



UNIVERSITÀ DEGLI STUDI ROMA TRE

Facoltà di Ingegneria
Corso di Laurea Magistrale in Ingegneria Informatica

Tesi Di Laurea

Blockchain













Laureando
Gianmaria Del Monte
Matricola 499829

Relatore
Prof. Maurizio Pizzonia

Anno Accademico 2019/2020

qui la dedica

Elenco delle cose da fare

	inserire immagine di come un indirizzo bitcoin deriva da una chiave pubblica	2
	cambiare stile tabella	3
	fare figura blockchain bitcoin	9
	definire blockchain pubbliche e private nel'introduzione a blockchain	9
	forse definire hash power	9
	inserire citazioni agli attacchi	9
	immagine eclipse attack - tipo http://cs-people.bu.edu/heilman/eclipse/eclipseicon1.png	9
	inserire riferimenti a trilemma	11
	immagine trilemma	11
	citazione pdf visa	11
	citazione EOS	12
	inserire figura mht	13

Introduzione

Introduzione

Indice

Introduzione	iv
1 Background	1
1.1 Bitcoin e Blockchain	1
1.2 Blockchain Trilemma	11
1.3 Strutture dati autenticate	12
2 Stato dell'arte	15
2.1 Algorand	15
2.2 Bernardini	15
2.3 Sharding	15
3 Una soluzione scalabile	16
3.1 Architettura	16
3.2 Analisi Multicast	16
3.3 Teoremi	16
Conclusione	17
Ringraziamenti	18
Bibliografia	19

Capitolo 1

Background

1.1 Bitcoin e Blockchain

Bitcoin è una tecnologia open-source per lo scambio di valute, denominate *bitcoin*, decentralizzato, presentato nel 2008 da una persona anonima, con lo pseudonimo di Satoshi Nakamoto [9]. A differenza di valute tradizionali che esistono fisicamente sotto forma di banconote, i bitcoin sono monete virtuali. Esse vengono scambiate tra i partecipanti alla rete Bitcoin che comunicano mediante il protocollo Bitcoin. Quindi con il termine Bitcoin, si indicano vari aspetti: la tecnologia in sé, lo stack protocollare di comunicazione adottato tra i partecipanti alla rete e la valuta scambiata. Bitcoin è una rete P2P, a cui partecipano nodi chiamati *peer*, in cui non esistono nodi speciali o più importanti di altri, come nei sistemi di pagamento elettronici tradizionali, in cui c'è un server centrale che gestisce tutti i pagamenti.

Il concetto fondamentale di Bitcoin è quello di *transazione*: una transazione trasferisce dei bitcoin da un conto sorgente ad un conto destinazione. Le transazioni possono essere create da qualsiasi peer della rete, che dimostri essere proprietario del conto sorgente e vengono inviate a tutti i nodi della rete. A differenza di pagamenti elettronici tradizionali, in cui un server centrale accetta o rifiuta le transazioni generate dai propri clienti, le transazioni sono accettate o rifiutate dalla rete Bitcoin secondo un meccanismo di *consenso distribuito*, utilizzando un approccio denominato *Proof-of-work*. Le transazioni accettate vengono quindi raggruppate in blocchi di transazioni, secondo un processo che richiede un'enorme quantità di potenza computazionale, che vengono aggiunti ai blocchi della *blockchain*. Questo processo è definito *mining* ed è svolto dai peer che ricoprono il ruolo di *miners*. Il mining ha due obiettivi:

1. creazione di bitcoin: ogni nodo che aggiunge un blocco alla blockchain

viene ricompensato dalla rete con una quantità di bitcoin fissata per ogni blocco e che decresce nel tempo;

2. validazione delle transazioni secondo le regole di consenso, assicurando che le transazioni siano non valide e non corrette.

Inizialmente il mining veniva effettuato da personal computer potenti. Man mano che i miner si aggiungevano alla rete Bitcoin, per cui diveniva sempre più difficile *minare* un blocco, si utilizzarono delle Graphical Processing Units, o GPU, come quelle utilizzate nei videogiochi. Tuttavia negli ultimi anni, a causa dell'elevato numero di miner presenti sulla rete, si utilizzano sistemi Application Specific Integrated Circuit, o ASIC, che implementano in hardware gli algoritmi di mining impiegati in Bitcoin per aumentare le performance. Sono state create anche soluzioni che hanno l'obiettivo di condividere la propria potenza computazionale in mining farm, o mining pool, con diversi partecipanti, in cui si suddividono i bitcoin guadagnati.

1.1.1 Wallet e indirizzi digitali

Un nodo per partecipare alla rete e scambiare i propri bitcoin, ha bisogno di una *chiave digitale* e di un *bitcoin address*. Un utente può generare un numero qualsiasi di chiavi digitali, che vengono memorizzate in un database locale, denominato *wallet*. La possibilità di generate autonomamente delle chiavi evidenzia la caratteristica fondamentale di Bitcoin, ovvero quella della decentralizzazione del controllo e del consenso, basate su prove crittografiche. In particolare, il modello di riferimento è quello della crittografia a chiave asimmetrica, in cui le chiavi digitali sono sottoforma di una coppia di chiavi, una *privata*, segreta, ed una *pubblica*, che deriva dalla chiave privata. È importante specificare che l'uso della crittografia non ha uno scopo di *confidenzialità*, d'altronde tutto in Bitcoin viaggia in chiaro, ma per *autenticità*, per dimostrare quindi chi si dice di essere, ovvero possessori di un certo conto. Secondo una relazione matematica, la chiave privata può essere usata per firmare un messaggio, o in questo caso una transazione, e la chiave pubblica, conosciuta da tutti i nodi della rete, per validare la firma del messaggio. Infatti, semplicisticamente, la chiave pubblica è usata per ricevere i bitcoin, mentre la chiave privata per spenderli, secondo delle modalità che saranno note nel proseguimento della lettura dei prossimi paragrafi, in cui si mostra la creazione di una transazione da parte di un client, e quindi le prove crittografiche che deve mostrare in modo da poter spendere i suoi bitcoin. In una transazione, l'indirizzo di destinazione è indicato da un *bitcoin address*, che deriva dalla chiave privata del client, mediante una serie di funzioni hash, come illustrato nella Figura ??.

inserire immagine di come un indirizzo bitcoin deriva da una chiave pubblica

associati al bitcoin address del nodo stesso, presenta la sua chiave pubblica e la firma di una certa stringa che dimostra che effettivamente lui è il proprietario di quella somma. In questo modo, chiunque nella rete, con queste due informazioni, può verificare che il nodo è realmente il proprietario di una certa quantità di bitcoin spesi in una transazione. È chiaro che chiunque in possesso della chiave privata di un nodo, può spendere a suo piacimento i bitcoin associati al relativo bitcoin address. Per questo, e anche per anonimizzare i creatori delle transazioni, un wallet genera ad ogni transazione una nuova coppia di chiavi.

Un wallet, come si già è detto in precedenza, è un client che memorizza le chiavi digitali di un nodo. Può essere: (1) *non-deterministico* e generare chiavi private in modo totalmente randomico, per cui il wallet è un database che memorizza queste chiavi private, e (2) *deterministico* e generare delle chiavi private a partire da un unico *seed*, che può essere anche una password digitata da un utente, per cui non ha bisogno di memorizzare alcuna chiave.

1.1.2 Transazioni

Le transazioni ricoprono un ruolo fondamentale in Bitcoin e nelle tecnologie di blockchain in generale. Le transazioni sono delle strutture dati che codificano un trasferimento di bitcoin da una o più sorgenti, denominate *input*, ad una o più destinazioni, denominate *output*. I campi di una transazione sono riportati nella Tabella 1.1.

cambiare
stile
tabella

Vediamo ora il ciclo di vita di una transazione. Ogni nodo nella rete può generare una transazione. Il nodo durante la creazione può essere online o offline. La transazione contiene una firma che dimostra che il nodo possiede i bitcoin specificati nell'input. La transazione viene quindi inviata in broadcast, raggiungendo tutti i nodi della rete. Questa viene messa nella lista delle transazioni in attesa, *pending transactions*, verificata da ogni nodo, e se è valida, ovvero è conforme alle regole di consenso, viene inclusa all'interno di un blocco insieme ad altre transazioni e aggiunta in coda alla blockchain.

Non esiste il concetto di conto relativo ad un account Bitcoin, per cui le transazioni formalmente non spostano valuta da un conto all'altro. In altre parole non esiste un database o una DHT come si vedrà in altre tecnologie blockchain presentate nei prossimi paragrafi, ma i bitcoin posseduti da un certo indirizzo derivano scandendo tutta la blockchain dal blocco iniziale al blocco attuale, considerando le *unspent transactions output*, o UTXO. Le UTXO sono i mattoncini con cui vengono generate le transazioni e sono quelle transazioni, registrate nella blockchain, a cui è associato un valore espresso in *satoshi*, dove 1 satoshi è equivalente a 10^{-8} bitcoin, e bloccate da un segreto di cui è a conoscenza solo il possessore. I satoshi corrispondono

Dimensione	Campo	Descrizione
4 byte	Versione	Specifica la versione del protocollo
1-9 byte variabile	Contatore input Input	Indica quanti input sono inclusi Lista di uno o più input
1-9 byte variabile	Contatore output Output	Indica quanti output sono inclusi Lista di uno o più output
4 byte	Time	Timestamp Unix

Tabella 1.1: I campi di una transazione.

ai centesimi presenti nelle valute tradizionali, come l'Euro o il Dollaro, solo che, a differenza di queste ultime in cui l'unità di base può essere suddivisa massimo in 100 parti, il bitcoin, può essere suddiviso fino a 100 milioni di parti.

Le UTXO sono generate a partire dall'output di una transazione, aggiunta in un blocco della blockchain, che non è stata ancora spesa in un'altra transazione. Gli output di una transazione consistono in due parti:

1. il valore espresso in satoshi da trasferire;
2. un *locking script*, che specifica il modo in cui i bitcoin possono essere sbloccati.

Per semplicità un locking script può essere pensato come l'indirizzo della destinazione, che diventerà, una volta che la transazione è stata accettata e quindi aggiunta in un blocco della blockchain, il proprietario della quantità di bitcoin indicati nella transazione stessa.

Gli input in una transazione sono riferimenti ad UTXO. In particolare specificano l'UTXO da utilizzare come sorgente tramite l'hash della transazione e l'indice dell'UTXO nella transazione considerata. Il creatore della transazione deve includere un *unlocking script* per ogni UTXO, contenente una firma che dimostri che il peer che ha creato la transazione è proprietario di quei bitcoin.

Molte transazioni includono delle *tasse*. Esse hanno un triplice scopo. Il primo è incentivare i miner a scegliere la propria transazione. I miner, oltre a ricevere un compenso per la creazione del blocco, ricevono le tasse di tutte le transazioni presenti nel blocco creato. Infatti, quando i miner provano a generare un blocco, scelgono dal proprio pool di transazioni in attesa quelle con le tasse più alte. Il secondo è quello di incentivare un peer ad essere un miner, in modo tale che la rete sia più sicura. Come si vedrà successivamente la sicurezza dell'intero sistema aumenta con l'aumentare dei

nodi che partecipano alla rete. L'ultimo, disincentiva nodi malevoli a *spam* di transazioni, perchè questi si troverebbero a spendere una piccola quota in tasse per ogni transazione generata. La quantità di tasse da pagare per una transazione normalmente dipende dalla dimensione in byte delle transazioni, in genere sui 370 byte, ma molto dipende dalle richieste di mercato. Al momento di scrittura di questa tesi, per una transazione di circa 370 byte composta da 2 input e 2 output, affinché sia accettata entro 2 blocchi, quindi entro 20 minuti, sono necessari 34k satoshi, pari a 3.20 USD (fonte [1]).

Come si è visto, nella struttura dati di una transazione non è presente un campo tasse esplicito. Le tasse sono infatti implicite e sono calcolate da ogni miner come la differenza tra la somma degli input e la somma degli output. In formule,

$$Tasse = \sum T_{in} - \sum T_{out}$$

dove T_{in} e T_{out} indicano l'input e l'output di una transazione T , rispettivamente.

Gli script sono il sistema con cui i peer Bitcoin validano le transazioni. Come si è detto in precedenza, ad ogni UTXO è associato un locking script, che rappresenta le condizioni che un nodo deve soddisfare per usare i bitcoin contenuti in esso. L'input di una transazione, riferito ad uno specifico UTXO, deve fornire un unlocking script, che contiene solitamente una firma che sblocca i fondi. Durante la validazione di una transazione, il peer esegue prima l'unlocking script e usa il suo output come input per il locking script. Se non ci sono errori e le condizioni nel locking script sono soddisfatte, ovvero l'intera operazione restituisce *true*, allora la transazione è considerata valida. Gli script sono scritti in un linguaggio stack-based, in cui sono presenti istruzioni aritmetico-logiche, istruzioni per il calcolo di hash, di verifiche di chiavi pubbliche, condizioni, ma non loop, il che lo rendono un linguaggio *Turing incompleto*. Questo fa sì che non è possibile creare un loop infinito e causare quindi un *Denial-of-Service*. Grazie ad un linguaggio di scripting è possibile scrivere un'infinità di script che danno modo di esprimere qualsiasi condizione. La più usata è certamente la *Pay to public key hash*, nota anche con P2PKH, che consente di specificare l'indirizzo Bitcoin a cui trasferire una certa somma di bitcoin. Ma è possibile creare fondi sbloccabili solo da più utenti insieme, o combinazioni più bizzarre o anche da una certa data in poi. Questi sono solo alcuni degli esempi che è possibile creare.

1.1.3 Mining

Il mining ha lo scopo principale di confermare le transazioni in attesa, generando un nuovo blocco da aggiungere alla blockchain, con un mecca-

nismo completamente decentralizzato, dove ogni nodo della rete, chiamati *miner*, possono contribuire al processo, senza che ci sia un'autorità centrale o un comitato speciale che gestisca tutta l'operazione. Circa ogni 10 minuti viene creato un nuovo blocco contenente le transazioni confermate da aggiungere in coda alla blockchain. Il mining ha anche lo scopo di generare dal nulla nuovi bitcoin, da qui il termine mining, che allude all'estrazione di pietre preziose dalle miniere, con una decrescita esponenziale nel tempo di creazione di valuta. Ogni nodo che riesce ad inserire un nuovo blocco alla blockchain, viene ricompensato con una quantità fissa di bitcoin, dimezzata ogni 210.000 blocchi (circa ogni 4 anni), partendo dalla quantità di 50 bitcoin a gennaio 2009. Dall'11 maggio 2020, infatti, la quantità di bitcoin ricevuta da ogni miner che riesce ad aggiungere un nuovo blocco alla blockchain è di 6,25 bitcoin. Usando questa formula, la ricompensa decresce esponenzialmente fino al 2140, quando circa 21 milioni di bitcoin (per l'esattezza 2.099.999.997.690.000 satoshi) saranno generati, approssimativamente dopo 13.44 milioni di blocchi. Un miner riceve come ricompensa anche le tasse delle transazioni che ha inserito nel nuovo blocco. Per cui dopo il 2140, i miner riceveranno come ricompensa solo le tasse presenti nelle transazioni. Quello della ricompensa è ovviamente un meccanismo atto ad incentivare la presenza di più miner in rete, che concorrono alla conferma di nuove transazioni, aumentando in questo modo la sicurezza dell'intera rete.

Finora si è parlato dei miner che cercano di creare un nuovo blocco da aggiungere alla blockchain, senza specificare il come. Vediamo quindi come i miner creano i nuovi blocchi della blockchain, secondo un meccanismo di consenso decentralizzato, che rappresenta il vero contributo di Satoshi Nakamoto alle reti P2P, secondo cui tutti i nodi presenti nella rete concordano su un certo stato della blockchain, senza un'autorità centrale che governi tutto. Il consenso è infatti raggiunto in maniera asincrona su tutta la rete, per cui ogni nodo può vedere lo stesso stato della blockchain. Quando una transazione viene generata da un qualsiasi peer della rete, essa viene inviata in broadcast a tutti i nodi della rete. Ogni nodo che riceve una transazione, prima di inviarla ai peer successivi, effettua numerosi controlli di validità, come la conformità alle regole sintattiche e regole di consenso, come verificare che la transazione non faccia *double spending*, ovvero la transazione spende più di una volta lo stesso UTXO, confrontando quindi gli input della transazione con gli output delle transazioni nei blocchi della blockchain e con quelle del pool delle transazioni, verificare che la somma degli input sia maggiore della somma degli output, verificare che l'output dell'unlocking script di ogni input sblocchi il locking script dell'UTXO a cui l'input si riferisce. Se una transazione passa tutti questi controlli, viene inserita nel pool delle transazioni del peer, uno spazio di memoria temporaneo in cui risiedono le transazioni

valide in attesa di essere inserite in un blocco, per poi essere inviata in broadcast ad ogni peer vicino. Ogni peer effettua tutti questi controlli. A questo punto, ogni miner sceglie le transazioni che formeranno il prossimo blocco, denominato *candidate block*, dal pool delle transazioni. I primi 50 kilobyte di un blocco sono formati da quelle transazioni ad *alta priorità*: la priorità viene calcolata in base all'età dell'UTXO che dovrà essere speso, ovvero in base alla profondità a cui si trova nella blockchain la transazione contenente l'UTXO rispetto al blocco corrente. La formula per il calcolo della priorità è la seguente:

$$Priorità = \frac{\sum_i value(T_i) * age(T_i)}{size(T)}$$

dove $value(T_i)$ è il valore dell'input i della transazione T espresso in satoshi, $age(T_i)$ è la profondità dell'UTXO a cui si riferisce l'input i della transazione nella blockchain, e $size(T)$ è la dimensione della transazione espressa in byte. Questo consente alle transazioni di essere inserite anche se non hanno alcuna tassa. Il resto del blocco, che ha una dimensione massima di 4 megabyte, viene riempito dalle transazioni presenti nel pool che hanno una tassa maggiore. Questo perché tutte le tasse presenti nelle transazioni vengono ricevute dal nodo miner che ha creato il blocco. Infine, la prima transazione che viene inserita nel blocco è quella denominata *coinbase*, una transazione speciale che non contiene alcun input, il cui output contiene come valore la somma delle tasse presenti nelle altre transazioni del blocco e la ricompensa per il blocco creato e avete come indirizzo destinazione quello del nodo miner. La *coinbase* rappresenta quindi la transazione tramite la quale il nodo miner riceve la sua ricompensa per il lavoro svolto.

Vediamo ora come funziona il consenso decentralizzato creato da Satoshi Nakamoto, che rappresenta il suo più grande contributo alle reti P2P. Una volta costruito il blocco, esso deve essere *minato*, ovvero si deve trovare la soluzione all'algoritmo *Proof-of-work* che rende valido il blocco, in altre parole tutti i nodi della rete lo riconoscono come valido. Esso si basa sugli *hash crittografici*, denominati in seguito più semplicemente con hash. Un'hash è una funzione che dato un input di una qualsiasi dimensione, produce una stringa di output di dimensioni fisse. Essa è semplice da calcolare, ma difficile da un punto di vista computazionale invertire, ovvero dato un hash h , calcolare l'input S tale che $h = hash(S)$. Ad esempio, la funzione SHA256, utilizzata in Bitcoin, produce un output di 256 bit. Inoltre, un'altra proprietà fondamentale è che a fronte dello stesso input si produce lo stesso output, perciò l'hash è una funzione deterministica. Secondo Proof-of-work un miner deve trovare un blocco, tale che l'hash del suo header sia minore di una certa soglia, denominata *target*. Questo è trovato variando una parte

dell'header del blocco denominato *nounce*, e viene inserito nell'header del blocco come prova del lavoro svolto. Poiché gli hash non seguono un certo pattern, l'unico modo di trovare un nounce che faccia assumere all'header del blocco un valore minore del target è quello di tentare tutte le combinazioni. Essendo l'hash facile da calcolare, verificare il lavoro svolto da un miner è un compito semplice. Il primo nodo miner che riesce a trovare una soluzione a questo puzzle crittografico, è il vincitore di questa competizione ed invia in broadcast il blocco minato come prova del lavoro svolto. Il target determina la difficoltà nel trovare un blocco che soddisfi le condizioni di consenso, in quanto questo determina un aumento esponenziale del numero di tentativi da fare per risolvere il problema. Poiché la potenza computazionale aumenta nel tempo, e quindi il numero di hash calcolabili nell'unità di tempo, c'è un meccanismo che reimposta il target ogni 2016 blocchi, circa ogni 2 settimane, in modo tale che in media un blocco sia minato in 10 minuti. Il miner che mina il blocco k tale che k sia divisibile per 2016, determina il nuovo target sulla base del tempo impiegato a trovare gli ultimi 2016 blocchi: se il tempo è maggiore di 20160 minuti aumenta il target, diminuendo quindi la complessità, se è minore diminuisce il target, aumentando la complessità del problema. Quando gli altri miner ricevono il blocco minato da un altro miner, smettono di minare il proprio blocco e cominciano a lavorare sul successivo, utilizzando lo stesso approccio appena descritto.

Può capitare che due miner nella rete trovino riescano a minare quasi contemporaneamente un blocco, per cui i vicini potrebbero lavorare a loro volta su viste della blockchain differenti. Questo genera nella blockchain delle *fork*, per cui i nodi nella rete hanno visioni differenti della blockchain stessa. Questo problema viene risolto da ogni nodo scegliendo il ramo più lungo, in questo modo si converge ad una visione comune della blockchain.

1.1.4 Blockchain

La blockchain è il registro contenente tutte le transazioni accettate, condivisa da tutti i nodi della rete. Essa è una lista semplicemente collegata i cui nodi sono i blocchi. I blocchi sono connessi al contrario, per cui ogni blocco punta al nodo precedente. Ogni blocco è identificato da un hash crittografico, calcolato con l'algoritmo SHA256 sull'header del blocco stesso. Ogni blocco punta al nodo precedente specificandone l'hash. La sequenza di blocchi crea quindi una catena, fino ad arrivare al primo blocco creato, denominato *genesis block*, creato nel 2009, e direttamente codificato nel client software di Bitcoin, per cui conosciuto da tutti i nodi della rete.

Un blocco è un contenitore formato da una sezione metadati, composto dalla dimensione del blocco stesso, da un block header di 80 byte e dalla lista

di transazioni validate. Il block header è composto dall'hash del precedente blocco, dal *root-hash* del Merkle Tree formato dalle transazioni contenute nel blocco, un *timestamp*, ovvero il tempo di creazione approssimato del blocco, un *difficulty target* e un *nounce*, utilizzati nella fase di mining. La figura ?? mostra un esempio di blockchain.

1.1.5 Attacchi noti

51% attack Qualsiasi tecnologia di blockchain per sua natura si basa su un meccanismo di consenso distribuito che garantisce una mutua fiducia. Se nelle blockchain basate su Proof-of-work un miner possiede più del 50% dell'hash power, può sfruttare l'attacco 51% per ottenere il 100% delle ricompense dal mining, poichè riuscirebbe a creare catene di blocchi sicuramente più lunghe di qualsiasi altro miner, per fare un attacco *double spending*, in cui si utilizza più volte lo stesso importo per i pagamenti, eliminare dalla blockchain gli ultimi blocchi confermati ed eventualmente corrompere la blockchain stessa. In una tecnologia come Bitcoin che incentiva economicamente i nodi a diventare miner e che ha raggiunto nel tempo un numero di nodi elevato, e quindi un grande hash power, un simile attacco è molto improbabile. Sicuramente non sono escluse da questo attacco le blockchain di piccole dimensioni, che hanno un hash power di gran lunga più piccolo di quello di Bitcoin: esempi di cripto-valute che sono state vittime del 51% attack sono Monacoin, Bitcoin Gold e ZenCash. In realtà l'avvento delle mining pool in Bitcoin, in cui più miner mettono a disposizione la propria potenza computazionale al servizio di un gruppo di più miner che divide il compenso in modo proporzionale all'hash power condiviso, apre la possibilità ad un'organizzazione del genere ad un 51% attack se la somma totale dell'hash power di tutti i nodi iscritti supera il 50% dell'intero hash power della rete. È stato il caso di **ghash.io**, in cui, dopo aver raggiunto nel gennaio del 2014 il 42% dell'intera potenza computazionale della rete Bitcoin, molti miner si sono disiscritti dal pool e il pool stesso rilasciò una dichiarazione in cui rassicurava la comunità Bitcoin di evitare di raggiungere la soglia del 51% [5].

Eclipse attack L'eclipse attack è un attacco perpetrato ai danni di una singola vittima. Un attaccante, che controlla un gran numero di nodi, ad esempio una botnet, può isolare un nodo del resto della blockchain. Questo gli consente di mostrare una blockchain differente da quella che è in realtà e di sfruttare la potenza computazionale della vittima per i suoi comportamenti malevoli. Heilman et al. [6], mostrano che è possibile utilizzare un attacco eclipse come base per altri attacchi:

fare figura blockchain bitcoin

definire blockchain pubbliche e private nell'introduzione a blockchain

forse definire hash power

inserire citazioni agli attacchi

immaginare eclipse attack - tipo <http://cs-people.bu.edu/heilman/eclipse>

- **Engineering block races:** Quando due miner riescono a minare un blocco nello stesso momento, solo uno entra a far parte della blockchain, mentre l'altro diventa un blocco *orfano*, e il miner che ha minato questo blocco non riceve alcuna ricompensa. Un attaccante che eclissa più di un miner, accumula i blocchi scoperti dai miner attaccati, per poi rilasciarli alla rete solo quando un miner non attaccato scopre un blocco. In questo modo i blocchi minati dalle vittime diventano orfani, e come risultato sia ha uno spreco delle risorse degli attaccati.
- **Splitting mining power:** Si eclissa un certo numero di miner in modo da eliminare dalla rete dell'hash power e consentire più facilmente un 51% attack.
- **Selfish mining:** Questo attacco consente ad un attaccante di sprecare risorse a miner vittime o di ottenere una ricompensa maggiore. L'attaccante mantiene private una lista di blocchi che ha minato ed elimina i blocchi minati dai miner eclissati che competono con i blocchi che lui ha minato. Nello stesso tempo sfrutta la potenza computazionale dei nodi vittime tentando di creare un fork più grande rispetto alla catena di blocchi della vera blockchain.
- **0-confirmation double spend:** Si ha nel caso di *0-confirmation transaction*, dove un venditore spedisce, nel caso di vendita online, o rilascia, nel caso di vendita in un negozio fisico, un bene ad un cliente prima che questa sia confermata in un blocco. Questo accade quando non si vuole attendere i circa 10 minuti di conferma di una transazione. L'attaccante effettua un eclipse attack al venitore, inviandogli una transazione T per comprare i beni ed una transazione T' al resto della rete, facendo un *double spending*. Il mercante rilascia il prodotto al cliente malevolo e poiché è eclissato non può inviare T al resto della rete. Quindi l'attaccante ottiene un prodotto senza aver pagato.
- **N-confirmation double spend:** In questo caso, a differenza del precedente, un mercante rilascia un bene al suo cliente solo se la transazione si trova almeno a profondità $N - 1$ nella blockchain. L'attaccante invia la transazione ai nodi eclissati, che la inglobano in una vista *vecchia* della blockchain, tra cui è presente anche il commerciante. L'attaccante, dopo aver ricevuto il prodotto, mostra alle vittime eclissate la vera blockchain, aggiornata nel frattempo dai nodi non eclissati. La blockchain vista dai nodi eclissati diventa orfana, poiché è un fork più corto rispetto alla vista della blockchain del resto della rete, e la tran-

sazione creata dall'attaccante viene quindi eliminata. Anche in questo caso, l'attaccante ottiene un prodotto senza aver pagato.

1.2 Blockchain Trilemma

Sebbene il *blockchain trilemma* influisca sull'analisi e progettazione di tecnologie di blockchain, non è presente una definizione formale in letteratura, anche se è citato in numerosi lavori, come ???¹. Il termine [12] fu coniato da Vitalik Buterin, creatore di *Ethereum*, una blockchain basata su Proof-of-work, individuando le tre caratteristiche che una blockchain deve avere per allargare i propri confini a livello globale: *decentralizzazione*, *sicurezza* e *scalabilità*. Il *trilemma* afferma che bisogna rinunciare ad almeno un'alternativa. Il blockchain trilemma si può rappresentare con un triangolo, come in Figura ??, ai cui vertici ci sono le tre proprietà e in cui l'ottimo è il centro.

Vediamo in dettaglio i tre elementi fondamentali del trilemma.

Decentralizzazione Per decentralizzazione si intende il fatto che le transazioni siano validate e confermate da un gruppo di nodi e non da un'autorità centrale o un comitato speciale, come accade in sistemi tradizionali, per cui le decisioni sono ottenute da un consenso distribuito. Quindi non è necessario riporre fiducia su una terza parte durante una qualsiasi operazione. In altre parole, le decisioni sono ottenute democraticamente da tutti i partecipanti alla rete, ed ogni proposta di cambiamento nel protocollo può essere accettata se più del 50% dei partecipanti è favorevole. Un esempio è Bitcoin Cash, un fork di Bitcoin avvenuto il 1 agosto 2017 [7], in cui si portò ad 8 MB la dimensione massima di un blocco, in modo da consentire un numero maggiore di transazioni accettate. È chiaro che una decentralizzazione ha come conseguenza una *qualità* di decisione più alta rispetto ad una presa da un'autorità centrale, poiché la conferma prima di raggiungere il consenso è ottenuta da più nodi. Il trade-off è quindi la velocità di conferma: se una transazioni richiede una conferma da più partecipanti la velocità è minore di una decisione presa da un'autorità centrale.

Scalabilità La scalabilità è la proprietà che consente realmente un'adozione a livello globale. Essa si riferisce al fatto che un sistema possa adattarsi ad un incremento di carico. Ad esempio, le due tecnologie di blockchain ad oggi più utilizzate, Bitcoin e Ethereum, possono gestire un carico massimo di 7 e 12 TPS¹ al contrario di Visa, che può arrivare a 65.000 TPS. EOS, una

¹TPS = transazioni al secondo

inserire
riferi-
menti a
trilem-
ma

immagine
trilem-
ma

citazione
pdf visa

blockchain progettata per essere scalabile, ha un throughput dichiarato di circa 2000 TPS, ma promette di poter processare milioni di transazioni in futuro, tuttavia ad un prezzo: la decentralizzazione.

citazione
EOS

Sicurezza La sicurezza deve essere il requisito fondamentale in una blockchain. Se la sicurezza è povera o del tutto assente, un attaccante può spendere più volte la stessa quantità di denaro, *double spending*, arricchendo se stesso a scapito di altri ed arrivare a modificare lo stato della blockchain, che per sua natura è immutabile. Questo può succedere ad esempio nel 51% attack, presentato nel Paragrafo 1.1.5.

È importante notare che il blockchain trilemma non è un teorema, come lo è il CAP theorem, un teorema fondamentale nell'ambito dei sistemi distribuiti. Esso sottolinea la difficoltà nel creare un sistema decentralizzato, sicuro e scalabile. Le blockchain sono ancora tecnologie giovani, poco mature per cui c'è ancora tanta strada da fare, in grado di migliorare ciò che già è stato fatto. Bitcoin, ad esempio, è una blockchain altamente sicura e decentralizzata, ma non scalabile: il numero di transazioni massime supportate è infatti di 7 TPS. Sebbene, non sarà utilizzabile a livello globale come unica valuta, rappresenta sicuramente un punto di svolta, dimostrando che una criptovaluta digitale in un sistema P2P, in cui non è necessario riporre la fiducia su un'autorità centrale o si è soggetti a regole imposte dalla medesima, è possibile e attuabile in pratica.

1.3 Strutture dati autenticate

Come si è visto nella Paragrafo 1.1.4, ogni block header della blockchain contiene un riassunto di tutte le transazioni presenti nel blocco, denominato *root-hash*, ed ottenuto da un Merkle Tree (MHT). Esso fa parte della famiglia delle *strutture dati autenticate*.

Una struttura dati autenticata (ADS) è una struttura dati contenente valori che permette una verifica di integrità, ovvero un client fidato può verificare l'autenticità di una risposta ottenuta da un server non fidato mediante una prova, denominata *proof*. In altre parole, le operazioni effettuate da una parte esterna non fidata, denominato *prover*, possono essere verificate efficientemente da un client, il *verifier*.

Supponiamo di trovarci nel seguente scenario. Un client ha bisogno di memorizzare una certa quantità di dati che eccede le sue disponibilità di memorizzazione fisica. Può memorizzare i suoi dati su un server esterno che ha una grande capacità di memorizzazione, come ad esempio un servizio di

cloud storage. Il client vuole essere sicuro che la versione dei file che memorizza in cloud sia l'ultima e che nessuno possa modificare il contenuto dei suoi file. Questo problema può essere risolto mediante le strutture dati autenticate, senza che il client memorizzi l'intera struttura dati e senza memorizzare l'hash di ogni singolo file. In quest'ultimo caso infatti dovrebbe memorizzare un numero di hash lineare con il numero di file che ha memorizzato in cloud. Applicazioni del genere sono molto frequenti, anche nel caso di Bitcoin, in cui esistono client *leggeri*, i cosiddetti *thin client*, che memorizzano solo l'header dei blocchi della blockchain, mentre tutte le transazioni contenute nei blocchi sono memorizzati su un server esterno. Una realizzazione di un protocollo efficiente che fa uso di ADS si può trovare in [10] e [4].

Un'implementazione efficiente di un'ADS è il Merkle Tree, utilizzato per la realizzazione di dizionari autenticati. Dato un insieme di elementi V , il Merkle Tree T è un albero binario completo bilanciato avente come foglie gli hash degli elementi in V , mentre i nodi intermedi sono il risultato dell'hash della concatenazione dei nodi figli. Il root di T contiene il *root-hash* di T . La Figura ?? mostra un esempio di Merkle Tree. Come ogni ADS, ad una richiesta $T.get(k)$, dove si richiede il valore v associato alla chiave k , si ottiene anche una *proof* associata a v . La proof è una struttura dati utile a verificare l'integrità del valore a cui è associata. In particolare, in un Merkle Tree la proof si ottiene dall'Algoritmo 1.1.

inserire
figura
mht

Algoritmo 1.1 Calcolo della proof in un MHT T associata alla chiave k

```

procedure GET_PROOF( $k$ )
  Sia  $n$  la foglia di  $T$  associato a  $k$ 
  Sia  $p$  il cammino da  $v$  al root
   $P \leftarrow []$ 
  while  $n.parent \neq T.root$  do
    if  $n.parent.right = n$  then                                      $\triangleright n$  è un figlio destro
       $Tag \leftarrow R$ 
       $h_f \leftarrow n.parent.left.value$ 
    else                                                              $\triangleright n$  è un figlio sinistro
       $Tag \leftarrow L$ 
       $h_f \leftarrow n.parent.right.value$ 
     $P.APPEND((Tag, h_f))$ 
     $n \leftarrow n.parent$ 
  return  $P$ 

```

La Figura ?? mostra un esempio di proof, rappresentata con linee tratteggiate per il valore ????. Considerato il cammino p da ??? al root, rappresentato in figura con una linea punteggiata, si selezionano i nodi ???, fratelli

dei nodi percorsi in p . La proof risultante è quindi la lista di coppie $[(???, ???)]$.

Un Merkle Tree utilizza uno spazio $O(|V|)$ e una dimensione della proof, un tempo per una query ed un tempo di verifica pari a $O(\log |V|)$.

Un client possiede solo una copia locale del root-hash, mentre i dati sono su un server non fidato, ad esempio su cloud. Per verificare la veridicità del valore v associato alla chiave k ottenuto a seguito di una richiesta di $get(k)$ al server, il client calcola il root-hash dalla proof associata a v e da v stesso, con l'Algoritmo 1.2.

Algoritmo 1.2 Calcolo del root hash dalla proof

```

procedure Roothash_From_Proof( $p, v$ )
     $h_1 = \text{hash}(v)$ 
     $L = \text{len}(p)$ 
    for  $i \leftarrow 1$  to  $L$  do
         $Tag, h_{p_i} \leftarrow p[i]$ 
        if  $Tag = R$  then
             $h_{i+1} \leftarrow \text{hash}(h_{p_i} \oplus h_i)$             $\triangleright \oplus$ : concatenazione degli hash
        else
             $h_{i+1} \leftarrow \text{hash}(h_i \oplus h_{p_i})$ 
    return  $h_L$ 

```

Capitolo 2

Stato dell'arte

2.1 Algorand

2.2 Bernardini

2.3 Sharding

Capitolo 3

Una soluzione scalabile

3.1 Architettura

3.2 Analisi Multicast

3.3 Teoremi

Conclusione

conclusione

Ringraziamenti

Ringrazio tutti

Bibliografia

- [1] Buy bitcoin worldwide. <https://www.buybitcoinworldwide.com/fee-calculator/>.
- [2] Andreas M Antonopoulos. *Mastering Bitcoin: unlocking digital cryptocurrencies*. " O'Reilly Media, Inc.", 2014.
- [3] CertiK. The blockchain trilemma: Decentralized, scalable, and secure? <https://medium.com/certik/the-blockchain-trilemma-decentralized-scalable-and-secure-e9d8c41a87b3>, 2019.
- [4] Gianmaria Del Monte. *Studio e realizzazione di un protocollo efficiente per l'integrità dei dati su Cloud*. 2018. Tesi di laurea triennale, Università degli studi Roma Tre.
- [5] Nermin Hajdarbegovic. Bitcoin miners ditch ghash.io pool over fears of 51 <https://www.coindesk.com/bitcoin-miners-ditch-ghash-io-pool-51-attack>, 2014.
- [6] Ethan Heilman, Alison Kendler, Aviv Zohar, and Sharon Goldberg. Eclipse attacks on bitcoin's peer-to-peer network. In *24th {USENIX} Security Symposium ({USENIX} Security 15)*, pages 129–144, 2015.
- [7] Alyssa Hertig. Bitcoin cash: Why it's forking the blockchain and what that means. <https://www.coindesk.com/coindesk-explainer-bitcoin-cash-forking-blockchain>, 2017.
- [8] Xiaoqi Li, Peng Jiang, Ting Chen, Xiapu Luo, and Qiaoyan Wen. A survey on the security of blockchain systems. *Future Generation Computer Systems*, 2017.
- [9] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system. Technical report, Manubot, 2019.

- [10] Diego Pennino, Maurizio Pizzonia, and Federico Griscioli. Pipeline-integrity: Scaling the use of authenticated data structures up to the cloud. *Future Generation Computer Systems*, 100:618–647, 2019.
- [11] Roberto Tamassia. Authenticated data structures. In *European symposium on algorithms*, pages 2–5. Springer, 2003.
- [12] Ethereum wiki project. On sharding blockchains. <https://github.com/ethereum/wiki/wiki/Sharding-FAQ>.