header hash*.

L' *hash* identifica univocamente un blocco e può essere ricalcolato da ogni nodo sem-

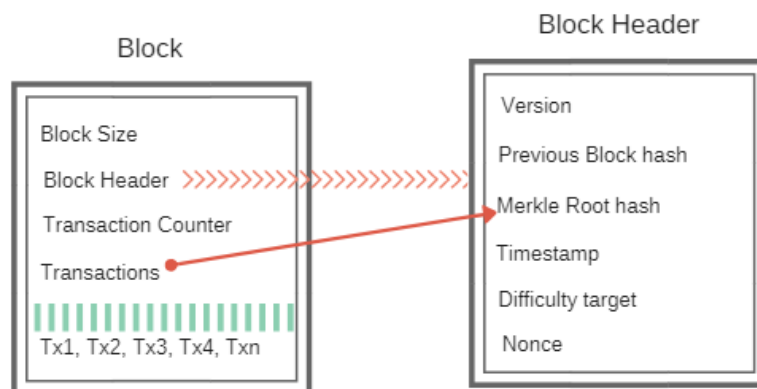


Figura 2.7: Struttura dei blocchi

plicemente riapplicando lo *SHA256* all' *header* del blocco stesso.

Si noti che l' *hash* di un blocco non è incluso all' interno della sua struttura dati , invece questo viene calcolato da ogni nodo non appena il blocco è propagato nel network.

Un secondo modo per identificare un blocco è guardando la sua posizione all' interno della *blockchain*, la *block height*.

Sebbene un blocco abbia sempre una specifica *height*, il viceversa non è sempre vero. Due o più blocchi possono avere la stessa *height*, quando sono in competizione per la stessa posizione all' interno della *blockchain*, questo scenario è chiamato *Blockchain Fork*. Neanche la *block height* fa parte della struttura dati dei blocchi.

**Genesis Block** Il primo blocco nella blockchain è chiamato *genesis block*, ed è stato creato nel 2009. Ogni nodo che entra a far parte del Bitcoin network ha una *blockchain* che al suo interno contiene almeno questo blocco, poichè si trova all'interno dell'implementazione *Bitcoin Core*. In questo modo ogni nodo ha un *root* fidato dal quale può iniziare a costruire la propria catena di blocchi.

### 2.4.3 Collegare i Blocchi nella Blockchain

I Bitcoin *full node* mantengono una copia locale della *blockchain*, che viene costantemente aggiornata non appena arrivano nuovi blocchi. All'arrivo di un nuovo blocco, il *full node* lo validerà e creerà un link per collegarlo alla *blockchain* già esistente, per stabilire questo link il nodo andrà a vedere il campo *previous block hash*, così da poter stabilire se il padre di questo nuovo blocco corrisponde all'ultimo blocco che lui ha in cima al suo *stack* di blocchi.

### 2.4.4 Merkle Trees

Ogni blocco della *blockchain* contiene un riepilogo di tutte le transazioni al suo interno tramite il *merkle tree*. Il *merkle tree*, conosciuto anche come *binary hash tree*, è una struttura dati utilizzata per verificare l'integrità di un grande insieme di dati. I *merkle trees* sono quindi alberi binari che contengono *hash* crittografici.

Sono usati in Bitcoin per tenere traccia di tutte le transazioni in un blocco, producendo una firma digitale di un intero insieme di transazioni, e fornendo un meccanismo efficiente per verificare se una transazione è effettivamente inclusa in un blocco.

Il *merkle tree* è costruito facendo ricorsivamente *hash* di coppie di nodi fino a che non rimane solamente un nodo, chiamato *root* o *merkle root*. L'algoritmo di *hash* crittografico usato è lo *SHA256* applicato due volte.

Usando questa struttura, dati  $N$  elementi all'interno del *merkle tree*, possiamo verificare se uno specifico elemento cui siamo interessati è incluso nel *tree* in al più  $2 \times \log_2 n$  calcoli.

Il *merkle tree* è costruito con un approccio *bottom up*. Nel' esempio in figura 2.8, si parte da quattro transazioni (A,B,C,D) che formano le foglie dell'albero. Coppie consecutive di nodi sono riassunte nel nodo padre, concatenando i loro due *hash* insie-

me. Il processo continua finchè non rimane solo un nodo in cima all' albero, il *merkle root*. Perciò i 32 byte registrati nell' *header* del blocco riassume tutte le transazioni che ne fanno parte. Poichè il *merkle tree* è un albero binario, questo necessita di un

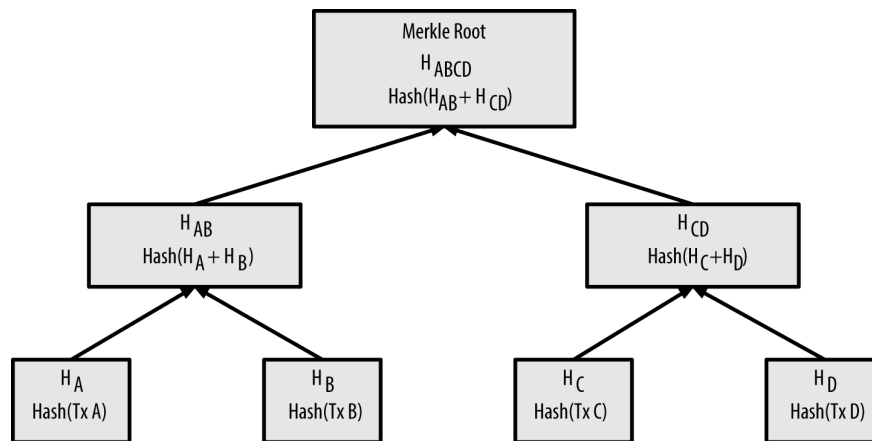


Figura 2.8: calcolo dei nodi in un merkle root

numero pari di foglie. Se ci sono un numero dispari di transazioni, l' *hash* dell'ultima transazione viene duplicato per creare un numero pari di foglie, l'albero che così viene a crearsi è conosciuto con il nome di *balanced tree*.

In Bitcoin è comune avere migliaia di transazioni incluse in un blocco, quindi possiamo immaginare come le dimensioni dell'albero binario siano molto più grandi dell'esempio visto precedentemente.

Per provare che una transazione è davvero inclusa in un blocco, un nodo deve produrre solamente  $\log_2 n$  funzioni di *hash*, costruendo così un *authentication path* o *markle path* che connetta quella specifica transazione al root dell'albero.

Nella figura seguente producendo un *merkle path* vediamo come si può provare che la transazione *K* è inclusa nel blocco. Il path consiste in quattro funzioni di *hash* che sono:  $H_L, H_{IJ}, H_{MNOP}, H_{ABCDEFGH}$ . I nodi *SPV* che non contengono una copia dell'

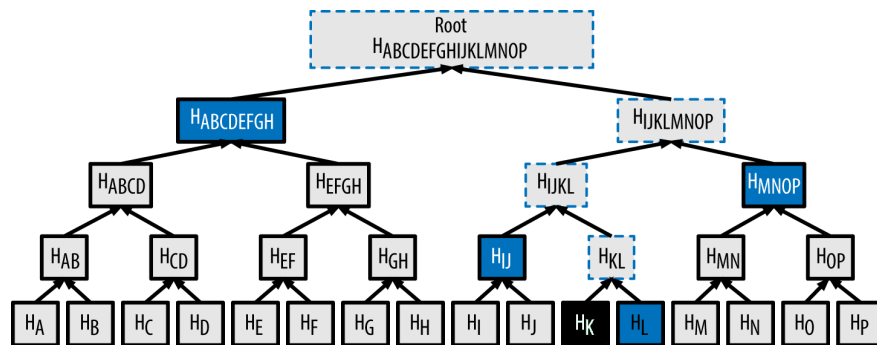


Figura 2.9: Merkle path usato per provare l'inclusione dell' elemento K

intera *blockchain*, usano i *merkle path* per verificare le transazioni, senza dover fare il download di tutto il blocco.

## 2.5 Mining per il consenso della rete

### 2.5.1 Introduzione

L'obiettivo principale del *mining* non è la ricompensa o la generazione di nuovi *coins*, ma il *mining* è il meccanismo con il quale le transazioni vengono validate o eliminate. Il *mining* è ciò che rende Bitcoin un sistema sicuro e decentralizzato che è la base per una moneta digitale basata su rete *P2P*.

Il *mining* dà sicurezza al sistema Bitcoin e permette di creare moneta **senza un' autorità centrale**. I *miner* validano le nuove transazioni e le registrano nel *public ledger* (libro mastro). Un nuovo blocco viene "minato" in media ogni 10 minuti. Le transazioni che diventano parte di un blocco e che sono quindi aggiunte alla *block-chain* sono considerate confermate, questo permette al nuovo proprietario dei bitcoin di poter spendere i fondi di quella transazione.

I *miner* ricevono due tipi di ricompense: ogni volta che viene minato un blocco vengono creati nuovi *coins* che vengono assegnati al *miner*, e inoltre questi ricevono i *fees*

da tutte le transazioni che fanno parte del blocco. Per guadagnare queste ricompense i *miner* devono risolvere un complicato problema matematico basato su un algoritmo di *hash* crittografico. La soluzione a tale problema, chiamato *Proof of Work* è inclusa nel nuovo blocco minato, ed è prova del lavoro computazionale compiuto dal *miner*. Nuovi bitcoin sono quindi creati grazie al processo di *mining*, in modo del tutto analogo a quando una banca stampa nuove banconote. La quantità di bitcoin che vengono creati all'aggiunta di un nuovo blocco nella *blockchain* diminuisce ogni 4 anni (ogni 210.000 blocchi). Facendo qualche conto si può notare che la ricompensa di bitcoin sarà diminuita esponenzialmente entro il 2140, data in cui non verranno più generati nuovi bitcoin oltre ai 20.999999998 milioni che saranno già in circolazione.

Abbiamo detto che i *miner* ricevono anche come ricompensa i *fees* da ogni transazione, oggi però la ricompensa in *fees* ricopre solamente il 5% del guadagno di un *miner*. Ad ogni modo poichè generazione di nuovi coins si abbasserà esponenzialmente, mentre il numero di transazioni per blocco aumenterà, arriverà un momento in cui la ricompensa in *fees* sarà la parte dominante del guadagno dei *miner*.

### 2.5.2 Consenso Decentralizzato

La *blockchain*, come visto nel capitolo precedente, è il libro mastro di tutte le transazioni, a cui ogni nodo della rete Bitcoin si affida.

Ma come possono tutti fidarsi di una singola *verità*, senza potersi e doversi fidare di nessun' altro all'interno della rete? I metodi di pagamento tradizionali dipendevano sempre da un' figura di autorità centrale, Bitcoin non ha questa figura, ma in qualche modo ogni *full node* ha una copia completa del libro mastro alla quale può affidarsi. Ogni nodo in qualche modo, attraverso lo scambio di informazioni sul network riesce ad arrivare alla stessa copia del libro mastro.



L'invenzione forse più importante introdotta da Satoshi Nakamoto è il meccanismo decentralizzato chiamato *emergent consensus*. Il consenso è una caratteristica emergente dell'interazione asincrona di migliaia di nodi indipendenti che seguono delle semplici regole.

Il consenso decentralizzato dei Bitcoin emerge dall'interazione di quattro processi che si verificano indipendentemente sui nodi della rete:

- Verifica indipendente di ogni transazione da parte di ogni *full node*:  
prima di propagare una transazione all'interno del network i *full node* verificano che la transazione sia valida, in caso negativo questa sarà già scartata dal primo nodo.
- Aggregazione indipendente di quelle transazioni nei nuovi blocchi fatta dai nodi *miner*, attraverso la computazione della *Proof of Work*.
- Verifica indipendente dei nuovi blocchi, da ogni nodo e assemblamento in un catena.
- Selezione indipendente da ogni nodo, della catena con più lunga.

### 2.5.3 Aggregare Transazioni nei Blocchi

Dopo aver validato le transazioni, un nodo Bitcoin le aggiunge nella *memory/transaction pool*, dove queste attendono di essere incluse in un blocco.

Descriviamo con un esempio ora il procedimento che compie un nodo *miner* all'interno della rete.

Il nodo *miner* di Gianni mantiene una copia locale della *blockchain*. Allo stesso tempo Alice sta comprando una tazza di caffè, in questo momento la *blockchain* di Gianni

contiene 277,324 blocchi. Il nodo di Gianni rimane in ascolto per ricevere sempre nuove transazioni, prova a minare un nuovo blocco e inoltre controlla se qualche blocco è stato aggiunto da qualche altro *miner* alla *blockchain*.

Mentre il nodo di Gianni sta minando sfortunatamente viene annunciato il blocco 277,315 , così inizia subito la competizione per il blocco 277,316.

Durante i 10 minuti precedenti, mentre Gianni cercava di minare il blocco 277,315, stava allo stesso tempo collezionando transazioni da includere nel blocco. Adesso dovrà controllare che tra le transazioni che ha collezionato non ce ne sia qualcuna inclusa nel blocco 277,315, in tal caso dovrà eliminarla dalla *transaction pool*.

Gianni costruisce immediatamente un blocco vuoto, candidato per la posizione 277,316, che viene appunto chiamato *candidate block*, poichè il *miner* ancora non ha trovato una soluzione all'algoritmo di *Proof-of-Work*.

#### 2.5.4 La Transazione Coinbase

La prima transazione in ogni blocco è speciale, viene chiamata *coinbase transaction*. Questa transazione viene costruita dal nodo i Gianni come ricompensa degli sforzi fatti.

A differenza della transazioni comuni, la *coinbase* non consuma *UTXO* per i suoi input, invece ha solamente un input chiamato *coinbase*.

#### 2.5.5 Minare il Blocco

Adesso che il nodo di Gianni è stato costruito, è tempo di minare il blocco, trovando una soluzione al *Proof of Work* che renda il blocco valido.

In termini semplici, minare è il processo in cui vengono eseguite tante funzioni di *hash* ripetutamente, cambiando un solo parametro, fin tanto che il risultato dell'*hash* non

combaci con uno specifico target. Il risultato della funzione di *hash* non può essere predeterminato, questa caratteristica fa in modo che l'unica maniera di produrre degli *hash* che combacino con uno specifico target sia solamente provando e riprovando in modo del tutto casuale, modificando l'input.

### 2.5.6 Algoritmo Proof-of-Work

Un algoritmo di *hash* prende in input un dato di lunghezza arbitrariamente lunga e produce un risultato di lunghezza fissa, la firma digitale dell'input. Per ogni specifico input, il risultato della funzione di *hash* sarà sempre lo stesso. La caratteristica principale di un algoritmo di *hash* crittografico è che è computazionalmente impossibile trovare due differenti input che producano la stessa firma (situazione nota come *collisione*).

Con l'algoritmo di *hash* usato in Bitcoin, *SHA256*, l'output ha una lunghezza fissa di 256 bit. Vediamo degli esempi in cui ad una particolare stringa viene aggiunto un numero (*nonce*) che cambia continuamente e alla quale applichiamo l'*hash* crittografico:

Io sono Marcello Politi0 := a80a81401765c8eddee25df36728d732...

Io sono Marcello Politi1 := f7bc9a6304a4647bb41241a677b5345f...

Io sono Marcello Politi2 := ea758a8134b115298a1583ffb80ae629...

Io sono Marcello Politi3 := bfa9779618ff072c903d773de30c99bd...

Io sono Marcello Politi4 := bce8564de9a83c18c31944a66bde992f...

Io sono Marcello Politi5 := eb362c3cf3479be0a97a20163589038e...

Io sono Marcello Politi6 := 4a2fd48e3be420d0d28e202360cfbaba...

Io sono Marcello Politi7 := 790b5a1349a5f2b909bf74d0d166b17a...

Io sono Marcello Politi8 := 702c45e5b15aa54b625d68dd947f1597...



fare per vincere, e questo ci dà una prova che abbiamo fatto diversi tentativi prima di vincere, lo stesso vale quindi per i *miner*, il fatto che trovino un *hash* che rispetti il target è un *proof* (prova) *of work* (del lavoro fatto).

Nell'esempio mostrato precedentemente il *nonce* vincente era il numero 13, e questo risultato può essere confermato da qualunque nodo indipendentemente. Tutti possono aggiungere il suffisso "13" alla frase "Io sono Marcello Politi" ed eseguirne l'*hash* per vedere che quello che ne viene fuori sia un numero minore del target. Conoscendo il target ognuno può fare una stima della difficoltà di risolvere la *Proof of Work* e conoscere quindi quanto lavoro ci vuole per trovare il *nonce*.

Abbiamo già visto come un *miner* costruisce un *candidate block* pieno di transazioni. Successivamente il *miner* calcola l'*hash* dell'*header* del blocco e verifica che sia più piccolo del target, se non lo è il *miner* modificherà il *nonce* e proverà nuovamente.

## Capitolo 3

# Analisi delle connessioni di un singolo nodo

### 3.1 Motivazioni

Abbiamo visto nei capitoli precedenti come può essere utilizzata la tecnologia introdotta da Satoshi Nakamoto, e analizzato anche al livello più tecnico le parti fondamentali che costituiscono l'ecosistema Bitcoin.

In questo, e nel successivo capitolo voglio focalizzarmi sulla topologia della rete *peer-to-peer* sulla quale si poggia Bitcoin. Si è già accennato il fatto che la topologia di tale rete è deliberatamente tenuta nascosta, ma quali sono le motivazione principali del perchè debba essere segreta? La conoscenza della topologia della rete non deve essere per forza considerata una minaccia o una vulnerabilità, sebbene potrebbe comunque essere utilizzata in modo inappropriato da parte di hacker che potrebbero creare attacchi che hanno come bersaglio la rete stessa o rendere note le identità degli *user*, sappiamo bene che l'anonaminità è una delle caratteristiche più apprezzate della tecnologia Bitcoin. Un esempio di attacchi di questa tipo è l' *eclipse attack*, basato su un network decentralizzato in cui chi attacca cerca di isolare uno specifico

*user* anzichè attaccare l'intera rete, e ovviamente conoscendo la topologia della stessa risulterebbe molto più facile identificare quali siano i nodi con meno connessioni e quindi più isolati. Uno studio sulla topologia della rete potrebbe comunque rivelare delle caratteristiche interessanti riguardanti la stessa, si potrebbe vedere in che misura la rete sia veramente decentralizzata identificando *supernodi*, *nodi ponte*, *potenziali punti di rottura*, ecc.. Lo studio della topologia della rete di Bitcoin è stato argomento di studio da parte di vari gruppi di ricerca, ognuno dei quali ha affrontato il problema di inferire le connessioni del network in maniera e con tecniche differenti. La tecnica mostrata nel paper [2] si basa fortemente sul monitoraggio delle *pool* delle transazioni orfane e su come queste vengano propagate all'interno del network Bitcoin. Un approccio differente è invece quello descritto in [3] dove le connessioni tra i nodi vengono identificate attraverso l'analisi dei *timestamp* i quali vengono aggiornati continuamente quando i nodi della rete si scambiano messaggi. Infine vorrei citare il [5] nel quale viene mostrato come inferire la topologia della rete Bitcoin facendo osservazioni e comparazioni tra le informazioni riguardo al *delay* di propagazione della rete Bitcoin e il *delay* di propagazione di un modello matematico scelto come confronto.

In questa tesi avremo un obiettivo meno ambizioso, andremo ad analizzare come cambia la rete dal punto di vista di un singolo nodo. Per questo esperimento si è installato un *full node* che monitoreremo periodicamente per ricavare dei dati circa lo stato delle sue connessioni con altri *peers*, si andrà poi a fare un'analisi dei dati così raccolti, attraverso un programma scritto in python che evidenzierà varie caratteristiche derivate dall'elaborazione di questi dati come ad esempio la percentuale dei nodi che rimangono connessi dall'inizio alla fine della monitorizzazione, la probabilità in media che una connessione "crolli", metterà in risalto inoltre quei nodi che sono rimasti connessi per più tempo poichè potrebbero essere nodi stabili della rete, sarà mostrato con un

grafico a barre per ogni *peer* (identificato dall'indirizzo IP) la percentuale di tempo che è rimasto in connessione con il nostro *full node* e vedremo se l'analisi di differenti monitorizzazioni produrranno risultati simili.

## 3.2 Descrizione dell'applicativo

L'applicativo consiste nel richiamo periodico del comando *bitcoin-cli getpeerinfo* da parte di un *full node* in nostro possesso. Questo comando ritorna alcune informazioni riguardanti le connessioni del nodo verso altri nodi della rete, possiamo vedere un esempio nella figura 3.1.

```
getpeerinfo

Returns data about each connected network node as a json array of objects.

Result:
[
  {
    "id": n,                (numeric) Peer index
    "addr": "host:port",    (string) The IP address and port of the peer
    "addrbind": "ip:port",  (string) Bind address of the connection to the
peer
    "addrlocal": "ip:port", (string) Local address as reported by the peer
    "services": "xxxxxxxxxxxx", (string) The services offered
    "relaytxes": true|false, (boolean) Whether peer has asked us to relay
transactions to it
    "lastsend": ttt,        (numeric) The time in seconds since epoch
(Jan 1 1970 GMT) of the last send
    "lastrecv": ttt,        (numeric) The time in seconds since epoch
(Jan 1 1970 GMT) of the last receive
    "bytessent": n,         (numeric) The total bytes sent
    "bytesrecv": n,         (numeric) The total bytes received
    "conntime": ttt,        (numeric) The connection time in seconds
since epoch (Jan 1 1970 GMT)
    "timeoffset": ttt,      (numeric) The time offset in seconds
    "pingtime": n,          (numeric) ping time (if available)
    "minping": n,           (numeric) minimum observed ping time (if any
at all)
    "pingwait": n,          (numeric) ping wait (if non-zero)
    "version": v,           (numeric) The peer version, such as 7001
    "subver": "/Satoshi:0.8.5/", (string) The string version
    "inbound": true|false,   (boolean) Inbound (true) or Outbound (false)
    "addnode": true|false,   (boolean) Whether connection was due to
addnode/-connect or if it was an automatic/inbound connection
    "startingheight": n,    (numeric) The starting height (block) of the
peer
    "banscore": n,          (numeric) The ban score
    "synced_headers": n,    (numeric) The last header we have in common
with this peer
    "synced_blocks": n,     (numeric) The last block we have in common
with this peer
  ]

```

Figura 3.1: <https://bitcoincore.org/en/doc/0.16.0/rpc/network/getpeerinfo/>



Lanciando più volte il comando appena descritto potremmo ricavare dati utili da elaborare con lo scopo di capire il comportamento del nostro nodo e i suoi vicini all'interno del network.

L'applicativo consiste in due programmi scritti in python, *raccolta\_dati\_csv.py* e *analisi\_dati\_csv.py*. Il primo serve appunto per raccogliere informazioni circa il nostro *full node*, quando viene richiamato questo richiede un parametro, *param1*, che indica il nome del file che il programma andrà a creare per salvare i dati raccolti (si veda figura 3.2). L'intervallo di tempo tra due "lanci" consecutivi è di 5 minuti, il programma

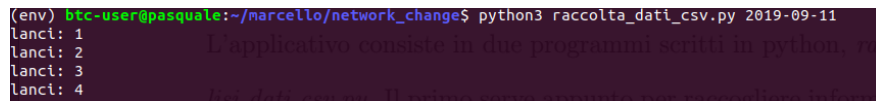


Figura 3.2: Esempio lancio del programma *raccolta\_dati.py*

raccoglierà dati ad oltranza finché non verrà fermato. Ogni ora comunque sarà creato un file di backup dei dati fino a quel momento ottenuti. I file creati saranno dei *CSV*, in cui in ogni riga, il primo valore indicherà il timestamp, mentre gli altri, saranno la lista di indirizzi IP che erano connessi al nostro nodo in quel preciso momento. Come già accennato i risultati saranno salvati in una cartella identificata dal nome *param1*, all'interno della quale troveremo il file dei dati effettivi denominato *param1.txt*, inoltre sarà creata anche un'altra cartella con all'interno un medesimo file di backup.

Una volta recuperati questi dati (si veda figura 3.3) dalla monitorizzazione del *full node*, il programma *analisi\_dati\_csv.py* li andrà ad elaborare per estrapolarne alcune statistiche. Questo secondo programma dovrà essere richiamato con un parametro che indichi il *path* del file di cui siamo interessati. (si veda figura 3.4).

Alla fine dell'elaborazione del programma saranno create tre cartelle: *risultati\_param1*, *geolocate\_param1*, e *intervalli\_param1*. Nella prima cartella troveremo un file con al-

```

1568212282.2876675,["47.244.129.27","47.52.77.214","128.199.89.12","34.234.104.4","83.164.131.243","188.166.69.73","108.175.2.240","15
1568212582.3854675,["47.244.129.27","47.52.77.214","128.199.89.12","34.234.104.4","83.164.131.243","188.166.69.73","108.175.2.240","15
1568212882.4929233,["47.244.129.27","47.52.77.214","128.199.89.12","34.234.104.4","83.164.131.243","188.166.69.73","108.175.2.240","15
1568213182.593538,["47.244.129.27","47.52.77.214","128.199.89.12","34.234.104.4","83.164.131.243","188.166.69.73","108.175.2.240","15
1568213482.6972077,["47.244.129.27","47.52.77.214","128.199.89.12","34.234.104.4","83.164.131.243","188.166.69.73","108.175.2.240","15
1568213782.8104665,["47.244.129.27","47.52.77.214","128.199.89.12","34.234.104.4","83.164.131.243","188.166.69.73","108.175.2.240","15
1568214082.9052317,["47.244.129.27","47.52.77.214","128.199.89.12","34.234.104.4","83.164.131.243","188.166.69.73","108.175.2.240","15
1568214383.0222101,["47.244.129.27","47.52.77.214","128.199.89.12","34.234.104.4","83.164.131.243","188.166.69.73","108.175.2.240","15
1568214683.0415962,["47.244.129.27","47.52.77.214","128.199.89.12","34.234.104.4","83.164.131.243","188.166.69.73","108.175.2.240","15
1568214983.0921026,["47.244.129.27","47.52.77.214","128.199.89.12","34.234.104.4","83.164.131.243","188.166.69.73","108.175.2.240","15
1568215283.2050607,["47.244.129.27","47.52.77.214","128.199.89.12","34.234.104.4","83.164.131.243","188.166.69.73","108.175.2.240","15
1568215583.2303383,["47.244.129.27","47.52.77.214","128.199.89.12","34.234.104.4","83.164.131.243","188.166.69.73","108.175.2.240","15
1568215883.2490777,["47.244.129.27","47.52.77.214","128.199.89.12","34.234.104.4","83.164.131.243","188.166.69.73","108.175.2.240","15
1568216183.2710886,["47.244.129.27","47.52.77.214","128.199.89.12","34.234.104.4","83.164.131.243","188.166.69.73","108.175.2.240","15
1568216483.3535774,["47.244.129.27","47.52.77.214","128.199.89.12","34.234.104.4","83.164.131.243","188.166.69.73","108.175.2.240","15
1568216783.4657855,["47.244.129.27","47.52.77.214","128.199.89.12","34.234.104.4","83.164.131.243","188.166.69.73","108.175.2.240","15
1568217083.5817042,["47.244.129.27","47.52.77.214","128.199.89.12","34.234.104.4","83.164.131.243","188.166.69.73","108.175.2.240","15
1568217383.5970106,["47.244.129.27","47.52.77.214","128.199.89.12","34.234.104.4","83.164.131.243","188.166.69.73","108.175.2.240","15
1568217683.7098148,["47.244.129.27","47.52.77.214","128.199.89.12","34.234.104.4","83.164.131.243","188.166.69.73","108.175.2.240","15
1568217983.7920513,["47.244.129.27","47.52.77.214","128.199.89.12","34.234.104.4","83.164.131.243","188.166.69.73","108.175.2.240","15

```

Figura 3.3: Esempio dati raccolti.py

```

(env) btc-user@pasquale:~/marcello/network_change$ python3 analisi_dati.py 2019/2019-09-08\ 17:04:41.txt

```

Figura 3.4: Esempio lancio del programma analisi\_dati.py

cune statistiche. Alle prime righe saranno identificati il giorno e l'orario in cui la monitorizzazione ha avuto inizio e quello in cui è terminata (si veda figura 3.5). Dopo

```

ho monitorato dal giorno : 2019-09-11 16:31:22
al giorno : 2019-09-19 09:35:23

```

Figura 3.5: Date di inizio e fine monitorizzazioni

di chè ci sarà una lista formata da tre valori: lancio - percentuale - numero di peer connessi (si veda figura 3.6). Come intuibile il primo di questi valori indica il numero di lancio considerato, il secondo parametro invece indica quanti peer in percentuale in quel dato lancio sono rimasti invariati rispetto alla prima monitorizzazione, mentre l'ultimo indica quanti peer sono connessi in quel momento. In figura 3.7 vediamo una lista che indica i lanci in cui si è notato il massimo numero di peer connessi (in questo caso 24) e una lista che indica invece i lanci in cui il numero di peer connessi è stato minimo (in questo caso 22). Una tabella riassuntiva con la media di peer connessi ad

lancio	percentuale	numero-peer-connessi
1	1.000000 %	24
2	1.000000 %	24
3	1.000000 %	24
4	1.000000 %	24
5	0.958333 %	24
6	0.958333 %	23
7	0.958333 %	23
8	0.958333 %	24
9	0.958333 %	24
10	0.958333 %	24
11	0.958333 %	24
12	0.958333 %	23
13	0.958333 %	23
14	0.958333 %	24
15	0.958333 %	23
16	0.958333 %	23
17	0.958333 %	24
18	0.958333 %	24
19	0.958333 %	24
20	0.958333 %	24
21	0.958333 %	24
22	0.958333 %	24
23	0.958333 %	24
24	0.958333 %	24
25	0.958333 %	24

Figura 3.6: Percentuale dei *peer* invariati per ogni lancio

ogni lancio, la relativa varianza, il numero minimo e massimo di peer connessi durante la monitorizzazione, e in quanti lanci questi minimi e massimi si sono verificati. Infine il numero totale di indirizzi IP (*peer*) dei quali il nostro *full-node* è venuto a conoscenza. Le due liste successive che vediamo sempre in figura contengono invece gli indirizzi dei *peer* che si sono presentati con minor frequenza, e quelli che chiamiamo "stabili", cioè che sono sempre rimasti connessi. Il programma mostrerà inoltre un rudimentale grafico a barre (figura 3.8) dove per ogni indirizzo IP ci sarà la percentuale di tempo per cui è rimasto connesso. Alla fine del file (figura 3.9) *connessioni peer* conterra per ogni *peer* il numero di volte che è comparso durante la monitorizzazione del comando *bitcoin-cli getpeerinfo*, e una tabella riassuntiva. In tale tabella abbiamo la media delle probabilità con cui "crolla" una connessione e la sua varianza, la percentuale minima di connessione per cui un *peer* è stato connesso e il numero di *peer* che hanno

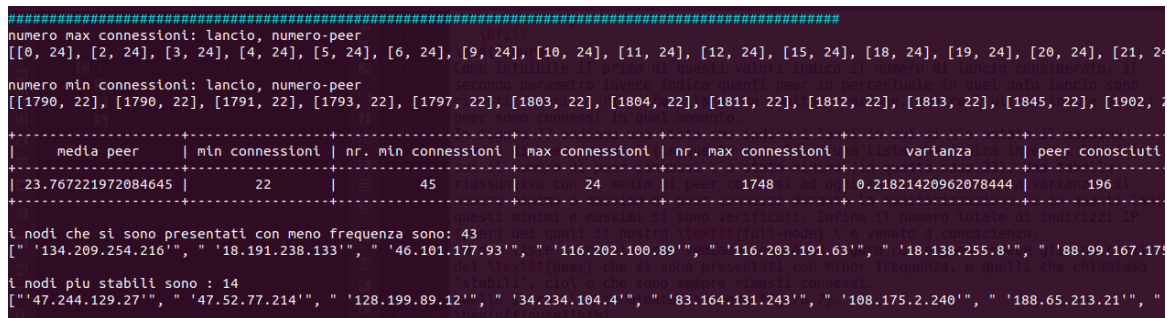


Figura 3.7: Statistiche sulle connessioni dei *peer*

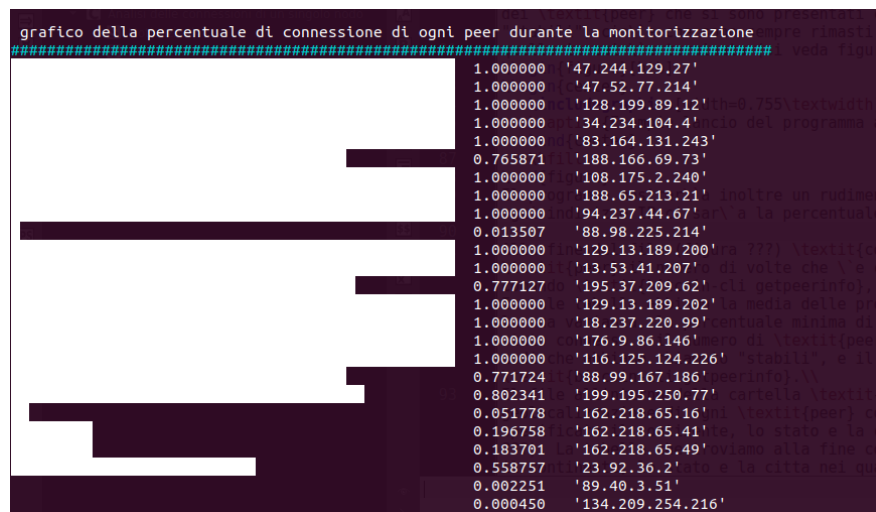


Figura 3.8: Grafico a barre

questa percentuale, il numero di nodi che abbiamo chiamato "stabili", e il numero di lanci totali del comando *bitcoin-cli getpeerinfo*.

Il file all'interno della cartella *geolocate\_param1* contiene dati riguardo la geolocalizzazione di ogni *peer* conosciuto. Per ogni indirizzo IP è specificato il continente, lo stato e la città in cui esso si trova (vedi figura 3.10). La tabella che troviamo alla fine contiene alcuni campi importanti tra i quali: il continente, lo stato e la città nei quali abbiamo incontrato il maggior numero di *peer* con il relativo numero, e analogamente



Figura 3.9: Statistiche sulla percentuale di connessione di ogni peer

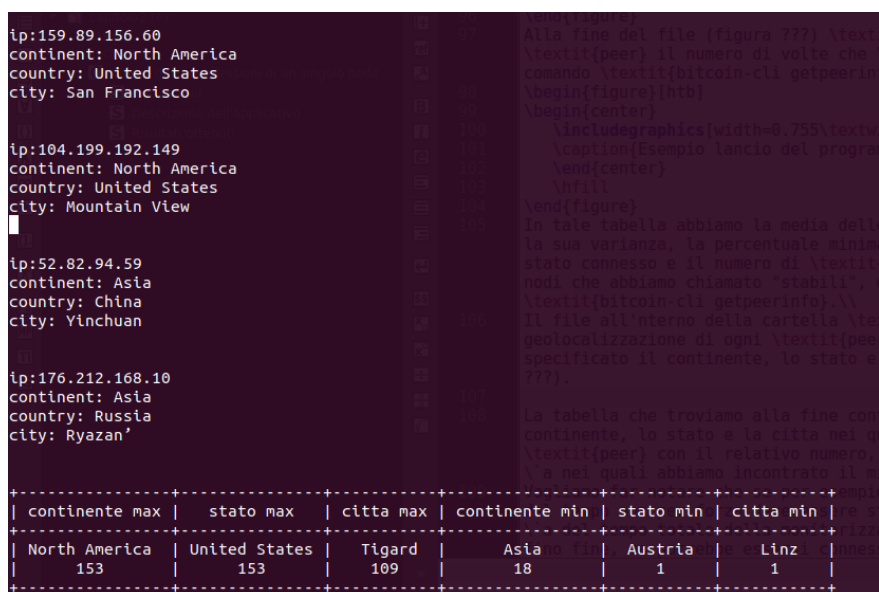


Figura 3.10: Geolocalizzazione dei *peer*

il continente, lo stato e la città nei quali abbiamo incontrato il minor numero di *peer*.  
Vogliamo far notare che se per esempio un *peer* è stato connesso per il 50% del tempo non per forza deve essere stato connesso dall'inizio fino alla fine prima metà del tempo totale della monitorizzazione, oppure dall'inizio della seconda metà fino fine, ma potrebbe essersi connesso e riconnesso più volte fino a totalizzare la percentuale prima indicata. Per questo scopo nell'ultima cartella, *intervalli\_param1*, per ogni *peer* viene indicato l'intervallo di tempo in cui questo è stato connesso, nel caso quindi di connessioni discontinue, un indirizzo IP sarà affiancato da più intervalli di tempo (si veda figura 3.11).

```

{
  "104.199.192.149": [['1568816105.5455933', '1568817005.782057']],
  "108.175.2.240": [['1568212282.2876675', '1568878523.421949']],
  "110.167.89.198": [['1568276499.6977787']],
  "116.125.124.226": [['1568212282.2876675', '1568878523.421949']],
  "116.202.100.89": [['1568220684.5929203']],
  "116.202.97.138": [['1568804402.4142632']],
  "116.203.191.63": [['1568221584.753498']],
  "128.199.89.12": [['1568212282.2876675', '1568878523.421949']],
  "129.13.189.200": [['1568212282.2876675', '1568878523.421949']],
  "129.13.189.202": [['1568212282.2876675', '1568878523.421949']],
  "13.53.41.207": [['1568212282.2876675', '1568878523.421949']],
  "131.114.2.151": [['1568231487.5338876', '1568530123.2425718'],
                    ['1568531623.635092', '1568546027.300046'],
                    ['1568550828.5866337', '1568798400.6723945']],
  "132.68.60.222": [['1568307162.0943441']],
  "134.175.33.214": [['1568217383.5970106', '1568218584.0057435'],
                    ['1568799601.0979095', '1568801701.6300192'],
                    ['1568837111.517596', '1568838011.8318245']],
  "134.209.254.206": [['1568804702.4892762', '1568805002.6056473']],
  "134.209.254.216": [['1568213782.8104665']],
  "145.20.145.2": [['1568765090.7337306', '1568765390.8500304']],
  "146.185.139.235": [['1568751586.3979032', '1568751586.3979032']],
  "159.69.41.62": [['1568778294.7378025']],
  "159.89.156.60": [['1568808303.5412517']],
  "162.218.65.100": [['1568594940.6934564', '1568606643.9221272']],
  "162.218.65.104": [['1568508517.5736115', '1568514519.068664'],
                    ['1568621048.1254487', '1568626749.5957692']],
  "162.218.65.105": [['1568612645.6481466', '1568616546.7483199']],
  "162.218.65.110": [['1568495313.970944', '1568500115.1619873'],
                    ['1568501015.374027', '1568503415.986686']],
}

```

Figura 3.11: Intervalli di connessione per ogni *peer*

### 3.3 Risultati ottenuti

In questo esperimento abbiamo raccolto dati per 10 giorni dal *2019-09-11 16:31:22* al *2019-09-19 09:35:23*. Ecco espressi in tabella alcuni risultati che abbiamo ottenuto:

Numero Lanci	media peer	varianza	% nodi invariati	peer conosciuti
2220	23.767	0.218	70.1	196

max connessioni	nr. max connessioni	min connessioni	nr. min connessioni
24	1748	22	43

Sappiamo che il protocollo Bitcoin cerca di stabilire per ogni *peer* 24 connessioni con altri nodi per renderlo "ben connesso". Da questi dati vediamo che, per quanto riguarda la nostra analisi, la media delle connessioni si aggira ad un numero molto

vicino a 24 e inoltre osservando che la varianza ha un valore basso possiamo dire che in generale il numero di connessioni rimane sempre vicino al 24.

Non si è mai verificato un caso in cui si è superato questo numero di connessioni in un dato istante, ma invece questo è stato il massimo valore registrato dal nostro programma, mentre il minimo è stato 22, che da appunto conferma, come la varianza, che il numero di connessioni resta sempre "quasi" costante. La maggiorparte delle monitorizzazioni (con precisione 1748/2220) mostrano che il numero di connessioni che Bitcoin cerca di mantenere per ogni nodo è esattamente 24. Il numero di peer però con cui siamo venuti in contatto è 196, quindi significa che più volte delle connessioni sono "crollate" e se ne sono stabilite delle nuove. Dal grafico a barre riportato dal programma, è possibile analizzare la percentuale del tempo totale di connessione per ogni *peer* con il nostro *full-node*. Riportiamo qui solamente i dati che più ci interessano:

media prob. peer	varianza	% min. connessione	nr. nodi con % minima	nr. nodi stabili
0.878	0.083	0.045%	43	14

non conoscendo come si comporta il protocollo Bitcoin riguardo la topologia della rete, e quindi non sapendo se ci possano essere connessioni con alcuni *peer* più probabili di altre, abbiamo dato ad ogni *peer* incontrato una probabilità di connettersi al nostro *full-node* a seconda di quante volte è stato "incontrato" in passato. Il primo valore della tabella indica la media di queste probabilità con accanto la relativa varianza, ma questo potrebbe anche non essere quindi un dato che rispecchi effettivamente l'andamento delle cose nel protocollo Bitcoin. Viene riportato inoltre nella tabella la percentuale minima della connessione di un nodo durante la monitorizzazione. Questa

percentuale è molto bassa e come vediamo sono ben 43 i nodi che hanno il medesimo valore, questo può star a significare che c'è stato bisogno più volte di cambiare connessione prima di trovare un nodo più "stabile". Sono invece 14 i nodi che sono rimasti sempre connessi, e questi potrebbero essere nodi "stabili" della rete, che ci assicurano una connessione all'interno al network Bitcoin e che quindi probabilmente sono necessari. Per quanto riguarda la geolocalizzazione abbiamo ottenuto:

continente max	stato max	città max	continente min	stato min città min	
North America 153	United States 153	Tigar 109	Asia 18	Austria 1	Linz 1

Dove i campi indicano i luoghi e il numero (massimo e minimo ) di peer di cui siamo venuti a conoscenza, rispettivamente per continente, stato e città. La maggior parte dei *peer* che abbiamo incontrato, che ricordiamo erano 196, vengono dal nord america, e di questi tutti sono statunitensi, quindi precisamente il 78% dei *peer* sono americani. Questo potrebbe voler dire che questo è il luogo dove la tecnologia si è diffusa maggiormente? In ultimo luogo vogliamo vedere se è possibile che un qualche nodo che si disconnette dal nostro *full-node* sia propenso, o almeno ci sia qualche probabilità che possa ristabilire una connessione. Come mostato dai dati ricavati dal nostro programma di analisi, vediamo che questa situazione capita non di rado, mostriamo gli intervalli di tempo (rappresentiamo i timestamp) di alcuni peer per cui si è verificato questo fatto:

131.114.2.151	162.218.65.241	23.92.36.28
(1568231487.5338876, 1568530123.2425718)	(1568540025.947396, 1568543326.6773708)	(1568584437.7690363, 1568586538.3642185)
(1568531623.635092, 1568546027.300046)	(1568668761.6218576, 1568668761.6218576)	(1568587138.5555046, 1568595540.8496878)
(1568550828.5866337, 1568798400.6723945)	(1568669961.9677079, 1568673563.005906)	



## Capitolo 4

# Conclusioni e sviluppi futuri

In questa tesi si è fatta una panoramica di tutto ciò che riguarda l'ecosistema Bitcoin, con particolare attenzione a ciò che concerne la rete peer-to-peer che è una parte di importanza centrale di tale tecnologia. Successivamente è stato presentato un applicativo in grado di fornire informazioni riguardo una piccola parte della topologia della rete Bitcoin, cioè quella riguardante le connessioni di un singolo *full node* con i suoi *peers*.

Siamo quindi così riusciti a fare un'analisi più o meno accurata per quanto riguarda le connessioni di un singolo nodo, e i risultati di tale analisi sono stati evidenziati nel capitolo 3. Questo lavoro avrebbe come obiettivo però quello di riuscire a monitorare ogni nodo della rete, per poter inferire le connessioni non di un solo *full node*, ma di ogni partecipante della rete, per questo deve essere considerato soltanto come un passo intermedio per il raggiungimento di un obiettivo più ambizioso.

# Appendice A

**codice latex di questa tesi:**

<https://github.com/March-08/Thesis>

**applicativo : `analisi_dati.csv.py`**

[https://github.com/March-08/Thesis/blob/master/analisi\\_dati.csv.py](https://github.com/March-08/Thesis/blob/master/analisi_dati.csv.py)

**applicativo : `raccolta_dati.csv.py`**

[https://github.com/March-08/Thesis/blob/master/raccolta\\_dati.csv.py](https://github.com/March-08/Thesis/blob/master/raccolta_dati.csv.py)

# Elenco delle figure

1.1	QR code per la richiesta di pagamento in BTC . . . . .	13
2.1	Crittografia su curva ellittica: visualizzazione della curva ellittica con p=17, caso continuo . . . . .	17
2.2	Crittografia su curva ellittica: visualizzazione della curva ellittica con p=17, caso discreto . . . . .	17
2.3	Conversione da una chiave pubblica ad un indirizzo Bitcoin . . . . .	19
2.4	Un nodo della rete Bitcoin con tutte e quattro le funzionalità: <i>wallet</i> , <i>miner</i> , <i>full blockchain database</i> e <i>network routing</i> . . . . .	25
2.5	Handshake tra due peer della rete Bitcoin . . . . .	27
2.6	Propagazione di indirizzi Bitcoin . . . . .	28
2.7	Struttura dei blocchi . . . . .	32
2.8	calcolo dei nodi in un merkle root . . . . .	34
2.9	Merkle path usato per provare l'inclusione dell' elemento K . . . . .	35
3.1	<a href="https://bitcoincore.org/en/doc/0.16.0/rpc/network/getpeerinfo/">https://bitcoincore.org/en/doc/0.16.0/rpc/network/getpeerinfo/</a> . . .	44
3.2	Esempio lancio del programma <code>raccolta_dati.py</code> . . . . .	45
3.3	Esempio dati raccolti.py . . . . .	46
3.4	Esempio lancio del programma <code>analisi_dati.py</code> . . . . .	46
3.5	Date di inizio e fine monitorizzazioni . . . . .	46

3.6 Percentuale dei *peer* invariati per ogni lancio . . . . . 47

3.7 Statistiche sulle connessioni dei *peer* . . . . . 48

3.8 Grafico a barre . . . . . 48

3.9 Statistiche sulla percentuale di connessione di ogni *peer* . . . . . 49

3.10 Geolocalizzazione dei *peer* . . . . . 49

3.11 Intervalli di connessione per ogni *peer* . . . . . 50

# Bibliografia

- [1] Andreas M Antonopoulos. *Mastering Bitcoin: Programming the open blockchain*. ” O’Reilly Media, Inc.”, 2017.
- [2] Sergi Delgado-Segura, Surya Bakshi, Cristina Pérez-Solà, James Litton, Andrew Pachulski, Andrew Miller, and Bobby Bhattacharjee. Txprobe: Discovering bitcoin’s network topology using orphan transactions. *arXiv preprint arXiv:1812.00942*, 2018.
- [3] Andrew Miller, James Litton, Andrew Pachulski, Neal Gupta, Dave Levin, Neil Spring, and Bobby Bhattacharjee. Discovering bitcoin’s public topology and influential nodes. *et al*, 2015.
- [4] Satoshi Nakamoto et al. Bitcoin: A peer-to-peer electronic cash system. 2008.
- [5] Till Neudecker, Philipp Andelfinger, and Hannes Hartenstein. Timing analysis for inferring the topology of the bitcoin peer-to-peer network. In *2016 Intl IEEE Conferences on Ubiquitous Intelligence & Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCoM/IoP/SmartWorld)*, pages 358–367. IEEE, 2016.

- [6] Florian Tschorsch and Björn Scheuermann. Bitcoin and beyond: A technical survey on decentralized digital currencies. *IEEE Communications Surveys & Tutorials*, 18(3):2084–2123, 2016.