POLITECNICO DI TORINO

Laurea Magistrale in COMPUTER ENGINEERING

Tesi di laurea Magistrale

Soluzioni innovative alle debolezze di Bitcoin

Relatore

Prof. DANILO BAZZANELLA

Candidato

LUCA SAGLIA

OTTOBRE 2022

Sommario

Il mondo delle criptovalute ha avuto una forte crescita negli ultimi anni e al giorno d'oggi basta guardare un evento sportivo per essere bombardati da pubblicità di ogni tipo provenienti da questo settore.
Bitcoin, nata nel 2008, è stata la prima vera implementazione di questo concetto ed è anche la più popolare, complice anche il forte aumento di valore che ha subito tra il 2020 e la fine del 2021. Il fatto che l'attenzione di molti si sia focalizzata su questa tecnologia ha fatto sì che alcuni problemi venissero a galla: Bitcoin, infatti, nonostante sia di per sé un concetto innovativo sotto diversi punti di vista, ha numerose lacune legate principalmente alla sua scalabilità.
Tra le principali problematiche troviamo:

- L'eccessivo consumo energetico derivato dal processo di "mining".
- Il problema dei rifiuti elettronici causati dalla rapida obsolescenza dell'hardware destinato al mining.
- Il limitato numero di transazioni che possono essere compiute in un secondo.

Per risolvere queste problematiche sono state pensate diverse soluzioni: alcune hanno tentato di costruire sopra la blockchain di bitcoin un secondo 'strato' su cui è possibile eseguire le transazioni senza subire le limitazioni di velocità imposte dall'architettura di Bitcoin, queste vengono appunto chiamate "soluzioni layer 2".
Un altro approccio è stato quello di implementare un algoritmo di consenso differente da quello di Bitcoin, in modo da non basarsi sulla potenza di calcolo ma su altre risorse che non richiedano un consumo di energia così elevato. In questo studio si è voluto implementare fisicamente una soluzione per entrambi questi approcci e valutarne la resa economica portata dal contributo che un utente può dare alla rete.
Come soluzione Layer 2 si è scelto Lightning Network che è sicuramente la più conosciuta e si basa sul concetto della "payment channel network", ovvero una rete di pagamenti formata da canali peer-to-peer.
Per quanto il secondo approccio si è scelto di implementare un nodo della blockchain Chia Network la quale basa il suo protocollo di consenso sulla così detta "proof of space and time". Questo protocollo anziché sfruttare la potenza di calcolo sfrutta lo spazio di archiviazione garantendo un consumo energetico inferiore e riducendo drasticamente il problema dell'obsolescenza.
Per entrambe queste soluzioni si è quindi descritto il processo di messa a punto di hardware e software e lo studio della possibile rendita economica portata dalla contribuzione alla rete.

Ringraziamenti

Prima di cominciare questa trattazione mi sembra doveroso fare alcuni dovuti ringraziamenti:

Vorrei ringraziare il mio relatore Danilo Bazzanella per la grande disponibilità e il supporto che mi ha garantito durante questo percorso di tesi.
Ringrazio Mynecrypto SRL che mi ha dato la possibilità di lavorare a questo progetto e di immergermi nella realtà di una giovane startup, imparando molte cose e approcciando il mondo del lavoro da diversi punti di vista.

Questa laurea per me rappresenta il più grande traguardo che fin'ora ho mai raggiunto e vorrei condividerlo con le persone che più mi sono state vicine in questi anni.
Un Grazie speciale ai miei genitori che mi hanno supportato sotto ogni punto di vista, sempre e comunque, dal giorno del TIL ad oggi credendo fermamente in me e al fatto che questo giorno sarebbe arrivato.
Grazie a mia Nonna e mia Zia che non si sono mai tirate indietro per aiutarmi fin da quando ero un bambino e che continuano a farlo anche adesso ogni volta che possono.
Grazie agli amici di sempre e a quelli che ho conosciuto a Torino in questo percorso, che hanno fatto sì che questi anni siano stati ricchi di ricordi da raccontare con un sorriso.
E infine grazie a quel ragazzo di 20 anni che non si è lasciato scoraggiare dalle prime difficoltà di questo percorso, grazie al (più) giovane me per aver sognato questo giorno.

Indice

Capitolo 1

Introduzione

1.1 Incipit

Le criptovalute, se fino a qualche anno fa me lo avessero detto che sarei stato qui a scrivere una tesi di laurea sulle criptovalute non ci avrei creduto.
Ricordo che me ne parlò un mio amico nel 2010 ma avevamo entrambi 13 anni e ne rimasi abbastanza confuso: cosa voleva dire che "le monete erano dei codici"? Poco dopo già stavo correndo appresso ad un pallone o giocando alla PlayStation, non ci diedi grande peso. Tuttavia un po' perché mi ha sempre intrigato la tecnologia, un po' perché sono curioso mi sono ripromesso che un giorno avrei approfondito l'argomento, non sapevo quando, ma ero certo che lo avrei fatto.
Gli anni passano e il mondo delle criptovalute, che nel frattempo è cresciuto a dismisura, sembrava essere svanito completamente dal mio radar; in parte anche perché io stesso nel tempo avevo sviluppato un po' di scetticismo verso questo argomento.
La mia occasione di fare chiarezza si concretizza nel 2021 quando ho l'occasione di seguire in quest'università il corso 'Blockchain e Cryptoeconomia', da lì rimango affascinato dal concetto della blockchain, tanto che ho deciso di parlarne in questa tesi.

1.2 Un po' di storia

Il primo abbozzo di criptovaluta vera e propria lo si ha nel lontano 1998 con il paper "b-money, an anonymous, distributed electronic cash system" [Dai, 1998] dell' ingegnere informatico Wei Dai dove vengono già citati alcuni degli elementi principali delle criptovalute. Non fu mai implementato, tuttavia, verrà poi accreditato nel whitepaper di Bitcoin l'importante contributo che diedero questi concetti.
Solo nel 2008, 10 anni dopo, il 31 ottobre un certo Satoshi Nakamoto pubblica il

paper “Bitcoin: A Peer-to-Peer Electronic Cash System” [Nakamoto, 2008] dove viene descritto per la prima volta il protocollo Bitcoin, pochi mesi dopo nel gennaio 2009 Nakamoto effettua la prima transazione e da lì prenderà vita la rete Bitcoin. La vera identità di Nakamoto è tutt'ora sconosciuta e, seppure ci siano diverse teorie interessanti, ritengo personalmente che sia meglio così; ciò associa a Bitcoin l'immagine di un bene collettivo: “di tutti ma di nessuno” e inoltre, questo avvolge la sua genesi con un velo di mistero.

1.3 Alla base delle criptovalute

La comprensione di Bitcoin è fondamentale per imparare la logica dietro le criptovalute, ognuna di queste tende a distaccarsi da Bitcoin sotto certi aspetti ma ne resta fedele ai principi base.
Per parlare di questa tecnologia è importante prima aver chiari alcuni concetti, per questo ci terrei a dare alcune definizioni (semplicistiche) dei 3 ingredienti principali di una blockchain:

1.3.1 Funzione di hash

La funzione di hash è una funzione che prende come parametro in input un qualsiasi dato e ne restituisce una stringa di lunghezza fissa, questa stringa ha alcune caratteristiche molto importanti:

- Dato lo stesso parametro in input si avrà sempre la stessa stringa in output.
- Data la stringa in output è molto difficile, se non impossibile risalire al dato in input.
- Una qualsiasi variazione del dato in input causa uno stravolgimento della stringa in output. possiamo quindi pensare alla funzione di hash come all'impronta digitale di un dato, qualora quest'ultimo cambiasse, anche il suo hash cambierà.

Per esempio:
`HASH(rosso) = 959ea3daf0e8531c5753d56999e8adff7f24b077`
`HASH(fosso) = 7ad91706b40f6b68fe99b421720279f0e829789f`

1.3.2 Crittografia asimmetrica

La crittografia asimmetrica consente, mediante la coppia di chiavi note come chiavi privata e chiave pubblica, di cifrare i dati consentendo di decifrare ciò che è stato cifrato con la chiave pubblica solo con la chiave privata e viceversa. Mantenendo

segreta la chiave privata e distribuendo quella pubblica è possibile sfruttare la crittografia asimmetrica in due modi:

- le persone che devono comunicare con noi cifrano i messaggi con la nostra chiave pubblica, in questo modo solo noi possiamo decifrare quei messaggi

$$cifro(DATO, pubK) = DATO\_CIFRATO \tag{1.1}$$

$$decifro(DATO\_CIFRATO, pvtK) = DATO \tag{1.2}$$

- Se vogliamo provare che siamo noi gli autori di un determinato messaggio possiamo cifrarlo con la nostra chiave privata, in questo modo le altre persone decifrando il messaggio con la nostra chiave pubblica possono provare che siamo noi gli autori, in quanto unici possessori della chiave privata

$$cifro(DATO, pvtK) = DATO\_CIFRATO \tag{1.3}$$

$$decifro(DATO\_CIFRATO, pubK) = DATO \tag{1.4}$$

1.3.3 Firma digitale

La firma digitale non è altro che la combinazione dei due strumenti precedenti: se vogliamo firmare un determinato dato, per esempio una transazione, possiamo eseguirne la funzione di hash, in modo da ridurne la dimensione ad un valore fisso e successivamente cifrare con la nostra chiave privata l'hash. In questo modo chiunque può verificare che siamo noi gli autori di una determinata transazione confrontando l'hash di quest'ultima con il valore che si ottiene decifrando con la nostra chiave pubblica la firma della transazione. I due valori devono risultare identici.

1.4 Bitcoin

1.4.1 Funzionamento

Il whitepaper di Nakamoto ha l'ambizione di rivoluzionare i metodi di pagamento elettronici, sostiene infatti che la vera debolezza di questi ultimi sia il modello basato sulla fiducia in un ente intermediario (istituzioni finanziarie) e tutti i problemi che questo comporta. Per ovviare a questo problema propone una soluzione basata sul peer-to-peer in cui non vi è più alcun ente centrale: ogni transazione

è firmata digitalmente dal pagante la cui chiave pubblica è legata all'indirizzo su cui precedentemente ha ricevuto una somma di bitcoin pari o maggiore di quella che sta cercando di pagare. La fiducia viene posta sullo storico delle transazioni contenuto nella blockchain che ogni nodo della rete possiede e può verificare.
La difficoltà si sposta quindi su un altro punto: ogni nodo della rete deve costantemente avere in memoria la blockchain contenente le stesse transazioni nello stesso ordine. Qui entra in gioco il cosiddetto meccanismo di consenso: ovvero la strategia che deve attuare il protocollo per decidere quale blocco di transazioni aggiungere di volta in volta all' interno della blockchain, e nel caso di Bitcoin si tratta della proof of work (POW).
Ogni nodo che partecipa alla rete di Bitcoin può decidere di contribuire alla rete raccogliendo le transazioni, che vengono comunicate pubblicamente, inserendole in un nuovo blocco e proponendo quest'ultimo agli altri membri della rete.
Perché sia accettato dagli altri il blocco prodotto deve prima di tutto contenere transazioni valide, e questo è facilmente verificabile in quanto ogni altro nodo possiede una copia della blockchain.
Dopodichè, siccome ogni nodo potrebbe proporre un blocco in qualsiasi istante (potrebbero esserci N blocchi proposti, quale scegliere unanimamente? Come sincronizzarsi?) c'è la necessità di un sistema di arbitraggio che regoli la decisione del blocco successivo. Un blocco proposto è quindi promosso solo se il valore del suo hash contiene X zeri iniziali, dove X è un numero variabile ed è noto come "difficoltà". Il nodo per soddisfare questa condizione deve effettuare un indefinito numero di volte la funzione di hash del blocco andando a modificare ogni volta un apposito valore all'interno di quest'ultimo in modo da avere risultati diversi per ogni tentativo.
Questo processo può essere visto come un puzzle e anche per un calcolatore elettronico può richiedere una quantità di tempo non indifferente, ciò permette di garantire che non ci siano troppi blocchi proposti nello stesso istante (idealmente uno ma possono verificarsi i cosiddetti casi di fork). Non è solo una questione di tempo però: visto che l'hash ha un valore imprevedibile, risolvere questo puzzle è computazionalmente molto complesso e perciò solo chi è dotato di un hardware sufficientemente potente può completarlo in tempi ragionevoli. Possedere questo hardware comporta l'aver effettuato una spesa economica proporzionale alla potenza e quindi alla probabilità di risolvere questo puzzle e ciò implica che solo chi ha investito nel progetto (e quindi ci tiene al corretto funzionamento di Bitcoin) può pubblicare blocchi validi.
Questo concetto appena descritto è quindi la sopra citata Proof of work, è inoltre bene specificare che i nodi che compiono questo lavoro sono detti "miners" e ogni volta che riescono a pubblicare un blocco valido per primi si accaparrano una remunerazione che è composta dalle commissioni delle transazioni contenute nel blocco e da una ricompensa aggiuntiva (a patto che qualcuno non sia arrivato prima,

in quel caso devono riiniziare da capo con un nuovo blocco e nuove transazioni) il ché li incentiva a mandare avanti l'infrastruttura.

1.4.2 La difficoltà

Con il passare degli anni e con l'aumentare del numero di miners però risolvere il puzzle è diventato sempre più difficile: il numero di zeri richiesti all'inizio dell'hash del blocco infatti non è costante ma varia in continuazione ed è comunemente indicato come "difficoltà". Il protocollo Bitcoin è in grado di autoregolarsi in modo che si riesca a minare mediamente un blocco valido ogni 10 minuti e lo fa proprio aumentando o diminuendo la difficoltà.

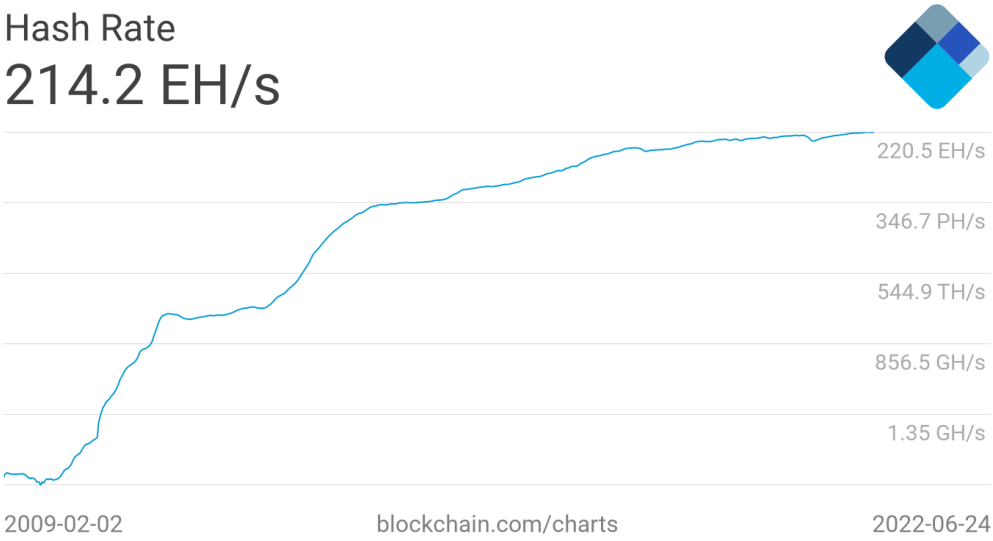


Figura 1.1: Grafico tratto da https://www.blockchain.com/charts/hash-rate

se prima del 2010 il numero di hash al secondo che veniva compiuto da tutti i nodi Bitcoin (hashrate complessivo) era di pochi megahash (10^6) ai giorni d'oggi siamo nell' ordine di grandezza degli ettahash (10^{18}). Questo ci fa capire la grande attenzione che è riuscito ad attirare su di sé Bitcoin negli ultimi anni.

1.4.3 La scarsità digitale

La più grande proprietà di Bitcoin, ma in generale della maggior parte di tutte le criptovalute è la sua capacità di creare scarsità digitale: il numero di bitcoin circolanti infatti non è illimitato, ma ha un tetto fissato a 21 milioni di bitcoin,

terminati questi i miners non riceveranno più la ricompensa per i blocchi minati ma si limiteranno a incassare le commissioni delle transazioni contenute nel blocco che hanno appena "minato". Ciò rende quindi Bitcoin una "risorsa" difficile da ottenere e relativamente rara, la continua affluenza di richiesta è inferiore alla quantità che riesce ad essere immessa sul mercato e questo fa si che si che abbia un comportamento deflattivo.

1.4.4 I limiti

Il problema energetico ed ambientale

In questo ultimo decennio abbiamo visto con i nostri occhi una presa di coscienza da parte della società per quanto riguarda il problema dell'inquinamento, una battaglia importante che forse rappresenta la sfida più grande per la mia generazione.
Nella primavera del 2021 Bitcoin è stato preso molto di mira proprio per il suo dispendio energetico: si calcola che annualmente consumi un ammontare di energia paragonabile al fabbisogno annuale della Finlandia [Huang et al., 2021]. Il mining è stato, fino alla prima metà del 2021, perlopiù concentrato in zone in cui vengono sfruttati intensivamente i combustibili fossili e si stima che ogni transazione di Bitcoin rilasci circa 300 kg di CO2 nell'aria [Ponciano, 2021].

Secondo un report di Galaxy Digital [Rybarczyk et al., 2021] invece i consumi di Bitcoin non sono cosi elevati se confrontati con la loro stima effettuata sul sistema bancario tradizionale. Si riporta infatti che i consumi annui di Bitcoin si aggirano intorno ai 114TWh mentre quelli del sistema bancario sono stimati intorno ai 264TWh. Viene inoltre riportato che il processo di estrazione e raffinazione dell'oro (riserva di valore per eccellenza) consuma all'incirca 241TWh ogni anno.
Ammesso che i sistemi di pagamento tradizionali consumino effettivamente di più, ad onor del vero, bisogna tenere in considerazione che Bitcoin è attualmente una nicchia e quindi rimane in proporzione meno efficiente, ma al crescere del numero di utenti Bitcoin il consumo energetico non necessariamente varierebbe significativamente dal momento che che il numero di miners attuali riesce già a garantire una sicurezza sufficiente.

Un' altro dato interessante da prendere in esame è quello dello spreco energetico causato dai dispositivi in standby: secondo l'agenzia internazionale dell' energia [IEA, 2014] nell' anno 2013 i dispositivi in standby hanno consumato all' incirca 616TWh di energia (questo dato con ottima probabilità è aumentato negli ultimi anni vista l'enorme diffusione di dispositivi IOT) su un totale di 23'503TWh di energia prodotti nel mondo durante lo stesso anno.
Tutto ciò non non per giustificare la spesa energetica di Bitcoin, ma è importante evidenziare che quando si prende in considerazione il sistema "mondo" le cifre sono sempre inevitabilmente molto alte.

Vorrei poi soffermarmi sulla profonda soggettività che ci può essere nel definire una risosrsa, in questo caso l'energia, sprecata o meno.

il problema dei rifiuti digitali

Oltre che il consumo energetico, un' altra questione che bisogna affrontare è quella dei rifiuti elettrici: la continua crescita della difficoltà porta l'hardware utilizzato a risultare obsoleto e ad essere sostituito molto velocemente.
Un recente studio [de Vries and Stoll, 2021] ha stimato che a maggio 2021 erano all' attivo un minimo di 39,75 kilotonnellate di hardware destinato al mining e che ogni device ha un rendimento utile di circa 1,29 anni, Ciò produce circa 30kilotonnellate di rifiuti hardware ogni anno. Come si è arrivati fin qui?
Il mining inizialmente veniva praticato con semplici CPU, ma ben presto ci si è resi conto che questo processo poteva essere reso più efficiente sfruttando la capacità di parallelizzare le operazioni delle GPU. Vista la grande resa che offriva questa attività ci si è spinti oltre progettando dei dispositivi hardware in grado di eseguire in maniera specifica un determinato algoritmo (quello richiesto per il mining della criptovaluta) nella maniera più efficiente possibile, ovvero i cosiddetti ASIC (Application Specific Integrated Circuit. Con questo nome si indica generalmente un dispositivo elettronico destinato ad un singolo e specifico utilizzo, tuttavia è molto utilizzato nel mondo delle criptovalute per indicare appunto un ASIC destinato al mining).
A differenza di CPU e GPU questi dispositivi una volta obsoleti per il mining non possono venire utilizzati per altre operazioni e perciò diventano rapidamente rifiuti. Il mining al momento è ancora troppo redditizio e conviene quindi investire in hardware più efficiente piuttosto che rimanere indietro in questa corsa alla potenza computazionale.
Le soluzioni proposte dagli autori dello studio già citato sono interessanti: oltre che un eventuale passaggio alla Proof Of Stake si ipotizza l'implementazione di un sistema per far variare il tipo di algoritmo di mining. In questo modo si scoraggia l'utilizzo di ASIC a favore di HW meno specifico come le GPU le quali possono trovare seconda vita in vari utilizzi (gaming, video editing...) una volta obsolete per il mining. Questa strategia è gia stata implementata dalla criptovaluta Monero, il cui mining risulta più efficiente tramite CPU (componente con un ampio spettro di utilità) rispetto che con qualunque altro HW.

Il problema della velocità

Un ulteriore dato che spinge la comunità crypto a cercare un' alternativa è quello della velocità. Possiamo valutare la velocità con due parametri: il flusso e la latenza.

Con flusso si intende il numero di transazioni al secondo che può compiere il sistema: Bitcoin consente un massimo di circa 7 transazioni al secondo, che è un valore ridicolo se confrontato alle transazioni dei sistemi di pagamento Visa o Mastercard (ben oltre le 1000 tps). Questo valore è limitato dal fatto che un blocco della blockchain ha una dimensione limitata (1 MB ogni blocco) e come detto in precedenza il sistema è autoregolato perché si abbia un blocco ogni 10 minuti circa.

---

1 MB ogni blocco
250B dim. minima transaz. (considerando la dimensione minima per ottenere il limite massimo di transazioni per blocco)
1 blocco ogni 10 minuti
=> 1'000'000/250 = 4000 transazioni per blocco
=>4000 tx /(10*60 s)=40/6 tx/s= 6.67 tx/s (*)
*calcolo approssimativo che non tiene dello spazio occupato dall'header del blocco e approssima la grandezza di 1 MB a 1000000B

Con latenza si intende il tempo che si dovrà attendere per vedere una transazione completata Questo limite è imposto da due fattori:

- La propagazione delle transazioni tramite la rete peer to peer, esse vengono tramandate da un nodo all' altro tramite "passaparola" ("gossip protocol") e il processo richiede un tempo non indifferente
- Il tempo di intercorrenza tra un blocco e l'altro (come già detto, 10 minuti in media)

l

E' inoltre bene specificare che in una blockchain una transazione è da ritenersi confermata solo dopo che sono stati inseriti un certo numero di ulteriori blocchi dopo il blocco in cui è situata, (questo perché potrebbero verificarsi eventuali fork, non analizzeremo però questo caso in questa tesi).

Il problema dello spazio di archiviazione

Supponiamo però ora per assurdo che Bitcoin riesca (utopicamente) ad avere un flusso pari a quello di un circuito Visa, quindi in media 1700 transazioni al secondo: se una transazione 250B allora ogni secondo si avranno:
250*1700 = 425000B 0,4MB
in un ora quindi:
0,4*60*60 = 1440MB 1,4GB
In un giorno la blockchain crescerebbe quindi di oltre 30Gb.

Ad oggi chi ha intenzione di implementare un full node di Bitcoin ha bisogno di uno spazio di archiviazione di oltre 400Gb e la sua dimensione cresce in maniera

lineare di circa 60Gb ogni anno, si potrebbe quindi stimare che Bitcoin tra 10 anni possa raggiungere la dimensione di un terabyte, il che risulterebbe ancora una dimensione ragionevole, specie in vista di eventuali evoluzioni future dei sistemi di archiviazione.
Il limite delle 7 transazioni per secondo riesce quindi a mitigare questo problema, tuttavia anche se un giorno si riuscisse ad aumentare il flusso delle transazioni ci si scontrerebbe contro quest'altra grande difficoltà.

Il Problema delle fee non proporzionali

Mentre nella stragrande maggioranza dei casi i tradizionali metodi di pagamento elettronico impongono imposte di transazione di pochi punti percentuali sull'importo, Bitcoin impone una spesa di transazione che non dipende dalla cifra da trasferire ed è variabile a seconda dell'andamento del mercato (vedi 2.5).
Il prezzo mediano di una transazione nell'ultimo anno è stato di circa $2,26 con un picco massimo di $10,95 e un minimo di $1,04. Ciò è dovuto al fatto che i 'miners', che hanno il compito di inserire le transazione nei blocchi (i quali verranno poi elaborati al fine di risolvere il 'puzzle' precedentemente descritto), danno priorità alle transazioni che hanno una tassa maggiore (infatti in verità non esiste una tassa fissa o minima, si basa tutto sul principio di domanda/offerta) e perciò le transazioni con fee troppo basse non verranno pressoché mai validate. Questo rende Bitcoin un ottimo metodo per trasferire ingenti somme in denaro (rompendo in parte anche i limiti geopolitico) ma un pessimo sistema per effettuare microtransazioni.

Tutti queste questioni sono ben note da tempo e diverse soluzioni stanno emergendo nel panorama delle criptovalute che è in continua evoluzione, in questa tesi ne descriverò alcune.

1.5 Oltre Bitcoin

Se è vero che Bitcoin rimane la criptovaluta per antonomasia, in quanto la capostipite e anche la più capitalizzata, non si può non considerare che con il tempo sono nati numerosi progetti, il più importante di tutti è certamente Ethereum. Ethereum porta con sé la novità degli smart contracts (in verità, per dovere di cronaca, qualcosa di simile era già presente su Bitcoin ma a livello molto primordiale) ovvero aggiunge la possibilità di programmare dei veri e propri contratti con un linguaggio di programmazione (Solidity).
La più grande applicazione degli smart contracts finora riscontrata è quella finanza decentralizzata la quale permette di ottenere prestiti con collaterale in criptovalute senza intermediario, finanziare startup, investire in borsa (tramite la tokenizzazione

delle azioni) e molto altro; tutto ciò senza la necessità di passare per un intermediario ma semplicemente possedendo un wallet sul proprio smartphone.
Questo ha aperto nuovi orizzonti e migliaia di sviluppatori stanno al momento lavorando su applicazioni decentralizzate, e, anche su nuove blockchain che come Ethereum integrano gli smart contracts.

Capitolo 2

Lightning Network

2.1 Panoramica e funzionamento

Lightning network nasce con l'ambizione di risolvere il problema della velocità e della scalabilità di Bitcoin. Questa tecnologia consente infatti a due nodi della rete di eseguire un indefinito numero di transazioni tra loro pagando effettivamente solo 2 commissioni sulla rete Bitcoin: uno per aprire un canale di pagamento bilaterale ed uno per chiuderla.
Il canale bilaterale di pagamento viene "caricato" con una somma arbitraria di btc che risulterà bloccata sulla rete btc (mainnet) finché non verrà richiuso con un'altra transazione sulla mainnet. Finché il canale rimane aperto le due controparti possono scambiarsi il denaro che loro hanno bloccato tra di loro con commissioni nulle in quanto in pratica una transazione di denaro nel canale non è altro che una decisione concordata di spostare il bilanciamento del denaro caricato sul canale: se inizialmente A ha caricato 5 btc sul canale che esso ha creato verso B, quando A avrà pagato ad A 1 btc a B allora risulterà che A possiede 4 btc e B 1 btc.
Le transazioni vengono firmate ma non vengono caricate sulla rete principale finché uno dei due attori non decide di chiudere il canale. In quel momento la seconda transazione sulla mainnet verificherà l'effettivo bilanciamento di denaro tra le due parti esaminando le transazioni che gli sono state passate dall'attore che ha deciso di chiudere il canale e scriverà sulla blockchain principale i valori risultanti per ciascun attore. Tuttavia chi decide di chiudere il canale per primo potrebbe non esporre allo smart contract (perché di questo si tratta, una delle rare forme di smart contract di Bitcoin) tutte le transazioni avvenute nel canale, magari proprio nascondendo quell'ultima transazione in cui aveva pagato la sua controparte con una cospicua somma. E' bene quindi sottolineare che questo protocollo gode di un sistema antifrode, dove qualora la controparte si accorgesse che il canale fosse stato chiuso in maniera fraudolenta, come nel caso descritto, allora può reclamare

entro 24 ore l'irregolarità esponendo la transazione mancante che, come tutte le transazioni sul canale, è stata firmata digitalmente da entrambi e quindi non può essere ripudiata. Così facendo ha diritto al rimborso e la parte truffatrice perde tutti i fondi che possedeva sul canale, anche quelli a cui aveva diritto. Per far sì che ciò funzioni però il truffatore non deve spendere la somma sottratta prima che scadano le 24 ore in cui è possibile fare il reclamo, infatti dopo la chiusura di un canale nessuno dei due attori può spendere la somma ritirata prima di 24 ore.

Il meccanismo appena descritto risulta molto comodo se due entità devono scambiarsi denaro molto frequentemente, in quel caso vale la pena creare un canale, tuttavia se ci fosse la necessità di fare un pagamento non ricorrente non sarebbe efficiente, toccherebbe infatti aprire e chiudere un canale (quindi 2 transazioni sulla mainnet con relative commissioni) per eseguire una sola transazione di btc. E' tuttavia possibile sfruttare questi canali punto-punto per creare una rete, in questo modo si possono effettuare pagamenti sulla rete lightning anche se non si ha aperto un canale diretto con il destinatario.
In breve: se A (pagante) non possiede un canale aperto con B (ricevente) ma entrambi possiedono un canale con C allora quest'ultimo può essere usato come ponte: C verserà dapprima la somma da inoltrare a B e solo dopo si farà rimborsare da A. Il nodo C è incentivato a fare da intermediario perché così facendo avrà diritto ad una piccola tassa di commissione.
Ora il dubbio sorge spontaneo: cosa impedisce a A di rifiutarsi di rimborsare C dal momento che B ha già ricevuto il suo pagamento a nome di A? Il meccanismo è sicuro perché sfrutta delle transazioni condizionali: C pagherà B solo dopo che A gli avrà firmato digitalmente un contratto in cui garantisce che se gli mostrerà una prova di pagamento a B allora verrà rimborsato. La prova di pagamento non è altro che una chiave segreta conosciuta inizialmente solo da B il quale avrà distribuito preventivamente ad A l'hash di quest'ultima in modo da effettuare poi la verifica. Questo procedimento può essere ripetuto con N 'nodi ponte' creando cosi una vera e propria rete per i pagamenti.

C'è tuttavia una questione da tenere in considerazione: quando A sceglie di aprire un canale verso di B sarà il primo a dover preoccuparsi di caricare la liquidità iniziale, questo farà sì che B non potrà fare pagamenti verso A a meno che prima A non sposti della liquidità dall'altro lato (con eventuali pagamenti), si dice in questo caso che il canale non è bilanciato, la liquidità che si ha a disposizione per effettuare pagamenti prende il nome di "outbound capacity", mentre quella che si ha a disposizione per riceverli è la "inbound capacity". Per avere un nodo efficiente è sempre bene cercare di avere canali il più possibile bilanciati. E' possibile bilanciare i propri canali tramite appositi algoritmi a patto che si abbia più di un canale aperto, tuttavia questo non va a variare l'inbound capacity complessiva di tutti i canali.

Inoltre questa funzione è ancora scarsamente disponibile (vedremo in seguito). Per questo motivo è molto importante riuscire ad attirare più nodi possibili ad aprire un canale verso il proprio nodo, in modo da avere un' elevata inbound capacity: più inbound capacity si ha, maggiore outbound capacity è possibile far fruttare per inoltrare transazioni (e quindi più profitto sulle commissioni).

2.2 L'implementazione

Mi è stato richiesto presso l'azienda in cui faccio tirocinio di implementare un nodo lightning network per due motivi principali: studiare la possibilità di lucrare sull'inoltro delle transazioni come pocanzi descritto avere la possibilità di creare dei canali con eventuali altri indirizzi in modo da trasferire con ricorrenza bitcoin senza commissioni.

2.2.1 Le scelte hardware e software

Ho scelto di adoperare un Lenovo ThinkCentre Edge 72 in quanto già disponibile in azienda. Dispone delle seguenti specifiche:

- cpu: intel core i5 443305
- ram: 8gb
- storage: 1TB HDD

E' una macchina di qualche anno, ma riesce a fare quello che deve. Un lightning node non richiede infatti particolari risorse hardware, se non uno spazio di storage relativamente grande in quanto la blockchain di bitcoin si aggira attorno ai 400GB. Un' altra opzione hardware presa in considerazione è quella del raspberry PI 4, disponibile sul mercato con un prezzo (al momento) inferiore ai 100€. Risulta essere un' opzione interessante perché grazie all'architettura della sua CPU riesce a garantire dei consumi molto ridotti. Esistono molte soluzioni all-in-one che integrano nodo Bitcoin e nodo LN, queste sono facilmente configurabili e molto user friendly.
Considerando una spesa nulla per il Thinkstation in quanto gia disponibile in azienda ed una spesa di circa 85 euro per il Raspberry PI4 si è constatato che la spesa sarebbe rientrata dopo circa 400 giorni di utilizzo. Si è scelto di utilizzare comunque il Lenovo in quanto come azienda appena avviata si voleva limitare la spesa iniziale, ed eventualmente valutare di passare a Raspberry in seguito.

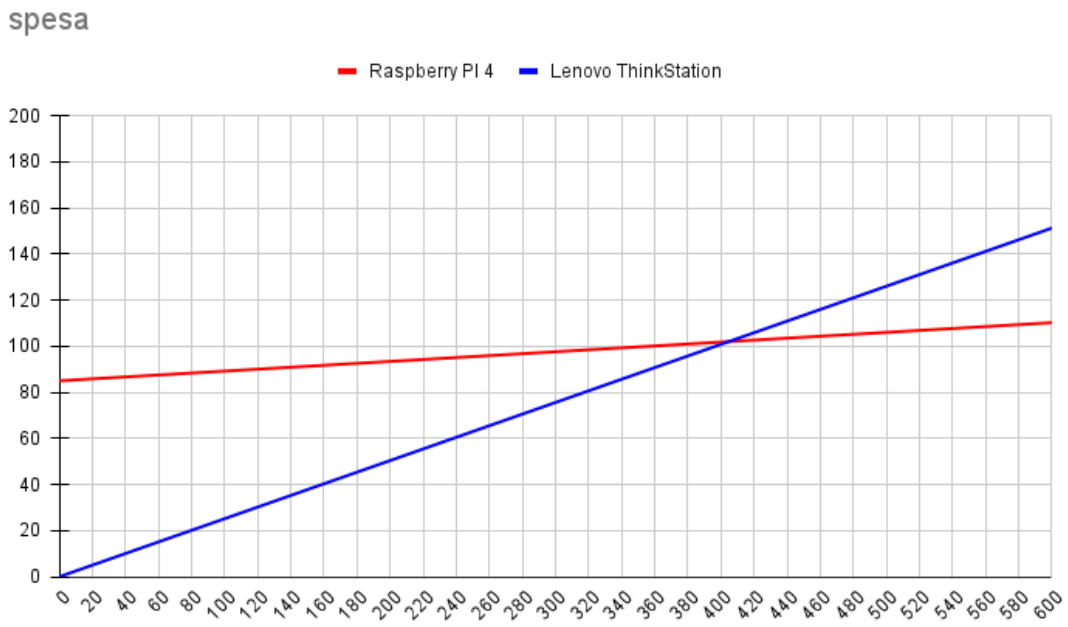


Figura 2.1: asse X: giorni; asse Y : spesa in € Considerando:
assorbimento medio RB PI4: 6W
assorbimento medio Lenovo : 30W
costo energetico: 0.35€/kWh

Sul Lenovo è stato installato il sistema operativo Ubuntu 20.04.03 LTS il quale garantisce supporto fino ad aprile 2025. La scelta di Ubuntu mi è sembrata la scelta più appropriata per una macchina server data la presenza della shell bash e la semplicità con cui è possibile gestire i servizi tramite systemd (come ssh server). In più garantisce un buon livello di supporto grazie all' enorme community che gli gira attorno e la maggior parte delle guide per implementare sistemi server sono per OS unix-like.
Ho scelto la versione desktop di Ubuntu e non quella server per motivi di praticità, nonostante infatti sia possibile controllare la macchina tramite altri pc via SSH inizialmente avrei comunque voluto installare un'interfaccia grafica della lightning network solo sulla macchina host (ne parleremo dopo in dettaglio). Questo a discapito di appesantire lievemente il carico di lavoro per via della GUI di Ubuntu, che tuttavia può essere eventualmente disattivata in seguito.
Per quanto riguarda Il nodo bitcoin e quello lightning network, come ho accennato in precedenza, sono già disponibili dei pacchetti all-in-one, come ad esempio Umbrel: Questo software nasce inizialmente per raspberry PI ed è ad oggi possibile farlo

girare su una qualsiasi macchina, tuttavia quest'ultima opzione è disponibile solo in via sperimentale. La mia scelta è stata di installare inizialmente Umbrel per prendere dimestichezza con l'ambiente della lightnting network facendo qualche transazione sulla rete di test[1]. Tuttavia quest'implementazione non è l'ideale per un prodotto destinato ad un uso aziendale: Si tratta di una soluzione build-in, tutto funziona, ma non si può sapere esattamente come, cercare di capirlo rischia di risultare più complesso di costruirsi una soluzione, inoltre è un progetto relativamente nuovo che potrebbe contenere ancora molti bug e, come accennato in precedenza, la versione disponibile per PC è sperimentale.
Ho scelto cosi di implementare una soluzione personale in cui andare a scegliere singolarmente ogni parte del nodo.

Le tre parti principali in cui si articola un nodo lightning network sono le seguenti:

- Nodo Bitcoin
- Software Lightning node
- interfaccia grafica (opzionale)

2.2.2 Il nodo Bitcoin

Il nodo bitcoin è molto importante perché fornisce al nostro LN (Lightning Node) l'accesso alla blockchain. Sarebbe possibile collegare il LN anche a nodi Bitcoin di terzi tuttavia l'azienda ha preferito usufruire di un nodo personale per questioni di sicurezza ed affidabilità.
Sono disponibili vari software che implementano Bitcoin, io ho optato per Bitcoin Core per via della sua grande adozione che garantisce un buon supporto e vasta documentazione. La maggior parte delle guide di implementazione di nodi LN, infatti, suggerisce di utilizzare questo software. E' importante sottolineare che si tratta di un full node, ovvero un nodo che contiene tutta la blockchain di Bitcoin, in grado di generare e firmare transazioni. Il setup di questo componente risulta relativamente facile, è possibile anche utilizzare un'interfaccia grafica da cui si può monitorare con comodità alcuni parametri importanti tra i quali: numero di peers a cui si è connessi con relativa velocità di connessione e stato di sincronizzazione della

[1]Testnet: ogni blockchain possiede una rete di test, dove i neofiti possono prendere dimestichezza con essa e gli sviluppatori possono provare i loro smartcontracts. Ovviamente tutto ciò senza spendere veramente dei soldi, in quanto le monete che vengono utilizzate nella rete di test non hanno valore ed è possibile riceverne gratuitamente tramite alcuni portali web semplicemente inserendo il proprio indirizzo.

blockchain. Sempre dalla propria interfaccia grafica è ovviamente anche possibile creare wallet e gestirli.

La parte che potrebbe risultare più complessa per un neofita è quella che riguarda la corretta configurazione del file "bitcoin.conf", successivamente vedremo nel dettaglio come per far comunicare correttamente Bitcoin Core con il nostro LN si dovrà andare a mettere mano a questo file. Al momento è importante solo considerare l'opzione "testnet": settata a 0 se si vuole operare nella rete test, viceversa ad 1 per la main.
Per avviare il programma è necessario entrare nella cartelle di bitcoin core (precedentemente scaricata da internet) e avviare il programma (rispettivamente in modalità GUI o CLI) con i comandi ./bitcoin-qt o ./bitcoind. A questi comandi è possibile passare gli stessi parametri impostabili nel file bitcoin.config, in più è possibile anche settare la cosiddetta "datafolder" tramite il parametro "–datafolder=/path/to/datafolder". Questo parametro specifica al programma la cartella in cui dovrà andare a cercare e/o salvare alcuni dati utili all'esecuzione, quali ad esempio i file di log, il file bitcoin.conf e anche i blocchi della blockchain.
Avviato il programma (in modalità GUI o CLI) la blockchain comincerà ad aggiornarsi e ci vorrà un po di tempo perché questo processo si concluda, tuttavia se si ha già a disposizione una copia della blockchain è possibile copiarla all' interno della cartella <datafolder>/blocks per velocizzare (nel caso si stesse lavorando in testnet: <datafolder>/testnet3/blocks).
A questo punto con il nodo bitcoin funzionante si è già in grado di effettuare tutte le operazioni possibili sulla chain di Bitcoin come creare un wallet, effettuare pagamenti e riceverli.

2.2.3 Configurazione LND

Esistono diverse implementazioni di nodo Lightning Network e ciascuna di queste risponde al cosiddetto standard BOLT (Basis Of Lightning Technology). Le principali che ho preso in considerazione sono:

- Eclair: sviluppato in linguaggio Scala da ACINQ, la stessa azienda fornisce anche un comodo wallet mobile per interfacciarsi alla rete lightning (non necessita di scaricare la blockchain sullo smartphone)

- C-Lightning: scritto in C, il suo sviluppatore principale è Rusty Russel, famoso per il suo contributo al kernel Linux. L'utilizzo di un linguaggio così conosciuto e l'esperienza alle spalle di Russel garantiscono un' alta affidabilità. Consente inoltre di estendere il proprio nodo con plugins di terze parti.

- LND: sviluppato da Lightning Labs nel linguaggio GO, di gran lunga il più utilizzato e di conseguenza anche quello con più documentazione. E' inoltre l'unico che consente al momento la creazione di canali verso se stessi.
- Electrum: Electrum è in realtà un famoso client Bitcoin, che consente di gestire il proprio wallet e le relative transazioni on chain, tuttavia da Luglio 2020 è stata integrata un'implementazione di wallet lightning. Così facendo è possibile ottenere un nodo Bitcoin e un nodo LN con un singolo software (che presenta anche una discreta interfaccia grafica).

Inizialmente la scelta è stata quella di utilizzare Eclair in quanto è una delle implementazioni più longeve (nonostante questa sia comunque una tecnologia agli albori), tuttavia una volta configurato mi sono scontrato per la prima volta con il problema pratico dell outbound capacity: per avere dei canali efficienti è necessario effettuare il ribilanciamento di essi. Eclair ad oggi non consente il ribilanciamento in quanto non permette di effettuare pagamenti diretti a se stessi.
Electrum è stato scartato per la stessa ragione.
Per quanto riguarda C-lightning, la ritengo un' ottima opzione per i motivi già citati, tuttavia anch'esso non fornisce la possibilità di aprire canali verso se stessi. Se è vero che questa lacuna potrebbe essere colmata da un eventuale plugin è anche vero che questo sarebbe di terze parti e ciò metterebbe comunque in discussione l'affidabilità.
Ciò mi ha inevitabilmente portato a utilizzare l'implementazione di Lightning Labs.

La comunicazione tra nodo Bitcoin e lodo Lightning avviene mediante protocollo RPC [2]. Per fare si che ciò accada bisogna configurare correttamente il nodo bitcoin e il nodo lightning, entrambi mediante i propri file di configurazione (su Ubuntu 20.04.3 di default si trovano rispettivamente in /.bitcoin/data/bitcoin.conf e /.lnd/lnd.conf).

I files di configurazione

Principali opzioni nel bitcoin.conf:

- testnet =1
- server: se settata ad "1" il nodo bitcoin si mette in ascolto per ricevere richieste RPC, di default settata a "0"
- rpcuser e rpcpassword: credenziali che verranno richieste al client rpc, dovranno essere impostate uguali sul file di configurazione di LND

[2]per saperne di più: `https://www.ibm.com/docs/en/aix/7.2?topic=concepts-remote-procedure-call`

- txindex: questo valore se settato ad “1” crea degli indici per poter effettuare ricerche più efficienti nella blockchain, questa funzione è necessaria se vogliamo che un nodo lightning si appoggi sul nostro nodo bitcoin.
- txindex: questo valore se settato ad “1” crea degli indici per poter effettuare ricerche più efficienti nella blockchain, questa funzione è necessaria se vogliamo che un nodo lightning si appoggi sul nostro nodo bitcoin.
- rpcport=8332 e rpcconnect='127.0.0.1': indirizzo e porta su cui si metterà in ascolto il servizio rpc. Settando l'indirizzo di loopback ho la garanzia che il servizio sia visibile solo dalla macchina su cui è hostato, non consentendo a alcun tentativo di connessione dall'esterno

Principali opzioni nell' lnd.conf:

- datadir: cartella in cui LND salverà alcuni file relativi alla sua esecuzione, tra cui anche i file di log.
- bitcoin.active: se settato a true indica che il nodo lightning è destinato ad essere usato con la rete bitcoin, in alternativa potrebbe essere settato con litecoin.active=true per interfacciarsi con la rete Litecoin.
- bitcoin.testnet: anche in questo file di configurazione bisogna indicare se si vuole utilizzare la rete test o meno, in caso affermativo va settata questa opzione a “true”
- bitcoin.node: indica l'implementazione di nodo Bitcoin a cui si deve interfacciare LND, nel nostro caso “bitcoind”, che è il demone di Bitcoin Core
- bitcoind.rpcuser e bitcoind.rpcpass: anche qui vanno indicate le credenziali per la connessione al servizio RPC
- restlisten=127.0.0.1:8080: indica l'indirizzo e la porta su cui verrà aperto un servizio di API rest; questo ci sarà utile per la comunicazione con l'interfaccia grafica di cui parlerò in seguito, essa infatti comunicherà con il servizio di LND tramite queste API.
- alias: tramite quest'opzione possiamo impostare il nome con cui sarà visualizzato il nodo dall' esterno, per esempio sugli explorer.
- listen : con questa opzione si va a indicare a LND che dovrà mettersi in ascolto di richieste da altri nodi su una determinata interfaccia
- externalip=0.0.0.0:9735 : quest'opzione indica su quale indirizzo e porta sarà raggiungibile il nostro nodo sulla rete, settando l'IP con “0.0.0.0” garantiamo la visibilità su tutte le interfacce IP della macchina.

2.2.4 L'interfaccia grafica

Per avere un'interfaccia User friendly ho usato Ride-The-Lightning (abbreviato RTL, che prende in prestito il nome dalla canzone dei Metallica): RTL ci consente di effettuare (quasi) tutte le operazioni che normalmente si effettuerebbero da riga di comando tramite con un' interfaccia web dal nostro browser. Questa applicazione sfrutta node.js ed è completamente open source (https://github.com/Ride-The-Lightning/RTL). Comunica con LND grazie alle API che quest' ultimo mette a disposizione (vedi file configurazione). Con RTL possiamo gestire sia il nostro wallet on chain che i nostri canali su layer 2 con un' interfaccia intuitiva e senza quindi dover avere dimestichezza con la riga di comando. Infatti, diversamente l'utente dovrebbe effettuare tutte le operazioni dal terminale.
Anche questo software possiede un file di configurazione (RTL-config.json) e vale la pena analizzare alcune voci:

- port : la porta su cui si potrà accedere al servizio offerto da RTL (quindi alla GUI)

- host: l'url su cui verrà hostato il servizio, nel nostro caso si è scelto di hostare il servizio su localhost, in modo da garantire l'accesso alla sola macchina ospitante e non al resto della rete locale (http://127.0.0.1). Se si volesse offrire il servizio ad altre macchine è possibile utilizzare l'interfaccia visibile dall' esterno (es: http://192.168.1.2) tenendo in considerazione le problematiche riguardanti la sicurezza.

- multiPass : campo contenente la password per accedere al servizio, di default può essere settata "password", così facendo al primo accesso verrà richiesto di cambiare la password. Una volta fatto ciò questo campo verrà rimosso automaticamente dal file di configurazione e verra sostituito dal campo "multiPassHashed".

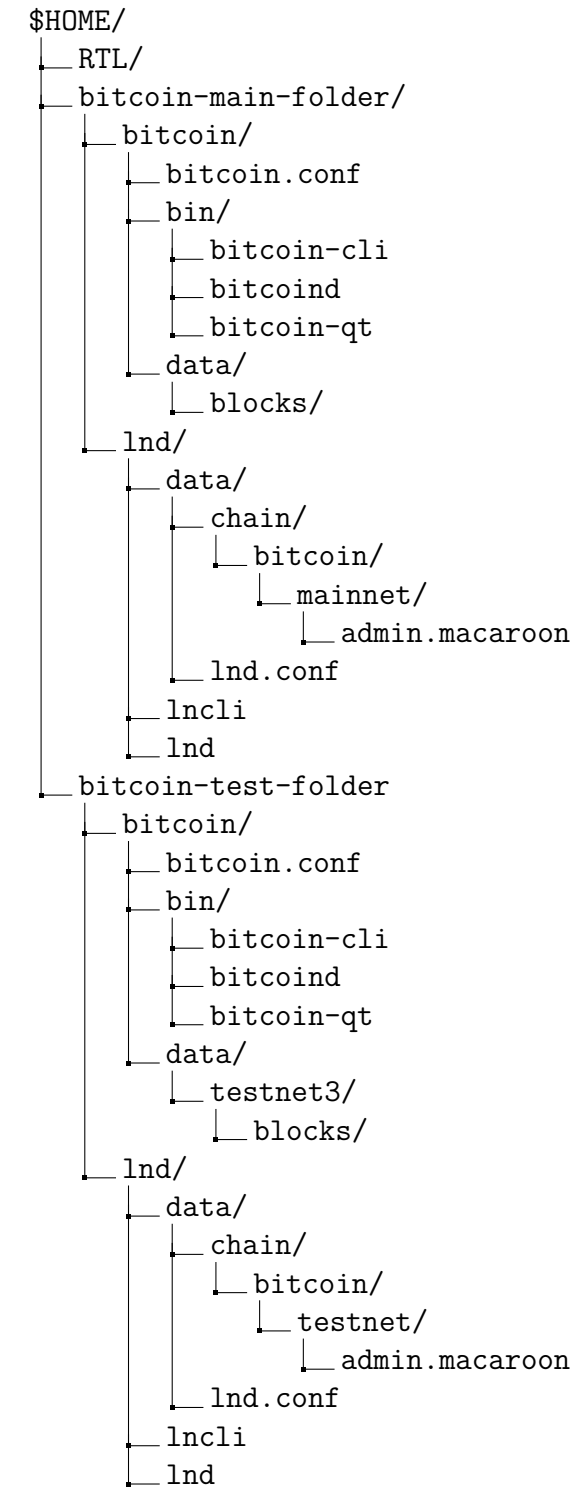
- multiPassHashed : l'hash della password richiesta prima di accedere alla pagina web hostata da RTL. NB: questo sistema espone a rischi molto elevati in quanto chiunque abbia accesso alla macchina può cancellare la voce "multiPassHashed" e sostituirla con 'multipass', assegnando un valore noto a quest'ultimo campo può accedere senza problemi ai servizii della GUI. è bene quindi garantire l'accesso al file di configurazione ai soli utenti autorizzati.

- nodes : questa voce è in realtà è un array e può contenere più oggetti 'nodo' al suo interno, questo perché è possibile gestire più istanze Lightning Node con una sola istanza di RTL. Ogni oggetto rappresentante una diversa istanza Lightning Node è strutturato nel modo seguente:

- index : l'indice univoco nella lista delle istanze l.n., se, come nel nostro caso ne abbiamo una sola allora avrà valore 1. Diversamente può essere utile averne due diverse: una per la testnet ed una per la mainnet (quindi 1 e 2).
- lnNode : un nome per riconoscere a quale istanza l.n. ci riferiamo.
- lnImplementation : l'implementazione di l.n. che abbiamo usato, nel nostro caso LND (ricordo che diversamente RTL può essere compatibile con altre implementazioni come Eclair o C-lightning, tuttavia si è scelta LND perché quella con il maggior numero di funzioni disponibili).
- Authentication : questo campo ospita a sua volta un oggetto contenente i dettagli per l'autenticazione di RTL con l'implementazione di l.n. Nel caso di LND per l'autenticazione si hanno 2 campi principali
  - macaroonPath : il percorso alla cartella contenente i "macaroon files" ovvero dei file contenenti i dati per autenticarsi a LND, molto simili a dei cookies come concetto.
  - configPath : il percorso al file di configurazione LND, di cui abbiamo già parlato.
- Settings : anche questo campo è composto da un' oggetto che contiene opzioni generali. Oltre a quelle relative al colore/tema dell' interfaccia grafica e al livello dei log, è bene citare:
  - lnServerUrl : url per raggiungere le api esposte da lnd
  - fiatConversion : valore booleano che indica la preferenza a convertire i satoshi/bitcoin in dollari.

2.2.5 Struttura e workspace

Per gestire questo progetto ho utilizzato 2 workspace paralleli: uno per la rete test ed uno per la rete main. Ciò significa che ho installato e configurato 2 nodi bitcoin e 2 nodi LND in parallelo in modo da avere una coppia configurata in main-net ed una in test-net. NB: questa pratica non sarebbe stata necessaria in quanto sarebbe bastata una sola coppia di istanze e andare a modificare di volta in volta i files di configurazione per passare da test-net a main-net, tuttavia, ho preferito dividere le due implementazioni per non rischiare di intaccare la configurazione main-net talora fossi andato a fare cambiamenti su quella test-net.
Per quanto riguarda il software RTL, è possibile gestire più istanze di LN contemporaneamente (vedi file di configurazione) e perciò è sufficiente una sola installazione. Per rendere meglio l'idea riporto la struttura delle cartelle del filesystem:


$HOME/
RTL/
bitcoin-main-folder/
bitcoin/
bitcoin.conf
bin/
bitcoin-cli
bitcoind
bitcoin-qt
data/
blocks/
lnd/
data/
chain/
bitcoin/
mainnet/
admin.macaroon
lnd.conf
lncli
lnd
bitcoin-test-folder
bitcoin/
bitcoin.conf
bin/
bitcoin-cli
bitcoind
bitcoin-qt
data/
testnet3/
blocks/
lnd/
data/
chain/
bitcoin/
testnet/
admin.macaroon
lnd.conf
lncli
lnd

E' possibile avviare il demone di btc andando nella cartella in cui è stato installato bitcoin e lanciando "bitcoind". A questo comando è possibile allegare il parametro "-conf" al fine di specificare il percorso in cui è situato il file di configurazione e il parametro "-datadir" per specificare la cartella in cui verranno salvati alcuni dati tra cui la blockchain. in caso contrario cercherà il file bitcoin.conf nel percorso attuale mentre verrà utilizzato di default il percorso "$HOME/.bitcoin/data" (creato se non esistente) per i dati. Nel mio caso, avendo due implementazioni di nodo bitcoin, devo assegnare al parametro "-datadir" una cartella diversa per ognuna in quanto quella di default verrebbe condivisa da entrambe. Oltre al demone bitcoin-core fornisce pure la versione con l'interfaccia grafica eseguibile con il comando "bitcoin-qt" che tuttavia noi non useremo. Avviatosi, il demone comincerà a sincronizzare la blockchain di bitcoin, il che potrebbe richiedere parecchio tempo nel caso della main-net.

Una volta sincronizzata la blockchain Bitcoin, è necessario avviare il demone LND dalla sua cartella di installazione. Come per Bitcoind è necessario passare all'eseguibile il parametro "–configfile" relativo al file di configurazione, di default "$HOME/.lnd/data/lnd.conf" (creato se non esistente). Anche qui necessario per evitare che main e test facciano riferimento allo stesso file di configurazione. Il demone dopo essersi avviato richiederà di sbloccare o creare un wallet. E' possibile farlo tramite l'eseguibile "lncli" presente nella stessa cartella eseguendo il comando "lncli unlock" o "lncli create".

2.2.6 Il wallet

Il wallet Bitcoin, a differenza di un portafoglio fisico, non contiene direttamente il denaro. Come concetto è più vicino ad un portachiavi, infatti è il contenitore delle nostre chiavi private, le quali ci consentono di utilizzare il denaro associato ad un determinato indirizzo sulla blockchain. Il denaro è invece "contenuto" sulla blockchain e può essere trasferito solo possedendone le chiavi. Possiamo distinguere 2 tipi principali di wallet blockchain:

- wallet non deterministici: in questi wallet ogni chiave viene generata in modo indipendente. Ciò significa che per effettuare il backup del wallet è necessario copiare ogni singola chiave privata.

- wallet deterministici: in questi wallet ogni chiave viene derivata da un "seed" che generalmente si presenta come una sequenza casuale di parole derivate da un dizionario predefinito. In questo modo basta effettuare il backup del singolo seed e non di tutte le chiavi.
  Dal seed viene generata una master-key e successivamente da quest'ultima viene ricavato un albero gerarchico di chiavi private sfruttando varie funzioni di derivazione.

Ormai i wallet sono per la stragrande maggioranza deterministici e Bitcoin ha adottato vari standard con il tentativo di unificare le varie specifiche per la gestione dei wallet, tra cui:

- BIP-32: specifica la struttura degli alberi gerarchici di chiavi e come vengono generati partendo da un seed di 128-512.
- BIP-39: specifica come derivare il seed partendo da una sequenza di parole, cosi da renderne più semplice la gestione da parte di esseri umani. Specifica inoltre il dizionario delle parole che è strutturato in modo che sia sufficiente inserire le prime 4 lettere per distinguere univocamente ogni vocabolo, sono state inoltre evitate le parole simili come "women" e "woman". Tutto ciò per evitare tediosi errori di inserimento/copia.
- BIP-44: definisce una logica per la struttura gerarchica degli alberi.

Tutte le specifiche elencate fanno parte dei BIP (Bitcoin Improvement Proposal), ovvero una collezione di proposte selezionate secondo un preciso iter dalla comunità di Bitcoin al fine di standardizzare e migliorare determinate funzioni, per saperne di più è possibile consultare il BIP-1 [Taaki, 2011], in cui viene definito BIP stesso. Non è tuttavia strettamente necessario adottare gli standard BIP, basta pensare ai wallet non deterministici, alla fine quello che conta è avere una chiave privata associata ad un indirizzo valido.

Nel nostro caso il wallet può essere gestito sia da Bitcoin-core che da LND, noi lo gestiremo con LND in modo da poter direttamente interfacciarci con la lightning network.

AEZEED

Il wallet di LND non adotta a pieno gli standard BIP, infatti anziché BIP-39 per la derivazione del seed dalla lista di parole utilizza lo standard AEZEED, che con BIP-39 condivide il dizionario di parole ma ha sostanziali differenze:
Mentre nel BIP-39 qualora si voglia utilizzare una passphrase per rendere più sicuro il seed, questa viene inserita prima della sua generazione:

$$BIP-39\_seed = HMAC-SHA512(random\_value, passphrase) \tag{2.1}$$

In questo modo il seed sarà derivato in funzione di un valore casuale e della passphrase ma questo procedimento risulta essere a senso unico, in quanto $HMAC-SHA512()$ non è invertibile. Al contrario AEZEED concatena il seed di derivazione ad alcuni byte di informazione e cifra poi il tutto con una passphrase ottenendo lo mnemonico da 24 parole. Queste non sono quindi direttamente utilizzabili per generare le chiavi ma devono essere prima decifrate conoscendo la

passphrase:

$$AEZEED\_24words = enc((INFO\_BYTES||16\_BYTES\_SEED), passphrase) \tag{2.2}$$

Con questo tipo di struttura è possibile modificare la passphrase successivamente.
NB: qualora non si volesse utilizzare una passphrase verrà utilizzata automaticamente la stringa "aezeed".
Come già accennato il seed viene concatenato ad alcuni bytes di informazione, che nello specifico sono:

$$INFO\_BYTES = 1B : version||2B : birthday\_timestamp \tag{2.3}$$

- version: indica la versione di AEZEED utilizzata
- birthday_timestamp: indica la distanza temporale del momento di creazione del wallet al blocco di genesi di Bitcoin. In questo modo è possibile sapere quando fermarsi durante la ricerca delle transazioni effettuate da un portafoglio appena ripristinato (funzione non presente in BIP-39).

NB: LightningLabs riporta che sarebbe in sviluppo un BIP relativo ad AZEED [Lightning Engineering, 2021]

Configurazione wallet su LND

Siccome è la prima volta che viene avviato LND si suppone che ancora non si abbia un wallet configurato, perciò è necessario crearne uno tramite "lncli create" che avvierà una procedura guidata per la creazione (o importazione, in quanto è anche possibile utilizzare un wallet esistente inserendo un seed compatibile AZEED) del wallet.
Supponendo di non avere nessun wallet da importare, la procedura è molto breve, e durante la quale viene richiesto di inserire due parole chiave che per comodità chiameremo unlock_pass e passphrase.
unlock_pass: è la password che utilizza LND per sbloccare il wallet. Ogni volta che il demone viene avviato, prima di accedere alle funzioni del wallet, verrà richiesto di sbloccarlo.
passphrase: è la password che, come abbiamo visto in precedenza, AEZEED utilizza per cifrare il seed. E' possibile ignorare la richiesta di questa password, verrà settata di default la stringa "aezeed" come da protocollo.
Al termine della procedura verranno stampate a video le 24 parole corrispondenti al seed cifrato AEZEED.

Se volessimo importare un wallet basterà rispondere positivamente alla domanda posta dal terminale durante la procedura e così facendo avremo la possibilità di inserire le 24 parole e successivamente la passphrase (o, nel caso non sia stato cifrato, la passphrase di default).
Eseguito correttamente il comando "lncli create" il wallet risulta sbloccabile e utilizzabile e da qui in avanti grazie a RTL ci si può dimenticare della linea di comando, ci torneremo in caso sia necessario effettuare ribilanciamenti di canali con autopilot.

2.2.7 Esposizione del nodo al di fuori della rete aziendale

Con la configurazione attuale il nodo lightning network è in grado di aprire canali verso l'esterno, ad ogni modo, non è pubblicamente visibile e ciò rende impossibile che i nodi esterni prendano l'iniziativa creando un canale nella nostra direzione. Questa proprietà è fondamentale se si vuole che il nodo abbia una funzione di instradamento, infatti creando canali verso di noi gli altri nodi ci offriranno della inbound capacity, fondamentale per poter far transitare le transazioni attraverso il nostro nodo.
L'unico altro modo per avere della inbound capacity senza che mai nessuno abbia creato un canale verso di noi sarebbe aver effettuato dei pagamenti, così facendo si sposterebbe parte della liquidità dall' altra sponda del canale. E' evidente che questa non è la via ottimale: non è detto infatti che ci sia la necessità di fare pagamenti non appena si sia aperto il nodo.
E' possibile esporre il nodo verso l'esterno tramite due strategie: facendo port forwarding sul gateway oppure tramite TOR.
La soluzione adottata in questo caso è stata la prima in quanto l'azienda possiede un IP pubblico statico (anche se non strettamente necessario, LND è in grado di rilevare il cambiamento dell'IP ed aggiornare gli altri peer del cambiamento) e non avrebbe avuto senso nascondere il nodo mediante l'anonimato che offre TOR se l'obiettivo è quello di rendere il nodo il più popolare possibile.
Non è però solo una questione di anonimato: un nodo esposto tramite tor (e quindi con dominio .onion) è visibile solo da altri nodi TOR, mentre un nodo esposto sulla rete tradizionale è visibile da entrambe le tipologie di nodo (ma può solo vedere nodi tradizionali).
C'è inoltre da considerare il discorso dell' affidabilità: la rete TOR è intrinsecamente più articolata e rende la comunicazione più lenta e delicata, un nodo di instradamento ad hoc deve offrire un servizio stabile ed efficiente.
Al netto di ciò la scelta del port forwarding quindi per un nodo di un attività commerciale con la potenziale ambizione di promuovere la creazione di nodi verso di esso risulta essere la scelta più azzeccata. Nel caso di un piccolo nodo privato però la scelta di TOR può rivelarsi molto pratica in quanto non richiede di aprire

una porta sul router domestico, il che comunque non è un' operazione immediata per un non esperto e in certi casi potrebbe addirittura non essere possibile se non contattando l'internet provider.
Di per sé anche configurare TOR per un utente non esperto potrebbe risultare impegnativo, tuttavia sono presenti numerose guide dove viene descritto passo-passo ogni singolo step[3].
Per dovere di cronaca è anche attuabile una configurazione ibrida, così facendo il nodo sarà raggiungibile da chiunque mediante la rete tradizionale e potrà raggiungere ogni tipo di nodo (utilizzando un proxy tor per connettersi ai nodi di quel tipo).

2.2.8 La gestione remota

Il sistema per come lo è stato descritto fino ad ora permette il controllo del nodo lightning solamente sulla macchina su cui è hostato, sarebbe utile poter accedere all'interfaccia di RTL anche da un altro dispositivo all'interno della rete aziendale così da poter gestire i canali e/o effettuare pagamenti con più comodità. Sebbene RTL sia dotato di un sistema di autenticazione mediante password la comunicazione mediante http avviene in chiaro e la parola chiave potrebbe essere letta da uno sniffer (anche se questo implica che lo sniffer abbia accesso alla rete aziendale).

Come si può vedere (Figura 2.2) l'autenticazione è stata sniffata nella rete interna con il software wireshark: il sistema è molto banale, viene passato al server il risultato della password hashata con l'algoritmo hash256. Nel nostro caso la password è "password2" ed è stato facile risalire al valore di partenza (Figura 2.3). Un eventuale attaccante non dovrebbe nemmeno disturbarsi a risalire alla password in chiaro, gli basterebbe sniffare il processo di autenticazione come mostrato nell'immagine (Figura 2.2) per ottenere il valore hashato, il quale può poi essere copiato in una successiva richiesta di autenticazione.

Bisogna perciò fornire un livello di protezione alla comunicazione in modo che l'autenticazione sul portale di RTL e anche la successiva comunicazione di dati avvengano in modo confidenziale. Per fare ciò ho individuato 3 maniere principali: tunneling SSH,TLS o TOR

Tunneling SSH

Questa soluzione il protocollo sfrutta il protocollo SSH in modalità "local tunneling". SSH è un protocollo che permette di creare un canale sicuro con mutua autenticazione, confidenzialità ed integrità dei dati. E' spesso usato per amministrare sistemi a distanza ed io stesso l'ho utilizzato talvolta per lavorare sui computer

[3]https://docs.lightning.engineering/lightning-network-tools/lnd/quick-tor-setup

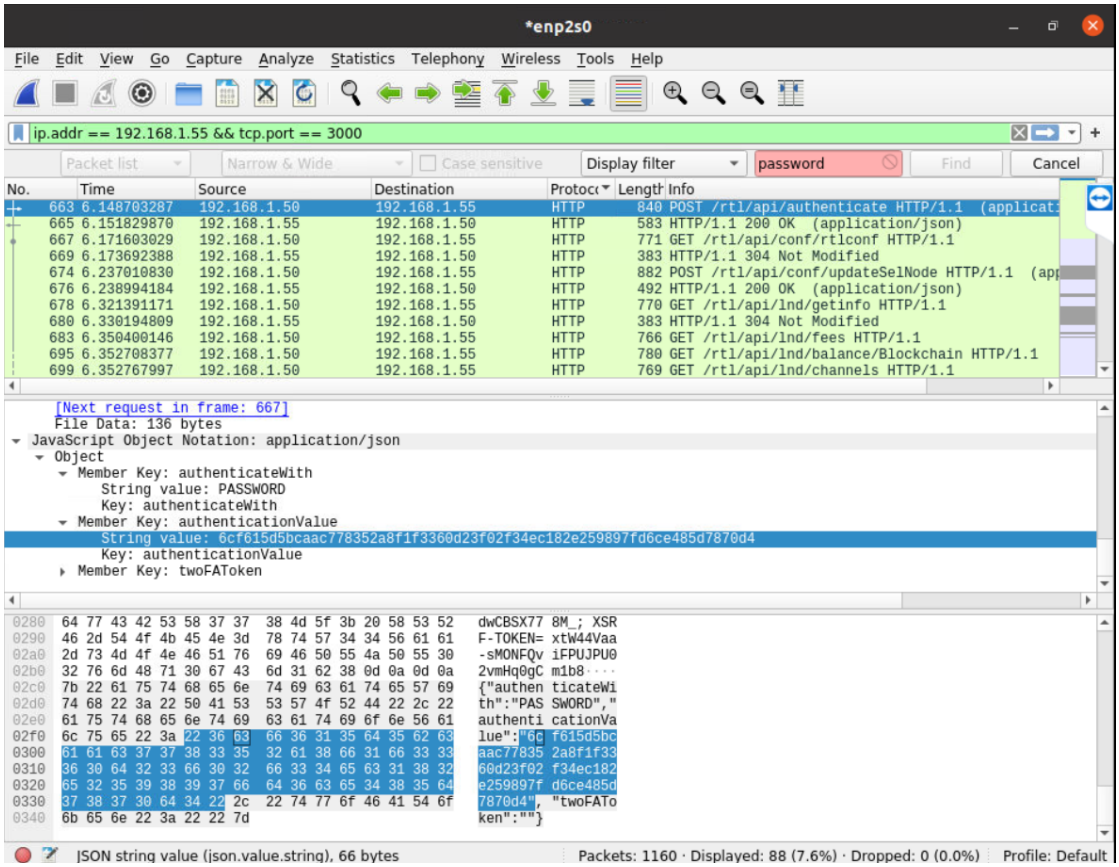


Figura 2.2: hash256 della parola chiave passato in chiaro al server, da questa foto si può inoltre evincere come anche dopo l'autenticazione tutto lo scambio dei dati avvenga in chiaro

dell' azienda direttamente dal mio portatile. Oltre che il semplice controllo remoto consente anche di operare in modalità "local tunneling" e "remote tunneling".
local tunneling: permette di inoltrare il traffico diretto verso una specifica porta del client (macchina locale) ad una specifica porta della macchina server sfruttando il canale SSH, In questo modo un servizio hostato dalla macchina server può essere esposto alla macchina client.
remote tunneling permette di inoltrare il traffico diretto verso una specifica porta della macchina server ad una specifica porta della macchina client sfruttando il canale SSH, in questo modo un servizio hostato dalla macchina client può essere esposto dalla macchina server.

Nel caso in esame è possibile sfruttare il local tunneling in modo che, aprendo un canale SSH con questa modalità sull' interfaccia di loopback del client, sia esposto il servizio hostato dal server, ovvero l'interfaccia RTL. Così facendo la comunicazione

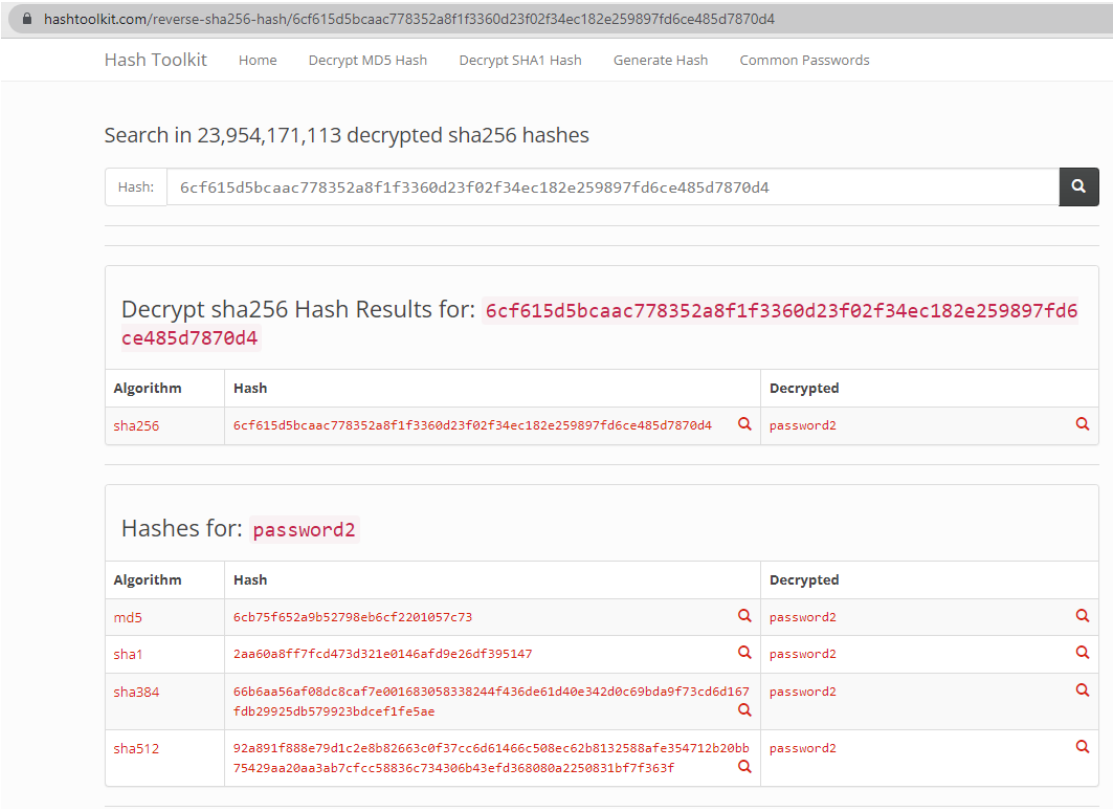


Figura 2.3: questa foto dimostra che quello passato dal client al server è effettivamente il valore della password hashato con algoritmo sha256

tra client e server SSH rimane protetta ed esponendo il servizio sull' interfaccia di loopback del client quest'ultimo è solo visibile al client stesso. Questa strategia risulta però essere più macchinosa e difficilmente attuabile su dispositivi mobili come smartphone e tablet, tuttavia è molto comoda se la macchina client è PC con sistema operativo unix like che in molti casi integrano già un software SSH client. Lo stesso vale per la macchina server in quanto la procedura per installare il server SSH e configurarlo su Ubuntu (in questo caso) è piuttosto immediata. E' possibile inoltre configurare SSH in modo che l'autenticazione avvenga mediante crittografia asimmetrica (più sicura della tradizionale password), tuttavia visto che si prevede che il server sia raggiunto da molteplici postazioni (il che richiederebbe la configurazione della chiave privata su ciascuna di queste) ho preferito mantenere l'accesso tramite password. Per rendere il tutto più robusto ho dotato il sistema di autenticazione a 2 fattori e di un contatore che dopo N tentativi errati di autenticazione blocca l'accesso ad un determinato IP.
Se si volesse controllare il nodo al di fuori della rete aziendale questa configurazione

è la più immediata, in quanto basterebbe impostare il port forwarding sul gateway per reindirizzare le connessioni su una determinata porta al nostro server SSH.

TLS

L'alternativa è sfruttare il protocollo TLS che garantisce integrità, confidenzialità e la possibilità di autenticare il solo server oppure entrambi i peer (nel nostro caso sfrutteremo solol'autenticazione del server, il client si autenticherà poi a livello di applicazione). E' possibile esporre il servizio con la protezione di TLS mediante un reverse proxy, nello specifico è stato utilizzato Nginx che va ad interporsi tra la rete locale e il servizio di RTL, quest'ultimo esposto sull'interfaccia di loopback al fine di non essere raggiungibile da host esterni. Questa soluzione è più impegnativa da mettere in piedi perché coinvolge i certificati x509 i quali necessitano in primis di essere digitalmente firmati da una certification authority.
Sono disponibili alcuni sistemi per automatizzare il processo di richiesta e di rinnovo dei certificati (come Let's Encrypt Certbot) ma in questo caso non risolverebbe il problema perché innanzitutto c'è bisogno di rendere sicura la trasmissione di dati sulla rete interna e, anche nel caso si volesse successivamente esporre un endpoint verso l'esterno, sarebbe necessario un nome di dominio.Per queste ragioni ritengo che la cosa migliore da fare sia utilizzare una Root certification Authority interna e utilizzarla per firmare il certificato associato all'IP privato della rete interna. Questa soluzione non necessità di alcuna operazione particolare sulla macchina client, se non quella di impostare la CA che abbiamo creato come affidabile sul browser. In caso si volesse accedere dal di fuori della rete locale si può comunque sempre utilizzare la soluzione con SSH.

Tor

E' possibile inoltre sfruttare TOR per ottenere un canale di comunicazione sicuro tra le parti ed avere anche in questo caso confidenzialità, integrità ed autenticazione del server. Questa soluzione permette di gestire il nodo sia dalla rete locale che dall' esterno, tuttavia ha due principali lati negativi:

- lentezza: come già detto, il protocollo tor per garantire l'anonimato fa rimbalzare i dati attraverso diversi nodi intermedi sparsi per il mondo, aumentando molto i tempi di latenza e limitando la banda.

- accessibilità: per accedere ad un servizio esposto sulla rete TOR è necessario avere un client in grado di farlo. sono disponibili diversi Browser che lo consentono (su PC, MAC e Android, mentre su App Store i client sono pochi e con

recensioni scarse, almeno al momento) , ma si tratta comunque di uno step in più e ciò non la rende quindi una soluzione più immediata rispetto alle altre due.

Questa è l'unica delle 3 opzioni che non è stata implementata in quanto al momento si ha come main goal quello di gestire il nodo dalla rete dell'ufficio e TOR sarebbe stato altamente inefficiente a questo scopo. Eventualmente si penserà ad implementarla per il controllo remoto qualora la strategia basata su SSH non sia soddisfacente.

2.3 Studio economico

Precedentemente ho già fatto riferimento al fatto che un nodo Lightning Network, qualora fosse utilizzato come intermediario per inoltrare un pagamento, ha diritto ad incassare una piccola tassa di transazione.

Vediamo ora nello specifico dal punto di vista economico se è possibile lucrare su un nodo LN.
Prima di tutto è bene precisare in cosa consiste questa tassa di transazione, la formula e la seguente:

$$tx\_fee = base\_fee + per\_sat\_fee * forwarded\_sats \tag{2.4}$$

Come si può notare, la tassazione comprende un valore fisso ed uno dinamico: la "base_fee" è un importo fisso che verrà imposto ad ogni transazione, mentre il valore per_sat_fee è un valore che verrà moltiplicato per i satoshi inoltrati.

2.3.1 stima dei costi

dimensionamento nodo

Ho cercato di simulare il guadagno di un nodo di medie dimensioni, prendendo i seguenti dati dal portale di `1ml.com` (Figura 2.4).

- 9.69 channel per node: Ogni nodo ha in media tra i 6 e i 7 canali aperti, il valore riportato è 6,9 e noi utilizzeremo quest'ultimo, nonostante non sia un numero intero, per ottenere una stima il più accurata possibile.

- 7040,09$ avg node capacity: Liquidità media bloccata su ciascun nodo
- 1406,00$ avg channel capacity: Liquidità media su ciascun canale (corrisponde a circa 0,22btc con il cambio attuale)

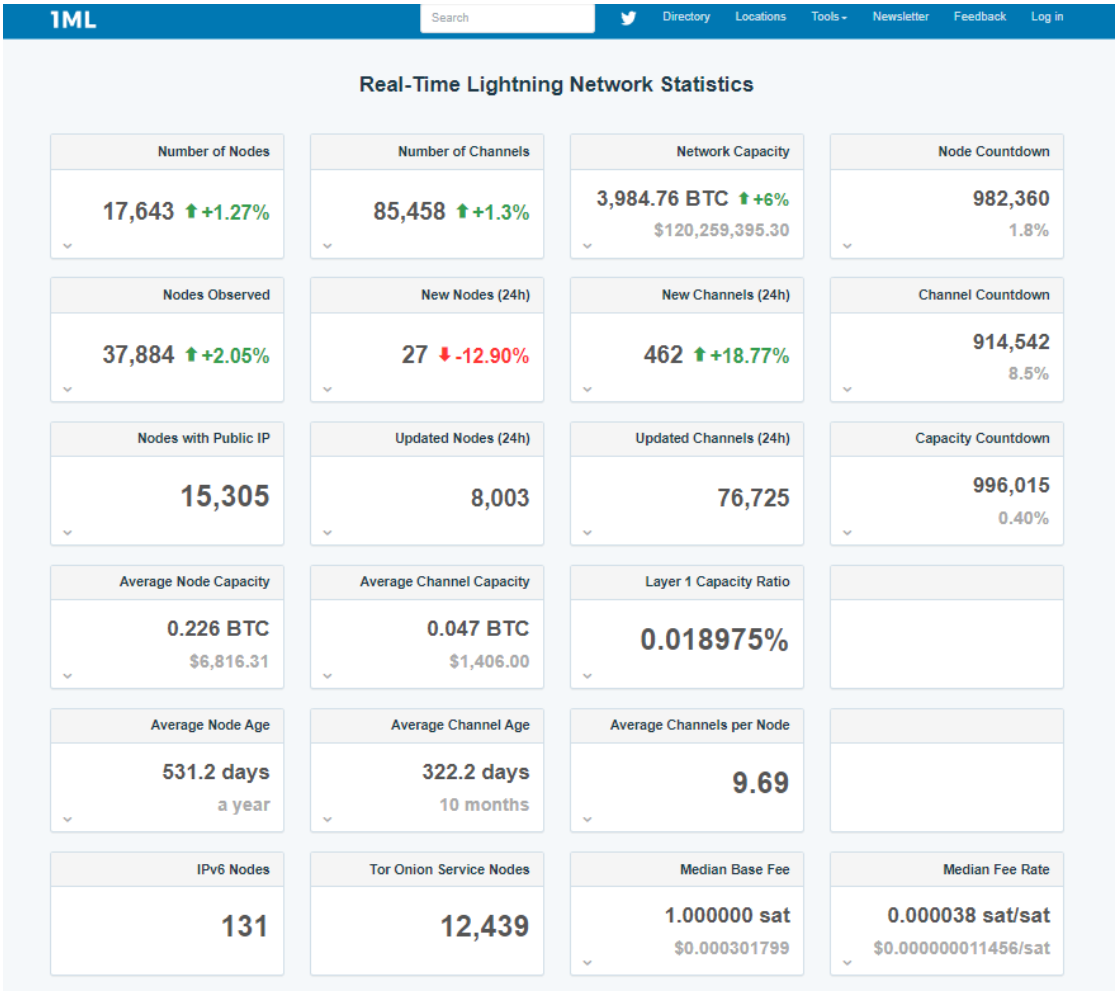


Figura 2.4: valori riportati sul portale di `1ml.com` in data 08/06/2022

- 1sat median base_fee: valore mediano della tassa fissa su ciascuna transazione
- 0,000038sat/sat median per_sat_fee: valore mediano della tassa per ogni satoshi inoltrato

Tasse di apertura e chiusura canale

per calcolare le spese di transazione relativi all'apertura e chiusura del canale mi sono basato sul grafico reperibile alla pagina `https://www.blockchain.com/charts/fees-usd-per-transaction` prendendo in considerazione i dati relativi agli ultimi 365 giorni (dataset scaricabile dal sito citato, io ho scaricato i dati il giorno 8/6/2022 che comprende i 365 giorni precedenti all' 8/6/2022).

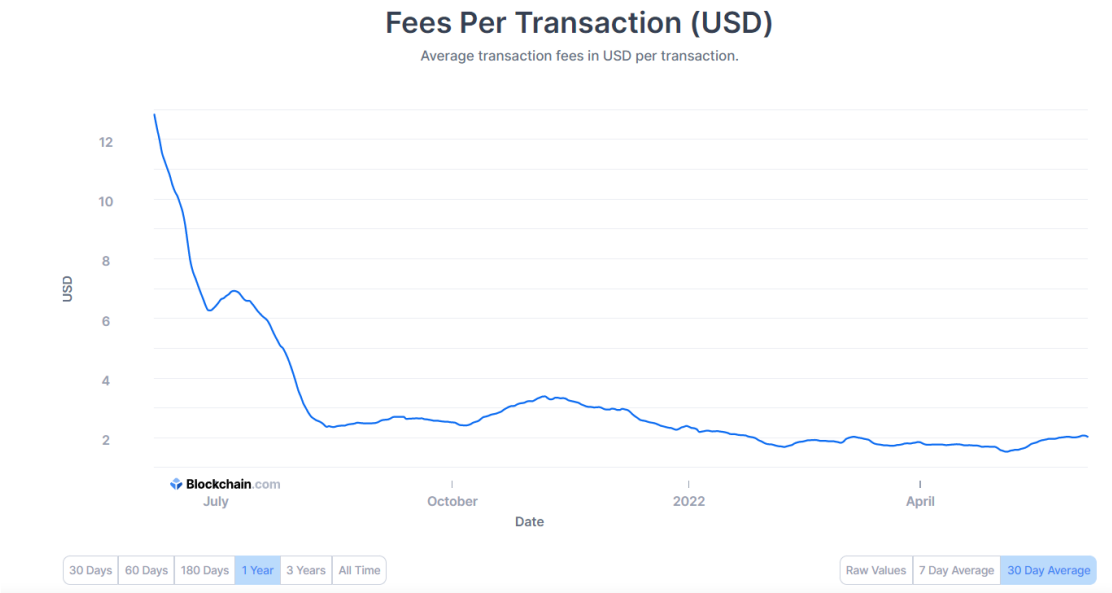


Figura 2.5: grafico riportato sul portale di `www.blockchain.com` in data 08/06/2022, si può notare come il boom crypto possa influire sul valore medio del costo delle transazioni.

Anzichè prendere in considerazione il valore medio (2,68$) che come vediamo dal grafico è fortemente influenzato dal boom crypto del 2021 ho preferito anche qui, trattandosi di un dataset piuttosto denso, prendere in considerazione la mediana (2,26$).

- 2,26$ median tx fee main net

Costo energetico

Per quanto riguarda la spesa energetica ho supposto un costo di 0,35$/kWh, mentre per il consumo energetico è stato preso in considerazione il consumo della macchina su cui è stato implementato il nodo, ovvero il Lenovo ThinkCentre Edge 72 che con il nodo operativo ha un consumo medio di 30W (misura effettuata con un misuratore di potenza)

2.3.2 Stima del traffico e guadagni

I valori più difficilmente stimabili sono quelli che riguardano il traffico di satoshis medio per ogni nodo/canale. In questo caso non sono disponibili fonti certe, in

quanto le transazioni che avvengono e i bilanciamenti tra i canali non sono informazioni pubbliche.
Al fine di effettuare una prima stima indicativa ho considerato che quotidianamente ci sia un traffico pari a circa il 20% [4] della capacità del canale.

Tutte le transazioni che considereremo in questo studio per semplicità avranno un valore di 30000sats, in questa casistica particolare quindi è possibile coprire il 20% della liquidità del canale con 31 transazioni.

cambio btc/sats	100.000.000
cambio usd/sats	0,000302
avg tx value (sats)	**30.000**
avg tx value (usd)	**9,05**
channel per node	9,69
daily avg tx per channel	**31**
daily traffic/investment rate	0,2
cambio eur/usd	1,04
costo energia €/kwh	0,35
costo energia $/kwh	0,36

Tabella 2.1: I valori evidenziati in grassetto sono quelli che modellano la quantità e il valore delle transazioni.

Come si può vedere dalle tabelle 2.3 e 2.4, i margini di guadagno sono piuttosto bassi, nella tabella 2.3 viene presa in analisi la differenza tra i guadagni e la spesa energetica, mentre nella tabella 2.4 viene tenuto conto anche della spesa di apertura e chiusura del canale. In quest'ultima tabella sono stati barrati i ricavi netti giornalieri e mensili in quanto essendo influenzati dalla spesa di apertura e chiusura non ha senso prenderli in considerazione per un periodo di tempo così ridotto.
Nonostante i bassi guadagni è interessante notare che in un anno verrebbero inoltrati più di 32btc, con un valore pari a quasi 1 milione di dollari al cambio attuale. Al netto si dovrebbe ancora sottrarre il costo di ribilanciamento dei canali in quanto questi ultimi, in base alla topologia della rete, tendono ad essere usati maggiormente in un verso piuttosto che nell'altro. Pertanto possiamo supporre che la frequenza di ribilanciamento dei canali sia correlata al numero di transazioni giornaliere. Tuttavia risulterebbe estremamente complesso stimare questi valori in quanto dipendono da variabili impredicibili quali la struttura della rete (in che modo sono collocati i nodi con maggiore affluenza) e l'ordine delle transazioni,

[4] Il valore 20% è approssimato al fine di ottenere un numero di transazioni intere

-	usd	sats	btc
avg tx val	9,05	30.000	0,00030
median tx fee mainnet	2,26	7.488	0,00007
opening+closing cost per-ch.	4,52	14.977	0,00015
tot opening+closing cost	43,80	145.126	0,00145
energy cost 1 day	0,04	145	0,00000
energy cost 30 days	1,31	4.342	0,00004
energy cost 1 year	15,94	52.827	0,00053
avg capacity channel	1.406,00	4.658.730	0,04659
avg node investiment	6.816,31	22.585.595	0,22586
daily traffic per channel	280,67	930.000	0,00930
tot daily traffic	2.719,72	9.011.700	0,09012
tot 30d traffic	81.591,66	270.351.000	2,70
tot 1y traffic	992.698,55	3.289.270.500	32,89
median base fee	-	1	-
median per-sats fee	-	0,00004	-
tot earn-per-day	0,20	679	0,00001
tot earn 30 days	6,15	20.366	0,00020
tot earn 1 year	74,78	247.792	0,00248
gross %year	1,10	1,10	1,10
rebalancing cost per-ch	?	?	?

Tabella 2.2: dati relativi ad un nodo che inoltra approssimativamente il 20% della sua liquidità

only gain-energy cost	1 day	30 days	1 year	% 1 year
$	0,16	4,84	58,84	0,86
€	0,16	4,65	56,58	0,86

Tabella 2.3

total	1 day	30 days	1 year	% 1 year
$	- ~~43,64~~	- ~~38,96~~	15,04	0,22
€	- ~~41,96~~	- ~~37,46~~	14,46	0,22

Tabella 2.4

perciò in questo studio non ho considerato i costi di ribilanciamento, supponendo che le transazioni siano perfettamente bilanciate in quanto a direzionalità.

Visto che la quantità di satoshis inoltrati quotidianamente sono stati stimati al

20% dell'investimento iniziale, ha senso cercare di variare quest'ultimo valore al fine di notare variazioni significative nel guadagno percentuale annuo. Come già detto per questo studio ipotizziamo transazioni da 30000sats, perciò l'unica opzione è quella di variare il "daily traffic/investment rate" (e di conseguenza il numero di transazioni) , mantenendo invece invariati i valori estratti dalle statistiche (vedi tabella 2.5).

daily t./i. rate	daily tpc	1Y traf. sats	1Y traf. btc	1Y traf. /	tot % 1Y
0,01	46587	164772286	1,6477	49.728,11	-0,82
0,02	93175	329544571	3,2954	99.456,22	-0,77
0,05	232936	823861428	8,2386	248.640,56	-0,60
0,1	465873	1647722855	16,4772	497.281,11	-0,33
0,199625228	*930000*	*3289270500*	*32,8927*	*992.698,55*	*0,22*
0,2	931746	3295445711	32,9545	994.562,22	0,22
0,3	1397619	4943168566	49,4317	1.491.843,33	0,77
0,4	1863492	6590891421	65,9089	1.989.124,44	1,32
0,5	2329365	8238614276	82,3861	2.486.405,55	1,87
0,6	2795238	9886337132	98,8634	2.983.686,66	2,42
0,7	3261111	11534059987	115,3406	3.480.967,77	2,97
0,8	3726984	13181782842	131,8178	3.978.248,88	3,52
0,9	4192857	14829505698	148,2951	4.475.529,99	4,07
1,0	4658730	16477228553	164,7723	4.972.811,10	4,62

Tabella 2.5: In *italico* è stata evidenziata la riga che fa riferimento alla stima fatta pocanzi, infatti per ottenere un numero intero di transazioni (31) il "daily traffic/investiment rate" è stato approssimato. E' stata quindi inserita la riga con il valore esteso per far combaciare i valori delle altre colonne con i valori mostrati nella tabella 2.2.
t./i. rate = traffic/investment rate
tpc = traffic per channel
traf. = traffic

2.3.3 Considerazioni finali

Dai dati ricavati risulta che anche ottenendo una media quotidiana del 100% dei satoshi inoltrati per canale si può arrivare dopo un anno ad incassare il 4,62% dell' investimento iniziale. Questa ipotesi è utopica tuttavia, si tratterebbe di inoltrare quasi 5 milioni di dollari (sempre facenti riferimento al cambio attuale) all'anno. Un nodo di piccole/medie dimensioni ha scarse probabilità di riuscire ad arrivare a quelle cifre, lo stesso patrimonio iniziale potrebbe essere investito diversamente per avere rendite ben più alte, come ad esempio su un exchange centralizzato.

Quindi aprire un nodo lightning network è economicamente svantaggioso? Per le realtà medio-piccole decisamente sì, ci sono poi colossi che sono riusciti a far fruttare questo business a fronte però di una grande disponibilità di liquidità.
I grandi nomi che operano in questo mondo da tempo godono di un grande bacino di nodi disposti ad aprire canali verso di loro, aumentando di conseguenza il loro appeal. Questi grandi hub hanno la facoltà di inoltrare un numero enorme di transazioni e ciò li rende gli unici veri aventi profitto.
Un ulteriore modello di business attuabile è quello del prestito di inbound capacity, dove si va ad offrire il servizio di apertura di un canale verso un nodo pagante, anche qui i nodi prestigiosi e ricchi risultano essere i più richiesti avendo molta liquidità e inbound capacity disponibile.
Questo scenario è fortemente influenzato dal bassissimo costo delle transazioni della lightning network che rende come abbiamo visto svantaggioso per le entità medio-piccole entrare nel giro generando così un circolo vizioso.
Al momento chi decide di aprire un canale lightning network lo fa principalmente con lo scopo di contribuire alla rete o perché ha esigenza di trasferire quotidianamente transazioni di piccolo o medio valore in bitcoin, ma non a scopo di lucro come invece può essere concepito il mining sulla rete layer-one.
Al netto di questi dati si è deciso di non implementare il nodo in main-net, almeno per il momento, provvedendo piuttosto a creare un canale con il nodo del gruppo studentesco BitPolito in modo da contribuire ai loro test.

Capitolo 3

Chia Network

3.1 Introduzione

I problemi di Bitcoin elencati in precedenza hanno spinto varie entità a studiare nuovi protocolli di consenso che riescano ad essere meno energivori della proof of work ma allo stesso tempo a garantire un adeguato livello di sicurezza.
Uno dei sistemi più recenti e interessanti è sicuramente la 'proof of space and time' adottata dalla blockchain Chia Network [Cohen and Pietrzak, 2019], la quale anziché la potenza computazionale sfrutta lo spazio di archiviazione. La probabilità di riuscire a validare un blocco è quindi proporzionale alla memoria su disco che viene assegnata al nodo. Ciò porta a due vantaggi principali dal punto di vista ambientale:
Prima di tutto l'energia necessaria per alimentare dei dischi di archiviazione è nettamente inferiore a quella richiesta dai miners di Bitcoin e quindi il sistema risulta molto meno dispendioso dal punto di vista energetico.
Il secondo vantaggio che porta questo sistema è che i dischi di archiviazione sfruttati dai nodi possono trovare nuova vita qualora non venissero più utilizzati per tale scopo. In più c'è da sottolineare che mentre nel caso della proof of work un hardware obsoleto veniva necessariamente sostituito da uno più efficiente per ottenere un rapporto hash/kWh più profittevole, con questa tecnologia si può semplicemente accostare l'hardware nuovo a quello vecchio dal momento che anche un miglioramento delle sue prestazioni (velocità lettura/scrittura) non impatterebbe sull'efficienza (mentre invece andrebbe ad aumentare la possibilità di validare un blocco visto che si aumenta lo spazio di archiviazione). Ciò, unito al fatto che le operazioni effettuate sui dischi di archiviazione non sono particolarmente usuranti (almeno per quanto riguarda i dischi rigidi), fa in modo che Chia non risenta del problema dei rifiuti digitali che affligge Bitcoin.
I 'miners' di Bitcoin vengono sostituiti dai 'farmers', chiamati cosi perché fanno

fruttare il loro spazio di archiviazione cosi come un contadino farebbe fruttare il suo campo coltivato (dando così un idea green del progetto accostando il protocollo di consenso alla coltivazione di piante).

3.2 Il protocollo

Il protocollo di Chia mira ad emulare la proof of work di Bitcoin sfruttando in maniera intelligente l'alternarsi di due concetti principali: quello proof of space e quello di Verifiable delay function (VDF).
I nodi della rete riempiono la memoria che decidono di dedicare alla blockchain con grossi file contenenti valori pseudorandomici denominati "plot". La fase di scrittura di questi file è denominata "plotting" e il tempo richiesto è strettamente dipendente dalla velocità del disco utilizzato e dal numero di core che possiede il processore.
I plot hanno una durata permanente e sono legati ad una chiave pubblica appartenente al nodo che li ha prodotti (è tuttavia possibile produrre plot per terzi inserendo la chiave pubblica di un altro nodo). Una volta prodotti vengono immagazzinati nella memoria di archiviazione e sono utilizzabili per eseguire la proof of space (farming).

3.2.1 Proof-of-space

La proof of space è il sistema che consente ad un farmer di dimostrare alla rete che ha destinato un certo spazio di archiviazione al servizio della blockchain di Chia. Ciò è reso possibile tramite un compromesso tra spazio di archiviazione e potenza computazionale.
L'idea di base è la seguente:
Si hanno 2 funzioni casuali (riconducibili a delle funzioni di hash):

$$f(id, x)_{|l} \tag{3.1}$$

$$g(id, x, x')_{|l} \tag{3.2}$$

dove con *id* si indica la chiave pubblica associata al plot (ovvero al blocco di dati) e con "$_{|l}$" si vuole indicare che il risultato della funzione è troncato ai primi l bit.
Il prover genera una prima tabella contenente le tuple

$$(x, f(x)) \tag{3.3}$$

dove x è un valore incrementale che va da 0 a $2^l - 1$.
Una volta generata questa tabella il prover deve cercare tutte le coppie di record in cui

$$x \neq x' \ \wedge \ f(x) = f(x') \tag{3.4}$$

Date queste combinazioni di valori x e x' viene generata una seconda tabella record strutturati nella maniera seguente:

$$(x, x', g(x, x')) \tag{3.5}$$

Successivamente questa tabella viene riordinata secondo $g(x, x')$ e viene salvata in memoria occupando uno spazio (statisticamente atteso, in quanto non si può sapere a priori il numero di combinazioni che rispettano 3.4) di $2^{l+1}l$ bits.
Dalla seconda tabella generata è quindi possibile, data una challenge y che è appartenente al medesimo dominio di $g(x, x')$ andare alla ricerca di un record che riesca a soddisfare le condizioni:

$$x \neq x' \ \wedge \ f(x) = f(x') \ \wedge \ y = g(x, x') \tag{3.6}$$

Il valore restituito dal farmer sarà la tupla $\sigma = (x, x')$, la quale è rapidamente verificabile in quanto al verificatore basterà rieseguire $f(x)$, $f(x')$ e $g(x, x')$ controllando poi il corretto soddisfacimento delle condizioni 3.6.

Questo procedimento nell'implementazione reale è annidato più volte e presenta alcune complessità aggiuntive (per tutti i dettagli è possibile consultare `https://www.chia.net/assets/proof_of_space.pdf`), tuttavia l'intento è illustrare come sia possibile per un nodo dimostrare di aver effettivamente occupato un determinato spazio di archiviazione rispondendo ad una challenge scorrelata dai plot posseduti. E' interessante evidenziare come un nodo potrebbe eventualmente prodursi il blocco di dati sul momento per risolvere la proof of space, evitando quindi di immagazzinare in memoria i plot. Tuttavia questo richiederebbe una potenza di calcolo spropositata, ciò rende sconveniente un approccio simile.

3.2.2 VDF

La verifiable delay function (VDF) è una funzione che permette di dimostrare l'effettiva esecuzione di un numero determinato di passaggi computazionali consecutivi. In base alla velocità dei processori odierni è possibile associare in maniera indicativa un periodo minimo necessario per l'esecuzione di queste funzioni.
La VDF garantisce questa proprietà di "attesa" in quanto è composta da un numero

N di passi consecutivi e non parallelizzabili. E' inoltre è necessario che la VDF sia verificabile in un tempo nettamente minore rispetto al tempo di generazione. Le implementazioni più recenti (tra cui quella utilizzata da Chia Network) si basano su un elevamento al quadrato all'interno di gruppi matematici (per maggiori informazioni riguardo i concetti matematici: [Long and Network, 2018].
Lo scopo delle VDF nella blockchain di Chia è proprio quello di garantire un delay tra la creazione di una proof of space da parte di un farmer e l'effettiva validazione del blocco. Se infatti su Bitcoin era sufficiente la sola proof of work per validare un blocco, in quanto di per sé il processo di mining richiede un tempo medio (autoregolato dal parametro della difficoltà come già visto), nessuno proibisce ad un eventuale farmer di produrre un numero indefinito di proof of space pressoché istantaneamente. Ciò permetterebbe ad un eventuale nodo malevolo di ricreare la storia della blockchain a suo piacimento e proporla poi agli altri membri della rete, Questo scenario è noto come "long range attack".
Per questo motivo entra in gioco nel protocollo la VDF: dopo che è stata prodotta una proof of space σ deve essere eseguita una VDF che prende come parametro $H(\sigma)$ noto come "qualità", dove $H()$ è una funzione di hash. La qualità di una proof of space non è altro che il numero di zeri iniziali del suo hash e la VDF che avrà come parametro $H(\sigma)$ dovrà eseguire un numero di passaggi inversamente proporzionale alla qualità del parametro. In altre parole, una VDF compiuta con un hash di una proof of space con una qualità più alta restituirà un risultato in tempi minori.

3.2.3 Il plotting

Il plotting è il processo che riguarda la scrittura del file, è la parte più onerosa dal punto di vista energetico, computazionale e anche in termini di tempo. Il risultato di questa operazione è un file comprendente 7 tabelle da cui si potrà poi effettuare la proof of space.
Il plotting è composto da 4 fasi e vedremo dopo nel dettaglio che ognuna di esse avrà un certo peso in termini di tempo e consumo energetico. Il compito che ognuna di esse svolge non è stato analizzato in questa tesi in quanto si è applicato un metodo puramente empirico per valutare la migliore strategia di plot.
Nel momento della creazione di un plot tra i tanti parametri (che verranno discussi in seguito) viene richiesto il valore K, questo valore indica la grandezza del file di plot.

K-size	Temp. Size	Final Size
K=25	1,8 GiB (1.9 GB)	600 MiB (629.1 MB)
K=32	239 GiB (256.6 GB)	101.4 GiB (108.9 GB)
K=33	512 GiB (550 GB)	208.8 GiB (224.2 GB)
K=34	1041 GiB (1118 GB)	429.8 GiB (461.5 GB)
K=35	2175 GiB (2335 GB)	884.1 GiB (949.3 GB)

Tabella 3.1: Tutti i possibili valori assegnabili a K con le rispettive dimensione su disco temp e dest (i tempi di plot sono conseguentemente proporzionali alla dimensione del file). NB: il valore K=25 è usato solo per scopi di test e l'algoritmo non tiene in considerazione i plot con K<32 per eseguire la proof of space. Sarebbe inoltre fattibile realizzare plot con k>35 ma i requisiti in termini di tempo e memoria comincerebbero ad essere estremamente alti.

Al momento non c'è motivo di produrre plot con k>32, tuttavia questo parametro permetterà in futuro di scalare e passare a plot di dimensione maggiore qualora ci fosse un evoluzione tecnologica.

I dischi coinvolti

Durante il plot vengono elaborati una grande mole di dati, molti più di quelli contenuti nel file finale. Tutte queste elaborazioni si appoggiano ad uno spazio di storage temporaneo (temp) che è possibile settare prima dell'inizio del plot. La cartella scelta per questo compito è bene che si trovi in un disco ad elevate prestazioni per il fatto che le operazioni su disco tendono a fare da collo di bottiglia all'intero processo.
Allo stesso modo è necessario settare lo spazio di memoria di "destinazione", ovvero dove andrà a collocarsi il file finito (dest). In questo caso non è strettamente necessario un disco ad alte prestazioni, tuttavia il processo di plot include una fase di copia dal disco temporaneo a quello finale, perciò una maggior velocità di scrittura potrebbe ridurre ulteriormente il tempo di plot.
Oltre al disco di destinazione e a quello temporaneo è possibile utilizzare un terzo disco noto come "temporaneo secondario" (temp2), andando ad impostare questo valore si indica dove il protocollo andrà a ricostruire il file di plot prima di riordinarlo (durante la fase 3 e 4), a questo punto il file dovrebbe essere grande circa il 110% del file finale. Se questa impostazione non viene settata allora verrà utilizzato la stessa cartella destinata allo spazio "temporaneo". Questa opzione può risultare vantaggiosa perché al termine della fase 4 il plot risulta terminato, perciò se si possiede un disco di "destinazione" abbastanza veloce allora si può valutare di settare temp2 = dest in modo che non sia necessaria la copia finale. Bisogna però considerare che nelle fasi 3 e 4 avvengono ancora numerose letture e scritture:

se il disco temp2(=dest) non è sufficientemente prestante il processo risulterebbe comunque più duraturo seppur senza la copia finale.

Le implementazioni

Allo stato attuale il software ufficiale di Chia Network supporta 3 implementazioni dell'algoritmo plotting:

- Chia proof of space: l'originale di chia network, con questa implementazione è possibile tramite il software principale avviare più plot in parallelo eventualmente facendoli partire ad intervalli regolari in modo da sfruttare la differente richiesta di risorse hardware tra una fase e l'altra.

- madMax: quest'implementazione è stata pubblicata su github nell'autunno 2021 ed inizialmente poteva essere eseguita solamente tramite un software a parte, presto però è stato inglobato nel nodo ufficiale di Chia.
  Punta a sfruttare al massimo i core della cpu in modo da parallelizzare il massimo numero di operazioni possibili durante un singolo plot. Non ha quindi senso plottare più plot in contemporanea in quanto l'obiettivo e concentrare tutte le risorse su un plot singolo che risulterà molto più veloce rispetto all'implementazione precedente

- Bladebit: quest'implementazione richiede una notevole quantità di ram dal momento che è progettato per eseguire tutte le operazioni utilizzando quest'ultima al posto che il disco. Grazie a questo riesce ad essere l'implementazione più efficiente sotto ogni punto di vista, tuttavia richiede un minimo di 416 GiB ($\sim$ 447 GB) di memoria ram. Dati i costi eccessivi necessari per raggiungere questo requisito (4 banchi di ram da 128 GB sarebbero venuti a costare oltre €4000), questo algoritmo non è stato preso in considerazione.

3.2.4 Il pooling

Chia Network implementa nativamente il farming su pool. Con il termine pool si indica un entità composta da un gruppo di farmers che si coalizzano unendo la loro capacità di storage, ciò permette di avere una probabilità molto più alta di riuscire a comporre una proof of space "di qualità" e quindi più probabilità di avere ricompense.
Quando un farmer appartenente ad una pool riesce a validare un blocco ha comunque diritto tutte le fee di transazione presenti all'interno di quel blocco. La ricompensa del blocco, che equivale ai nuovi XCH messi in circolazione dal protocollo si dividono nel seguente modo:

- 1/8 della ricompensa del blocco al membro della pool che è riuscito a comporre il blocco.
- i restanti 7/8 della ricompensa agli altri membri della pool (in proporzione allo spazio di archiviazione che mettono a disposizione della pool).

Il vantaggio delle pool è quello di riuscire a garantire delle entrate costanti ai farmer (a patto che la pool sia sufficientemente grande). Operando singolarmente e non in una pool infatti, nonostante in caso di successo nella composizione di un blocco si abbia diritto all'intera ricompensa, si ha una probabilità molto inferiore di successo (paragonabile al rapporto tra il proprio spazio di storage e quello totale della rete).

Il plot NFT

Chia network consente di associare i propri plot ad un cosiddetto 'plot NFT', il quale è appunto un NFT che può essere associato ai plot nel momento della loro creazione. Ogni plot può essere associato ad un solo NFT e tramite quest'ultimo è possibile assegnare i plot associati ad un determinato NFT ad una specifica pool. In questo modo è possibile dinamicamente spostare i propri plot da una pool all'altra con una sola transazione sulla blockchain, che consiste nel cambiare l'indirizzo a cui fa riferimento il proprio NFT.
Dal momento che il sistema di pooling non era disponibile immediatamente all' uscita di Chia network esistono dei plot che non sono mai stati assegnati ad un NFT, i cosi detti 'OG plots' (original plots), che non possono quindi essere usati per farmare in una pool. Il protocollo in ogni caso consente il farming di OG plots e plots con NFT in contemporanea sulla stessa macchina.

3.3 L'implementazione

Mi è stato richiesto dall' azienda di implementare un nodo Chia network per valutare possibili rendite contribuendo alla rete.

3.3.1 I primi passi

Inizialmente ho effettuato qualche prova di plotting con una macchina già disponibile in azienda, un Lenovo ThinkCentre Edge 72 (il medesimo modello utilizzato per il nodo Bitcoin, anche qui dal momento che era già disponibile in azienda i primi passi sono stati compiuti su questo dispositivo). Con questa macchina i primi tentativi di plot sono risultati molto lunghi, nell'ordine di grandezza delle 6-7 ore. Ciò non stupisce se si considera che il processore i5 443305 possiede 4 soli core e il disco rigido utilizzato come temp permette di arrivare a velocità rispettivamente di

lettura e scrittura attorno ai 100 MB/s (disco rigido a 7200 rpm). E' evidente che di questo passo fosse impossibile arrivare in tempi utili ad un numero di plot che ci permettesse di stimare i guadagni che può portare l'attività di farming. Per questo motivo si è voluto investire su una nuova macchina che potesse produrre plot in tempi più brevi.

3.3.2 La configurazione del plotter

Si è scelto di configurare un computer assemblato cercando di concentrare le risorse economiche disponibili su un processore con elevato numero di core e un disco ad alte prestazioni.
Il processore richiede un elevato numero di core perché gli algoritmi di plot disponibili attualmente consentono la parallelizzazione, tuttavia solo tramite CPU. Sembra si stia muovendo qualcosa per quanto riguarda lo sfruttamento di GPU per l'operazione di plot, tuttavia non in via ufficiale. Inoltre questa concezione è contro la logica di Chia che punta a limitare il consumo energetico.
La memoria SSD è invece un componente fondamentale in quanto il processo di plot richiede diverse operazioni di scrittura e lettura su disco. E' quindi importante che quest'ultimo sia prestante.

La configurazione scelta è la seguente:

Hardware

- Processore: AMD Ryzen 9 5900X, 12 Core, 24 threads, 4,8 Ghz di frequenza massima e 3,7 Ghz di base (400€) raffreddato tramite sistema a liquido Coolermaster Masterliquid con radiatore a doppia ventola (60€).
- Scheda madre: asus prime b550m-a (130€).
- SSD nvme (per la cartella temporanea): Sabren Rocket 4 plus SSD NVMe 2TB, con velocità di lettura e scrittura (350€). rispettivamente di 7100 MB/s e 6600 MB/s raffreddato con apposito dissipatore attivo per M.2 (23€).
- SSD nvme (per sistema operativo) 128 GB (30€).
- Ram: Corsair Vengeance RGB PRO 32 GB (2 x 16 GB) DDR4 3600 MHz (140€), Successivamente aggiornata con Corsair Vengeance RGB PRO SL 128 GB (4x32GB) DDR4 3200 MHz (740€)
- Alimentatore: Aercool VX Plus 550W (30€)
- Storage: Toshiba Nearline HDD 16TB 3,5 pollici SATA (270€ l'uno), successivamente ampliato con diversi HDD collegati ad un controller SATA.

Software

Per quanto riguarda il sistema operativo, nonostante sia riportato sulla documentazione ufficiale di chia network che su windows i plot risultano più lenti di un 5-10% rispetto a macOS o Linux, si è scelto comunque di procedere installando un sistema windows in modo da beneficiare di alcuni tools per il monitoraggio di sistema. Si valuterà eventualmente un passaggio ad un sistema Linux in seguito. L'applicazione del nodo di Chia è fornita direttamente dal team di sviluppo ed è scaricabile dal sito ufficiale. Questo dettaglio mette in risalto il divario in termini di decentralizzazione tra questa Blockchain e Bitcoin: Bitcoin non ha un'implementazione "ufficiale", ma diversi team di sviluppo si sono adoperati per distribuirne la porpria versione. Il fatto che Chia Network abbia un software ufficiale non la rende di per sé una criptovaluta centralizzata ma in ogni caso fa intendere che il team di sviluppo abbia un certo potere potendo scegliere quali funzioni implementare e quali no.
E' possibile utilizzare il software di Chia tramite interfaccia grafica o tramite linea di comando, i procedimenti e i settaggi che verranno descritti nell'elaborato fanno riferimento all'utilizzo mediante interfaccia grafica.

Parametri di settaggio del plotter originale:

- plot size: il valore di K.
- numero di plot da effettuare.
- in coda/parallelo (con eventuale ritardo tra l'avvio di un plot e l'altro nel caso del plot parallelo)
- numero di threads.
- numero di buckets: Possiamo vedere i buckets come dei contenitori situati nel disco che vengono utilizzati dal plotter durante i processi di riordinamento dei dati. minore sarà il numero di contenitori, più il processo di plot farà affidamento sulla RAM durante determinate fasi (e quindi meno cicli di input/output su disco). La documentazione ufficiale riporta come valore ottimale 128.
- bitfield plotting si/no: settando a 'no' questa opzione è possibile risparmiare il 12% delle scritture totali al prezzo di aumentare il carico di lavoro sulla ram. Questa opzione risulta ottimale se si plotta su dischi temporanei lenti.
- possibilità di escludere la cartella di destinazione, in questo modo il plot verrà salvato direttamente sulla directory temporanea.
- farmer public key: se modificato è possibile plottare a nome di un altro farmer

- pool public key: se settata permette di effettuare il plot a nome di un altro farmer (settando anche il campo precedente) associando il plot alla pool desiderata.
- directory temporanea: temp
- directory temporanea secondaria: temp2
- directory finale: final
- pool nft: per associare il plot alla pool

Parametri di settaggio del plotter madMAx:

- plot size: il valore di K.
- numero di plot da effettuare
- numero di threads
- moltiplicatore di thread per la fase 2
- numero di buckets
- numero di buckets per la fase 3 e 4
- possibilità di escludere la cartella di destinazione, in questo modo il plot verrà salvato direttamente sulla directory temporanea
- farmer public key: se modificato è possibile plottare a nome di un altro farmer
- pool public key: se settata permette di effettuare il plot a nome di un altro farmer (settando anche il campo precedente) associando il plot alla pool desiderata.
- directory temporanea: temp
- directory temporanea secondaria: temp2
- directory finale: final
- pool nft: per associare il plot alla pool

3.3.3 Il sistema di archiviazione

Per contenere i plot iniziali ci si è serviti di 3 HDD Toshiba SATA da 16 TB, collegati direttamente alla motherboard con gli opportuni cavi (SATA + alimentazione).
Una volta riempiti questi 3 dischi si è scelto di collegare un adattatore pcie-SATA alla motherboard cercando sul web dei dischi d'occasione con capacità minima di 1TB in modo da limitare le spese.
L'adattatore pcie-SATA (non incluso nel precedente elenco hardware) permette di espandere lo spazio di archiviazione con 20 dischi SATA, che sommati ai 4 slot gia presenti nella scheda madre (quest'ultima già scelta con un occhio di riguardo verso la quantità di slot SATA) peremettono di arrivare ad un totale di 24 slot SATA.
Sfruttando tutti gli slot e qualche ingresso usb è stato possibile raggiungere lo spazio di archiviazione di 69TB. Sarebbe inoltre possibile sfruttare parte del disco temporaneo (temp1) come spazio di storage dal momento che eseguendo un plot alla volta si andrebbero ad occupare al massimo 256GB, lasciando cosi inutilizzati oltre 1,5TB. Tuttavia quest'ultima scelta andrebbe a diminuire le scritture massime (e quindi i plot eseguibili) sull' SSD, quindi a meno che non si decida di non plottare più e di utilizzare la macchina solamente come farmer sarebbe meglio non considerare quest'opzione.

Il problema delle etichette dei dischi

La gestione di tutti questi dischi esterni ha sollevato un problema così banale da non essere stato previsto: Windows gestisce i dischi esterni assegnando loro una lettera, dal momento che però i dischi collegati superavano le lettere dell'alfabeto risultava impossibile identificare più di 26 dischi.
Fortunatamente Windows permette anche di montare le partizioni 'in stile Unix', ovvero assegnando ad un percorso del filesystem il contenuto della partizione e consentendo in questo modo di collegare potenzialmente una quantità di dischi infinita.

3.4 I test

Al fine di trovare una configurazione per ottimizzare il processo di plot si sono svolti diversi test cercando di trovare la configurazione migliore che bilanciasse tempo e usura del disco temporaneo (minore TBW possibile).

3.4.1 Test con plotter originale

Inizialmente si è voluto fare qualche test con l'algoritmo originale, andando a variare il numero di threads e la quantità di ram dedicati. E' interessante notare dalla

tabella 3.2 come al variare della ram assegnata non ci siano rilevanti variazioni di tempo a parità di threads. Il massimo che possiamo trovare è una variazione di 2,89 secondi che equivale al 2,78% del tempo più alto tra i due a parità di threads. Per quanto riguarda il numero di threads assegnati notiamo una differenza più netta, tra la prima e la terza riga, a parità di ram assegnata, si è arrivati ad una differenza di 9.30 secondi, che equivale al 9,17% del tempo più alto.
Da qui quindi è possibile intuire che nel processo di plot (con il plotter originale) influisca maggiormente il numero di thread piuttosto che la ram assegnata.

K	Threads	RAM usage	T1	T2	T3	T4	Ttot
25	4	4000 MiB	00:52.57	00:08.40	00:36.61	00:01.88	01:39.46
25	4	8000 MiB	00:53.84	00:08.54	00:37.07	00:01.86	01:41.31
25	8	4000 MiB	00:42.37	00:08.50	00:37.11	00:02.18	01:30.16
25	8	8000 MiB	00:43.20	00:08.80	00:38.79	00:02.26	01:33.05

Tabella 3.2: NB: il tempo di copia non è stato tenuto in considerazione in quanto trascurabile.

Successivamente si è provato anche un plot con dimensione k=32 il quale ha restituito un tempo di circa 3 ore (con settaggi di default).

3.4.2 Test con plotter madMax

Allo stesso modo si è voluto testare qualche plot con l'implementazione madMax: dal momento che madMax non prevede una parallelizzazione di più plot contemporanei si è scelto di dedicare direttamente il massimo numero di threads. I primi test sono stati sufficienti per decidere di approfondire lo studio del plotter madMax: con k=25 si sono ottenuti risultati tra i 13 e i 21 secondi, mentre con K = 32 è stato possibile ottenere un plot in circa 55 minuti (mantenendo i buckets di default, ovvero 256-256. Vedi tabella 3.3).
Nonostante infatti con il plotter ufficiale sia possibile parallelizzare i plot, questi hanno bisogno di un delay minimo tra una partenza e l'altra che tendenzialmente è pari alla durata della fase 1 che come vediamo dai test della tabella 3.2 è più del 45% del tempo di plot. Ciò significa che per un tempo di plot di 3 ore equivale a circa 80 minuti. Inoltre in caso di crash se ci fossero in esecuzione più plot in parallelo andrebbero persi tutti, mentre con un solo plot in composizione si perderebbe solo quest'ultimo. Ciò porta, insieme alla semplicità di gestire e ottimizzare un solo processo di plot per volta, alla scelta di utilizzare il plotter madMax piuttosto che l'originale.

K	Threads	buckets	T1	T2	T3	T4	TCopia	Ttot	TBW
32	24	1024-1024	30:04.27	14:57.76	26:56.11	01:32.28	06:31.67	80:02.09	1,387
32	24	512-512	13:02.16	09:10.86	22:47.36	01:32.10	06:32.35	53:04.83	1,514
32	24	256-256	17:12.06	12:33.94	18:22.11	01:12.76	06:31.27	55:52.14	1,514
32	24	128-128	23:50.12	14:27.31	09:02.03	01:02.13	06:31.00	54:52.59	1,422
32	24	64-64	24:18.07	15:04.15	06:38.00	01:03.43	06:27.60	53:31.25	1,298
32	24	512-64	13:02.06	09:16.59	10:45.08	01:04.01	06:32.85	40:40.59	1,386

Tabella 3.3

La configurazione ottimale

Per trovare la configurazione ottimale di buckets è stato misurato il tempo di plot e i TBW per ogni coppia di valore (fase 1-2 e fase 3-4). Come illustrato nella tabella 3.3 si è riscontrato che tutte le configurazioni si aggiravano attorno ai 1,3 - 1,5 TBW. Allo stesso tempo è possibile notare come un valore di buckets più alto assegnato alla fase 1-2 tenda a ridurre i tempi di queste fasi (escludendo il valore 1024, il quale sembra essere inefficiente su ogni fase) mentre un valore più basso sembra rendere di più nelle ultime due fasi.
Queste considerazioni hanno portato ad un ulteriore test con settaggi 512-64 il quale ha dimostrato questa tesi riuscendo ad ottenere un plot in un tempo totale di circa 40 minuti.

3.4.3 Ottimizzazione della vita del disco temporaneo

Il processo di plot come già accennato tende a eseguire numerose riscritture sul disco utilizzato come temp. I precedenti test sono stati eseguiti tutti utilizzando come disco temporaneo l'ssd nvme Sabrent da 2 TB che, secondo i dati della casa, dovrebbe garantire 1400 TBW.

TBW

TBW è una sigla che si riferisce a 'TeraByte Written', ovvero i TB fino ad ora scritti su un disco. Questo valore è particolarmente importante nel caso dei dischi a stato solido (SSD) in quanto (a differenza dei dischi rigidi noti anche come HDD) il numero di cancellazioni (e quindi di riscritture) di ogni cella di memoria è fisicamente limitato. Dopo il numero massimo di scritture garantite dalla casa madre il dispositivo SSD potrebbe quindi guastarsi. Per sfruttare al massimo la memoria vengono quindi messe in piedi algoritmi che fanno in modo di distribuire le scritture omogeneamente, evitando che il dispositivo si guasti dopo poco tempo

perché sono avvenute ripetutamente scritture e cancellazioni sulle stesse celle di memoria. Questo accorgimento fa sì che, a parità di tecnologie implementative, una memoria con X GB di spazio di storage abbia la metà dei TBW garantiti rispetto ad una memoria da 2X GB, quest'ultima ha infatti il doppio delle possibili celle utilizzabili per distribuire le riscritture.
Al fine di poter misurare gli effettivi TBW consumati ad ogni plot si è utilizzato il tool 'CrystalDiskInfo' che permette di tenere traccia di numerosi dati relativi ai dischi installati sul sistema, tra cui i TB totali scritti finora su un determinato disco. Facendo quindi la differenza tra questo dato prima e dopo il plot si ottiene i TB effettivamente scritti durante il processo.

Lo stato attuale

Con l'hardware utilizzato nei test, come mostrato dalla tabella 3.3 ad ogni plot si consumano circa 1,4 TBW (con i settaggi dei buckets ottimali), ciò significa che il disco ha un aspettativa di circa:
1400 TBW/(1,4TBW/plot) = 1000 plot
Per aumentare la vita del disco SSD si è scelto di impostare lo spazio temporaneo secondario (temp2) su un sistema ramdisk.
In questo modo tutte le operazioni effettuate nelle fasi 3 e 4 vengono eseguite direttamente in RAM anzichè sull'SSD.

Il ramdisk

Con RAM disk si intende un disco virtuale il cui spazio di archiviazione è allocato direttamente sulla RAM. Tramite un software apposito è quindi possibile dedicare parte della ram alla creazione di una partizione visibile al sistema operativo. In questo modo si può assegnare tale partizione come temp2 durante la creazione del plot.
Per fare ciò ci si è affidati al software open source 'imDisk Toolkit' ed è stato necessario effettuare un upgrade alla RAM, da 32GB a 128GB (i dettagli sono elencati nell'elenco della componentistica hardware).

I risultati ottenuti

Con questa nuova configurazione, creando una partizione ramdisk da 110GB è stato possibile ottenere un plot, in 44:52.63 consumando solamente 0.406 TBW sull'SSD. Questo risultato è più che soddisfacente: permette di allungare di quasi 4 volte la vita del disco con una frequenza di oltre 32 plot giornalieri.

3.5 Studio economico

In base ai risultati ottenuti dai test ora è possibile fare una valutazione delle potenziali rendite che potrebbe generare il farming di Chia. Sono state individuate 2 forme di rendita principali:

- Tradizionale, ovvero plotting + farming.
- plot-as-a-service, che consiste nella produzione di plot per clienti terzi.

3.5.1 Il costo di un plot

Tralasciando inizialmente il costo totale dei componenti del plotter, un plot nel suo processo di produzione consuma 2 risorse principali: energia e cicli di scrittura su SSD. Possiamo quantificare questi consumi partendo da questi valori:

- Costo energetico: 0,35€/kWh (lo stesso utilizzato nelle stime precedenti)
- Prezzo SSD Sabrent rocket 4 plus: €350 (relativo a giugno 2022)

Consumo energetico per plot

Per calcolare il consumo energetico di un plot si è utilizzato un misuratore di potenza che permette di misurare l'energia erogata a partire da un determinato istante di tempo. Grazie a ciò è stato possibile misurare il consumo energetico di ogni fase di plot:

	avg power (W)	time (s)	% time	consumed energy (kWh)
phase 1	321,13	782,06	32,04	0,07
phase 2	267,45	556,59	22,81	0,04
phase 3	292,69	645,08	26,43	0,05
phase 4	285,60	64,01	2,62	0,01
copy	233,06	392,85	16,10	0,03
tot	279,98	2440,59	100,00	0,19

Tabella 3.4

Possiamo quindi considerare:

$$\epsilon = costo\_energia * energia_consumata \tag{3.7}$$

$$\epsilon = 0{,}35€/kWh * 0{,}19kWh = 0{,}07€ \tag{3.8}$$

Dove con ϵ si intende l'incidenza del costo energetico su ogni plot

Consumo SSD per plot

supponendo circa 0,4 TBW per ogni plot, sfruttando la configurazione RAM disk, è possibile stimare tramiite la formula:

$$\psi = (SSD\_disk\_price/total\_TBW) * per\_plot\_TBW \tag{3.9}$$

$$\psi = (350€/1400TBW) * 0{,}4TBW = 0{,}10€ \tag{3.10}$$

Un plot verrebbe quindi a costare, tra consumo energetico e consumo ssd:

$$\phi = \epsilon + \psi \tag{3.11}$$

$$\phi = 0{,}07 + 0{,}10 = 0{,}17€ \tag{3.12}$$

considerazione su Ram disk

E' bene notare che nel caso non si fosse scelto di adottare una soluzione con RAM disk l'incidenza del costo dell'SSD sarebbe stata molto più alta (avendo avuto un consumo di 1,4 TBW ad ogni plot):

$$\psi' = (350€/1400TBW) * 1{,}4TBW = 0{,}35€ \tag{3.13}$$

a qui:

$$\phi' = 0{,}07 + 0{,}35 = 0{,}42€ \tag{3.14}$$

Ogni plot non prodotto con RAM disk ha un sovrapprezzo di circa:

$$\phi' - \phi = 0{,}42€ - 0{,}17€ = 0{,}25€ \tag{3.15}$$

NB: questo valore non tiene conto della differenza di tempo tra il plot con e senza RAM disk che, come visto in precedenza, presentavano una differenza di circa 4 minuti. pertanto potrebbero esserci eventuali ulteriori differenze in termini di consumo energetico, che risulterebbero comunque minime dal momento che il più della potenza viene assorbita dal processore che svolge il medesimo lavoro in entrambi i casi.
Si può quindi stimare che per ammortizzare il costo di una RAM da 128 GB destinata alla configurazione RAM disk (circa 740€) si dovrebbero produrre un minimo di:

$$740/0{,}25 = 2960 \text{ plots} \tag{3.16}$$

3.5.2 Costo della macchina

Una macchina assemblata con la configurazione descritta in precedenza prevede una spesa di circa 1765€ esclusi i costi dedicati allo spazio di storage dei plot. L'azienda ha deciso inizialmente di investire su 3 HDD da 16 TB per un prezzo di 270€ l'uno, facendo levitare la somma totale a circa 2575€ e garantendosi uno spazio di storage di 48 TB.
Successivamente si è ulteriormente ampliato lo spazio di archiviazione con un adattatore pcie-SATA cercando di rimediare online dei dischi a prezzi favorevoli di dimensione minima = 1TB.

3.5.3 Analisi guadagni con strategia plot + farming

Si vuole ora analizzare gli eventuali guadagni che si avrebbero producendo plot con il fine di contribuire alla rete, ovvero il farming. Al fine di avere un punto di vista più generale utilizzerò una variabile λ che corrisponde ad un prezzo medio al TB. Questa variabile è stata ricavata analizzando numerose offerte di HDD presenti online e facendo una media tra quelli più convenienti

$$\lambda = 16{,}66€/\text{TB} \tag{3.17}$$

Supponendo di riempire tutti gli slot sata disponibili (8 sulla motherboard + 20 sulla scheda di espansione) con dischi da 16TB si avrebbe uno spazio di archiviazione di:

$$16\text{TB} * (20 + 4) = 384\text{TB} \tag{3.18}$$

ad un prezzo di circa:

$$384\text{TB} * \lambda = 6397{,}44€ \tag{3.19}$$

che sommato al prezzo dell'adattatore di circa 200€ arriverebbe ad essere 6597,44€. Con 384TB, secondo il portale `https://chiacalculator.com/` si dovrebbe avere un entrata di circa 170€/mese. Questo valore è stato calcolato tenendo in considerazione i prezzi di XCH nel periodo di giugno 2022, purtroppo la caduta del mercato di questi ultimi mesi ha influito molto su questo studio, Questo calcolo, se fatto con i prezzi di novembre 2021 (ovvero il periodo in cui questo progetto di tesi è nato), avrebbe portato ad un entrata mensile di circa 646€.
Bisogna tuttavia considerare anche il costo energetico che comporta mantenere accesa la macchina per il farming: Considerando un consumo di base di circa 200W andrebbe poi sommato il consumo di ogni disco (circa 5W l'uno) che porterebbe quindi i consumi a circa 320W. In una giornata si andrebbero a consumare quindi attorno ai 7680Wh = 7,6kWh, con il prezzo dell'energia già considerato equivarrebbe a 2,69€ al giorno, circa 81€ al mese.

Ricapitolando:

- spesa macchina: 1765€
- spesa storage: 6397,44€
- spesa energetica per effettuare 384 plot: 26,88€ (NB: per 384 plot è sufficiente un solo SSD già conteggiato nel prezzo della macchina)
- tempo necessario per effettuare 384 plot: 11 giorni
- entrate lorde: 170€/mese
- entrate al netto delle spese energetiche per il farming: 89€/mese
- spesa iniziale totale: 8189,32€
- tempo per rientrare nell'investimento iniziale: 92,01 mesi $>$ 7 anni

Dal punto di vista economico quindi, il farming di Chia non è conveniente almeno nella condizione attuale, la principale ragione è dovuta agli alti costi delle memorie di storage che come si è visto portano la spesa iniziale ad essere piuttosto elevata. Inoltre la situazione geopolitica di questi mesi sta portando ad un incremento dei prezzi dell'energia, ciò accompagnato allo svalutazione generale del mercato cripto scoraggia ulteriormente questa pratica.

3.5.4 Analisi guadagni plot-as-a-service

Online sono disponibili diversi servizi che offrono il servizio di plot-as-a-service (ovvero la produzione di plot su richiesta) e l'azienda potrebbe valutare l'eventuale ingresso in questo business. Analizzando il mercato sembra che i plot siano venduti ad un prezzo minimo di 0,20$ che con il cambio attuale equivarrebbe a 0,19€ circa. Se si pensasse quindi di vendere i plot a prezzi competitivi, considerando che un plot ha un costo di produzione di 0,17€, si avrebbe un margine di guadagno di 0,02€ a plot.
Supponendo di plottare a pieno regime (32 plot al giorno) si riuscirebbero ad ottenere circa 0,64€. Considerando il prezzo della sola macchina senza SSD destinata alla directory temporanea (1410€), già compreso nel costo del plot, a cui è necessario aggiungere una minima capacità di storage per contenere i plot prodotti fino al download del cliente (almeno 3 giorni di plot = 32*3*106GB = 10,18 TB, un HDD di queste dimensioni dovrebbe aggirarsi attorno ai 170€).
La spesa iniziale ammonterebbe quindi a 1580€, per ripagare una tale cifra andrebbero prodotti:

$$1580€/0{,}02€/\text{plot} = 79000 plot \tag{3.20}$$

Un tale numero di plot richiederebbero più di 6 anni e mezzo per essere prodotti dalla macchina in nostro possesso.
Alcuni siti sostengono di poter eseguire il plot in soli 20 minuti, probabilmente grazie al supporto di RAM ad alta capacità, ciò implica un maggior carico sull'investimento iniziale ma si annullerebbe la parte del costo di plot relativo all'SSD. Il costo di un plot quindi equivarrebbe al solo costo energetico che presumibilmente potrebbe essere inferiore a quello da noi considerato.

3.6 Considerazioni finali

L'idea proposta da Chia Network per far fronte alle problematiche di Bitcoin è senz'altro valida dal momento che riesce ad emulare la proof of work senza ricorrere a grossi dispendi energetici. Ci sono comunque delle problematiche legate al vasto uso di dischi rigidi che il protocollo richiede: In vietnam ad esempio si è verificata una grande indisponibilità di dischi rigidi dal momento che nella vicina Cina c'è stato un boom di richieste dovuta proprio al farming di Chia [Lam, 2021].
La pratica di farming non sembra essere particolarmente redditizia visto gli ingenti costi dei dispositivi di storage e l'andamento generale del mercato delle criptovalute. Bisogna poi tenere in considerazione che la criptovaluta Chia è altamente deflattiva, non avendo un tetto di supply massima il suo valore nel tempo continuerà gradualmente a scendere riducendo i guadagni.

Bibliografia

[Cohen and Pietrzak, 2019] Cohen, B. and Pietrzak, K. (2019). The chia network blockchain.

[Dai, 1998] Dai, W. (1998). B-money-an anonymous, distributed electronic cash system.

[de Vries and Stoll, 2021] de Vries, A. and Stoll, C. (2021). Bitcoin's growing e-waste problem. Resources, Conservation and Recycling, 175:105901.

[Huang et al., 2021] Huang, J., O'Neill, C., and Tabuchi, H. (2021). Bitcoin uses more electricity than many countries. how is that possible? The New York Times, 3.

[IEA, 2014] IEA (2014). Around $80 billion wasted on power for online devices in 2013. https://www.iea.org/news/around-80-billion-wasted-on-power-for-online-devices-in-2013.

[Lam, 2021] Lam, B. (2021). China cryptocurrency craze drives hard drive shortage in vietnam. VnExpress.

[Lightning Engineering, 2021] Lightning Engineering (2021). Operational safety. https://docs.lightning.engineering/lightning-network-tools/lnd/safety#aezeed:~:text=A%20BIP%20for%20the%20aezeed%20scheme%20is%20being%20written%20and%20should%20be%20published%20soon.

[Long and Network, 2018] Long, L. and Network, C. (2018). Binary quadratic forms. Website, https://github.com/Chia-Network/vdf-competition/blob/master/classgroups. pdf.

[Nakamoto, 2008] Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system. Decentralized Business Review, page 21260.

[Ponciano, 2021] Ponciano, J. (2021). Bill gates sounds alarm on bitcoin's energy consumption–here's why crypto is bad for climate change. Forbes.

[Rybarczyk et al., 2021] Rybarczyk, R. et al. (2021). On bitcoin's energy consumption: A quantitative approach to a subjective question. https://docsend.com/view/adwmdeeyfvqwecj2.

[Taaki, 2011] Taaki, A. (2011). Bip purpose and guidelines. https://github.com/bitcoin/bips/blob/master/bip-0001.mediawiki.