ALMA MATER STUDIORUM – UNIVERSITÀ DI BOLOGNA

Corso di Laurea in Ingegneria e Scienze Informatiche

Oltre la Blockchain: Pagamenti Off-Chain e Lightning Network tra Scalabilità e Sicurezza

Tesi di laurea in:
Crittografia

Relatore
Prof. Luciano Margara

Candidato
Patrick Sbrighi

III Sessione di Laurea
Anno Accademico 2024-2025

Sommario

Questa tesi analizza il Lightning Network come soluzione al problema della scalabilità delle blockchain pubbliche, con particolare attenzione agli aspetti architetturali e di sicurezza. L'obiettivo è fornire una panoramica chiara dei meccanismi principali, evidenziandone vantaggi, limiti e possibili sviluppi futuri.

Indice

Introduzione

Con la pubblicazione del whitepaper su Bitcoin da parte di Satoshi Nakamoto[Nak08], viene per la prima volta presentata la blockchain, una tecnologia innovativa che ha ridefinito le fondamenta dei sistemi distribuiti. Sfruttando tale innovazione viene mostrata la possibilità di trasferire valore su scala globale senza la necessità di intermediari fiduciari o autorità centrali. Questa architettura non rappresenta solamente un avanzamento a livello informatico, ma simboleggia anche un cambiamento sociale ed economico, infatti, per la prima volta, la fiducia viene spostata dalle istituzioni agli algoritmi crittografici, permettendo alle parti di cooperare direttamente senza la necessità di conoscersi e fidarsi reciprocamente.

Tuttavia, questo sistema si basa su un registro distribuito mantenuto da una rete peer-to-peer, dove il consenso sullo stato dei fondi viene raggiunto attraverso un protocollo di gossip che diffonde in broadcast ogni transazione a tutti i partecipanti della rete. Tale approccio, sebbene garantisca sicurezza e decentralizzazione, impone pesanti limiti per quanto riguarda la capacità del sistema di gestire elevate quantità di transazioni.

Nello schema originale di Bitcoin ciascun nodo deve validare e memorizzare ogni transazione che avviene a livello globale, il problema principale, infatti, risiede proprio in questo. Se si verificasse un aumento smisurato del numero di transazioni il sistema crollerebbe, mentre se si aumentasse dimensione dei blocchi per inglobare più transazioni verrebbero richieste risorse computazionali e di archiviazione tali da escludere la maggior parte dei nodi domestici, portando inevitabilmente a una centralizzazione dei validatori e minando la natura trustless e decentralizzata del sistema.

Per ovviare a tale problema, e superare quello che verrà definito come "trilemma della scalabilità", è emersa la necessità di sviluppare soluzioni di secondo livello.

L'idea alla base è spostare la maggior parte delle transazioni "off-chain", ovvero al di fuori della blockchain principale, utilizzando quest'ultima non più come registro di ogni singola attività, ma come giudice finale e imparziale.

Il Lightning Network rappresenta la proposta più promettente e diffusa. Si tratta di una rete di canali di pagamento bidirezionali che permette agli utenti di scambiare fondi istantaneamente e con commissioni minime, aggiornando i saldi privatamente e trasmettendo alla blockchain solo le transazioni di apertura e chiusura del canale. Inoltre, attraverso l'uso di appositi contratti, il Lightning Network consente di instradare pagamenti attraverso nodi intermedi in modo sicuro e senza che questi possano sottrarre i fondi.

Malgrado ciò, come ogni sistema complesso, anche il Lightning Network presenta sfide architetturali. Se da un lato risolve il problema del throughput, dall'altro introduce nuove dinamiche legate alla liquidità, alla topologia della rete e a potenziali rischi di centralizzazione, dove pochi nodi potrebbero arrivare a gestire una porzione significativa del traffico.

Struttura della tesi

La struttura dell'elaborato sarà quindi la seguente:

- Il **Capitolo 1** analizza le limitazioni strutturali delle blockchain pubbliche e il trilemma della scalabilità, ponendo le basi per la necessità di soluzioni off-chain.

- Il **Capitolo 2** introduce le primitive crittografiche essenziali per i protocolli di secondo livello, come le funzioni di hash, le firme digitali e i timelocks.

- Il **Capitolo 3** esamina il ruolo di Bitcoin come livello base, focalizzandosi su come la blockchain garantisca la sicurezza dei fondi gestiti nei livelli superiori.

- Il **Capitolo 4** descrive nel dettaglio l'architettura dei canali di pagamento, spiegando i meccanismi di apertura, aggiornamento e chiusura, nonché le strategie per prevenire frodi tra le controparti.

- Il **Capitolo 5** estende l'analisi dal singolo canale all'intera rete Lightning, illustrando come avviene il routing dei pagamenti attraverso nodi intermedi preservando la privacy e la sicurezza.
- Infine, il **Capitolo 6** discute i limiti attuali della tecnologia, presentando un'analisi critica sulla centralizzazione della rete e sulle vulnerabilità sistemiche, delineando le possibili linee di ricerca futura.

Capitolo 1

Scalabilità delle blockchain e limiti strutturali

Le blockchain pubbliche, e Bitcoin in particolare, hanno dimostrato che è possibile ottenere consenso distribuito in un ambiente privo di fiducia reciproca. Esse hanno introdotto un grande cambio di paradigma nella gestione del valore digitale. Tuttavia, questa nuova soluzione architetturale comporta costi significativi in termini di efficienza.

In questo capitolo verranno analizzati i principi fondamentali che garantiscono la sicurezza di tali sistemi, ponendo l'attenzione a come gli stessi meccanismi di ridondanza e verifica diffusa, necessari per la decentralizzazione, ne limitino strutturalmente la scalabilità. Verrà presentato il cosiddetto "Trilemma della Scalabilità" e si osserverà come un'adozione globale dell'approccio on-chain renda necessaria l'introduzione di soluzioni di secondo livello, per colpa delle limitazioni fisiche di banda e della latenza.

1.1 Evoluzione e principi fondamentali delle criptovalute

Ad oggi, la maggior parte delle transazioni su Internet, e la totalità di esse prima del 2008, avviene tramite l'ausilio di terze entità fidate, quali banche o istituti finanziari, su cui ogni nodo della rete pone la propria fiducia. Sebbene questo

modello sia ampiamente funzionante, presenta tutte le debolezze tipiche dei modelli **trust-based**. Le transazioni infatti non possono essere irreversibili perché gli istituti finanziari devono poter gestire eventuali dispute tra utenti, ma questo rende il denaro elettronico meno *finale* rispetto a quello fisico e crea incertezza per i venditori. Gestire problemi, rimborsi e controlli antifrode genera dei costi aggiuntivi che ogni utente paga sotto forma di *commissioni* rendendo i micropagamenti economicamente sconvenienti. Infine la presenza di un ente centrale rappresenta un **single point of failure**, ossia un punto critico la cui compromissione può compromettere l'intero sistema. [Nak08]

Lo scopo principale delle criptovalute è invece quello di operare in un ambiente decentralizzato, senza alcuna autorità centrale, basato sulla crittografia, nel quale due nodi della rete possono effettuare delle transazioni tra loro senza la necessità di fidarsi di un intermediario. Le transazioni sono computazionalmente impossibili da invertire e il sistema **blockchain** protegge dalle frodi sia venditori che i clienti. La validità di ogni operazione è garantita tramite calcoli matematici complessi e può essere verificata da chiunque in qualsiasi momento. La crittografia sostituisce quindi la fiducia verso gli intermediari con una prova matematica, non è necessario fidarsi di banche o istituti finanziari perché la correttezza di ogni operazione è verificabile pubblicamente.[Nak08]

La **blockchain** è una combinazione di elementi crittografici di base come le **funzioni di hash**, le **firme digitali** e i protocolli di **consenso distribuito** (*proof-of-work*), che interagiscono tra loro per creare un sistema sicuro di registrazione delle transazioni.[Nak08] Come evidenziato da Narayanan e Clark [NC17], l'innovazione di Nakamoto non risiede nell'invenzione di nuove primitive crittografiche, quanto nell'aver combinato in modo ingegnoso strumenti preesistenti, con un sistema di incentivi economici, creando il primo sistema realmente funzionante e autonomo.

Il cuore della blockchain su cui si basa Bitcoin è il **protocollo di gossip**, in cui tutte le modifiche che vengono apportate al registro delle transazioni vengono inviate in broadcast a tutti i nodi partecipanti alla rete. A causa di questo la blockchain di Bitcoin da sola non è in grado di gestire tutte le transazioni che avvengono quotidianamente nel mercato mondiale. Visa, uno dei principali circuiti di pagamento mondiali, utilizzato in oltre duecento paesi, è in grado di gestire

picchi di 47000 transazioni al secondo, arrivando quindi ad una media di centinaia di milioni di transazioni al giorno[Als21]. Bitcoin, allo stato originale, è in grado di gestire solamente 7 transazioni al secondo, con blocchi di dimensione massima di 1 megabyte. Assumendo di poter avere i blocchi di dimensione infinita, e supponendo di utilizzare circa 300 bytes per ogni transazioni, per attuare la mole di lavoro gestita da Visa, Bitcoin genererebbe blocchi da 8 gigabytes ogni 10 minuti, arrivando quindi a 400 terabyte di dati all'anno.[PD16]

Chiaramente nessun personal computer sarebbe in grado di gestire tale catena, né a livello di banda, né a livello di quantità di dati da elaborare. Se Bitcoin dovesse diventare il metodo più diffuso di pagamento online, la rete collasserebbe molto velocemente, oppure si creerebbe un fenomeno di estrema centralizzazione, in cui a gestire la rete sono solo i pochi nodi in grado di gestire quel volume di lavoro. Meno nodi gestiscono la rete e meno sarà accurato il registro delle transazioni, inoltre tali nodi potrebbero smettere di agire nell'interesse comune ed iniziare ad agire per il proprio interesse, aumentando i costi di transazione per poter operare scorrettamente. In casi estremi, i nodi non in grado di fare da miner, potrebbero affidare i loro fondi ai nodi più potenti, dandone loro la completa custodia.[PD16]

Diventa quindi fondamentale che, nel caso in cui Bitcoin diventi estremamente popolare, ogni personal computer, con una normale capacità di calcolo, sia in grado di competere per validare i vari blocchi, risolvendo le criticità che rendono possibile la centralizzazione, e dando modo alle persone di fidarsi di tale sistema.

1.2 Il trilemma della scalabilità nei sistemi decentralizzati

Con l'aumentare delle transazioni, le blockchain pubbliche diventano lente e costose, smettendo di essere un sistema affidabile. La sfida che si sta affrontando è quella di migliorare questi meccanismi senza però dover sacrificare nessuno dei principi fondamentali delle blockchain.

Il **Trilemma della scalabilità** è un teorema coniato da Vitalik Buterin, cofondatore di Ethereum, che raggruppa i problemi principali a cui gli sviluppatori vanno incontro quando cercano di realizzare una nuova blockchain. Tali problemi

sono tre e sono *scalabilità*, *decentralizzazione* e *sicurezza*. Buterin afferma che queste tre siano le proprietà che una qualsiasi blockchain cerca di avere, e, affidandosi a tecniche semplici, sia possibile avere solo due di esse.[But21]

Tali proprietà sono così definite:

- **Scalabilità:** la blockchain deve essere in grado di elaborare un numero di transazioni maggiore rispetto a quelle che può verificare da solo un normale nodo.

- **Decentralizzazione:** il sistema deve essere *trustless*, non ci deve essere nessun nodo, o nessuno piccolo gruppo di nodi, in cui sia necessario porre la fiducia di ogni elemento della rete. Tale proprietà implica che il livello di fiducia tra i nodi debba essere ridotto al minimo.

- **Sicurezza:** la blockchain deve riuscire a resistere al maggior numero possibile di nodi, partecipanti alla rete, che cercano di attaccarla. Idealmente tale resistenza dovrebbe arrivare almeno al 50% dei nodi, anche se una catena è considerata valida da sopra il 25%.

La scalabilità è strettamente collegata con la decentralizzazione, ed è il principio cardine sul quale si basano le blockchain, nonostante sia molto complicato ottenerlo.

Mescolando le proprietà si ottengono tre diverse tipologie di sistemi, ognuno dei quali è costretto a sacrificare un principio, proprio come detto nel trilemma della scalabilità:

- **Blockchain tradizionali:** si basano sul fatto che ogni partecipante alla rete sia un nodo a se stante in grado di validare ogni transazione. In questo modo si ottengono decentralizzazione e sicurezza, ma non la scalabilità. Di questa categoria fanno parte Bitcoin, Litecoin e altre catene simili.

- **Catene high-TPS:** si basano su un numero limitato di nodi, tipicamente tra 10 e 100, che mantengono il consenso tra di loro, gli altri nodi della rete devono fidarsi della maggior parte di questi nodi. Questa soluzione ottiene scalabilità e sicurezza, ma è chiaro che non sia decentralizzata.

- **Ecosistemi multi-chain:** si basa sul concetto di *scalabilità orizzontale*, ossia la possibilità di avere più catene e di utilizzare un protocollo di comunicazione per farle comunicare tra loro. Questa soluzione risulta essere decentralizzata e scalabile, ma non sicura. Infatti ad un aggressore basterebbe ottenere la maggioranza dei nodi di consenso di una delle numerose catene per prenderne il controllo e generare problemi a valanga sulle altre.

[But21]

La conclusione è che cercare di ottenere scalabilità a livello del protocollo di base, senza compromettere la decentralizzazione e la sicurezza, è incredibilmente difficile con l'architettura attuale. Bitcoin ha scelto di non scendere a compromessi sulla sicurezza e sulla decentralizzazione; di conseguenza, la scalabilità deve essere ricercata altrove.

1.3 Off-chain e soluzioni di secondo livello

Considerando le criticità evidenziate dal *Trilemma della scalabilità* enunciato precedentemente, la comunità scientifica e gli sviluppatori di Bitcoin hanno spostato la loro attenzione verso soluzioni che non richiedono modifiche al protocollo di base, detto *layer 1*. L'idea è quella di fare in modo che la maggior parte delle transazioni avvenga **off-chain**, ovvero al di fuori della blockchain principale, andando di fatto a costruire un livello superiore in grado di gestire tali transazioni, detto *layer 2*.

Poon e Dryja [PD16] introducono la possibilità di creare canali di pagamento tra due nodi partecipanti alla rete, che si scambiano denaro senza la necessità di registrare ogni transazione sulla blockchain. Se, per esempio, due nodi si scambiano denaro ogni giorno, e solo a loro interessa di queste transazioni ricorrenti, non è necessario che anche tutti gli altri nodi della catena siano sempre a conoscenza di ogni loro movimento. Così facendo si permette agli utenti di Bitcoin di attuare molteplici transazioni senza far collassare il sistema e senza rischiare di avere problemi di centralizzazione.

La soluzione consiste quindi nel creare questi canali, detti **canali di micropagamento**, che consentono di inviare grandi quantità di fondi in modo decentraliz-

zato. In questo modo Bitcoin può scalare fino a miliardi di transazioni giornaliere utilizzando la potenza di calcolo di un normale laptop odierno.

I canali di micropagamento instaurano una relazione tra due parti, che aggiornano costantemente i propri saldi, evitando di passare sempre per la blockchain e registrando su di essa solo la transazione finale, figlia di tutte quelle precedenti avvenute off-chain. Questi canali non sono da intendere come reti separate da Bitcoin, ma rappresentano vere e proprie transazioni Bitcoin. Scegliere di inviare sulla blockchain principale un'unica transazione finale permette ad entrambe le parti di garantire il proprio saldo sulla rete.

Il funzionamento logico di base di questo meccanismo è il seguente:

1. **Apertura:** Due nodi bloccano una somma di denaro comune in una transazione sulla blockchain. Questo è l'unico momento in cui la rete globale viene coinvolta inizialmente. Questa transazione di apertura viene chiamata *Funding Transaction*.

2. **Operatività Off-Chain:** Le parti possono scambiarsi un numero illimitato di transazioni aggiornando privatamente la ripartizione dei fondi bloccati. Queste operazioni sono istantanee e non richiedono commissioni di mining, poiché sono semplici scambi di dati firmati crittograficamente tra i due utenti.

3. **Chiusura:** Quando le parti decidono di terminare la collaborazione, l'ultimo stato del bilancio viene inviato alla blockchain per la transazione finale. Non è necessario che entrambe i nodi richiedano di chiudere il canale, basta solo una delle due parti.

In questo modello, la blockchain non agisce più come un registro contabile di ogni singola transazione, ma assume il ruolo di un giudice imparziale che interviene solo in caso di disputa o alla fine del rapporto economico tra le parti.

Capitolo 2

Primitive crittografiche per i protocolli off-chain

La sicurezza e il funzionamento dei protocolli di secondo livello non si basano sulla fiducia verso terze parti, ma sull'utilizzo combinato di specifiche primitive crittografiche. Mentre la blockchain di Bitcoin utilizza queste tecnologie per garantire il consenso globale, il Lightning Network le impiega per creare contratti intelligenti, detti *Smart Contracts*, che permettono l'esecuzione sicura delle transazioni off-chain.

In questo capitolo verranno analizzati gli strumenti fondamentali necessari alla costruzione di un canale di pagamento: le *funzioni di hash* per la creazione di vincoli condizionali, le *firme digitali* per la gestione condivisa dei fondi e i *vincoli temporali* per la gestione delle dispute.

2.1 Funzioni di hash crittografico e loro proprietà

Le funzioni di hash svolgono un ruolo fondamentale nella crittografia moderna e hanno numerose applicazioni nell'ambito della sicurezza informatica. Sono infatti utilizzate per esempio per:

- Firme digitali
- Verifica dell'integrità dei file

- Sicurezza delle password
- Blockchain
- SSL/TLS protocolli

Una funzione di hash è una funzione **non invertibile** che trasforma una sequenza di bit di lunghezza arbitraria in una sequenza di bit di lunghezza predefinita, chiamata **digest**.

Le funzioni di hash specifiche per gli ambiti crittografici devono avere le seguenti proprietà:

- **Determinismo** → lo stesso input inserito nella stessa funzione hash deve dare come risultato lo stesso output
- **Efficienza** → il valore hash deve venir calcolato rapidamente, indipendentemente dalla dimensione dell'input
- **Non invertibilità** → dato un digest non deve essere possibile risalire all'input che lo ha generato
- **Effetto valanga** → piccole modifiche nel messaggio di input determinano modifiche significative, apparentemente non correlate, nell'hash di output

[Sta17]

Ognuna di queste caratteristiche è essenziale per rendere la funzione crittograficamente sicura. La problematica principale a cui queste funzioni devono essere in grado di resistere sono le **collisioni**. Avviene una collisione ogni volta che due input differenti generano lo stesso output. Di base, se un algoritmo genera delle collisioni significa che non è un algoritmo sicuro. [Sta17]

La famiglia di funzioni crittografiche di hash più utilizzata ad oggi è la **SHA**, *Secure Hash Algorithm*, i vari algoritmi che ne fanno parte si distinguono per differenza di lunghezza del *digest* e resistenza alle collisioni:

- **SHA-1** → è stato l'algoritmo più usato della famiglia sha, diffuso in numerose applicazioni e protocolli nonostante ormai sia considerato insicuro, dato che la sua sicurezza è stata compromessa dai crittoanalisti. Produce un *digest* di 160 bit, da un input lungo massimo $2^{64} - 1$ bit.

- **SHA-2** → Nel 2001 vengono pubblicate dal NIST altre quattro funzioni di hash, ognuna con un *digest* più lungo di quello originale. A far parte di questo sottogruppo sono **SHA-224**, **SHA-256**, **SHA-384** e **SHA-512**, che generano *digest* rispettivamente di 224, 256, 384 e 512 bit. Gli algoritmi **SHA-256** e **SHA-512** lavorano rispettivamente con *word* di 32 e 64 bit. La loro struttura è sostanzialmente identica anche se utilizzano un differente numero di rotazioni e di costanti addizionali. Gli altri due algoritmi, invece, sono semplicemente delle versioni troncate di questi, con hash calcolati con differenti valori iniziali. La sicurezza della famiglia SHA-2 è oggi considerata solida; tuttavia, la sua struttura è stata meno analizzata rispetto a SHA-1 prima della sua compromissione, motivo per cui il NIST ha promosso lo sviluppo di SHA-3.

- **SHA-3** → Nuovo membro della famiglia SHA trovato grazie ad un competizione lanciata dal NIST nel 2007, con l'obiettivo di sviluppare una nuova funzione di hashing per rafforzare le versioni precedenti.

[Sta17]

L'algoritmo SHA-256 riveste un ruolo centrale nel protocollo Bitcoin: viene infatti utilizzato per calcolare l'hash di ciascun blocco, per collegare i blocchi della catena tra loro e, soprattutto, nel meccanismo di **Proof-of-Work**, dove i miner devono trovare un valore di nonce tale che l'hash del blocco risulti inferiore a una soglia prefissata, detta *target*. Quindi è anche grazie a questo algoritmo se il sistema riesce a resistere alle modifiche retroattive e a garantire l'integrità della blockchain.

Nel contesto delle criptovalute, in particolare Bitcoin, ogni transazione e ogni blocco della blockchain vengono identificati da un hash calcolato applicando due volte la funzione di SHA-256, il processo si chiama **double hashing**. Questa doppia applicazione serve a ridurre la probabilità di collisioni e proteggere da eventuali attacchi. [Ant17]

Nel contesto specifico del Lightning Network, tuttavia, l'uso più rilevante delle funzioni di hash non è legato al mining o alla struttura a blocchi, ma alla creazione di vincoli condizionali per i pagamenti.

La proprietà di *resistenza alla preimmagine* viene sfruttata per generare segreti crittografici fondamentali per il routing sicuro. Il meccanismo, che costituisce la base degli **Hashed Time-Lock Contracts (HTLC)**, funziona nel seguente modo:

1. Il destinatario di un pagamento genera un numero casuale R, detto *pre-image* o segreto.
2. Viene calcolato l'hash di questo segreto: $H = \text{SHA-256}(R)$, denominato *payment hash*.
3. L'hash H viene condiviso pubblicamente con il mittente e con la rete, mentre R rimane segreto fino al momento dell'incasso.

I fondi vengono vincolati a H tramite uno smart contract: essi possono essere spesi solo da chi è in grado di rivelare la preimmagine R che, passata attraverso la funzione SHA-256, restituisce l'hash H specificato nel contratto. Data l'unidirezionalità della funzione, è computazionalmente impossibile per un attaccante risalire a R conoscendo solo H, garantendo così che solo il legittimo destinatario possa sbloccare i fondi [Ant17].

2.2 Firme digitali e firme multiple

2.2.1 Firme digitali

Le firme digitali vengono create per permettere ai sistemi di elaborazione di accertare l'identità di un utente. Il destinatario di un messaggio, o di una somma di denaro, deve poter essere in grado di verificare l'identità del mittente e l'integrità di ciò che ha ricevuto, dunque il metodo utilizzato deve rendere difficile ad un intruso di spacciarsi per qualcun altro o di modificare i messaggi inviati.

La firma digitale entra in gioco proprio nei sistemi *trustless*, quando il mittente e il destinatario non si fidano l'uno dell'altro. Il meccanismo, per essere valido, deve rispettare i seguenti tre requisiti:

1. **Non ripudio**→ non deve essere permesso ad un mittente di negare di aver mandato un messaggio da lui firmato

2. **Autenticazione e integrità**→ il destinatario deve poter autenticare il messaggio, ossia accertare l'identità del mittente e l'integrità del messaggio ricevuto

3. **Irrefutabilità della ricezione**→ il destinatario non deve poter sostenere di aver ricevuto un messaggio diverso da quello inviatogli dal mittente

[Sta17]

Il meccanismo crittografico adottato da Bitcoin si chiama **Elliptic Curve Digital Signature Algorithm**, ossia **ECDSA**. Con esso vengono firmate digitalmente le transazioni ed ogni nodo può dimostrare il possesso della propria chiave privata, senza però andare a rivelarla. Le chiavi private in Bitcoin sono molto importanti, perchè è con esse che si dimostra il possesso di una certa quantità di monete, se la chiave viene persa anche i fondi a lei associati saranno perduti, mentre se viene rubata consente di spendere il denaro dell'utente.

ECDSA si basa sulla crittografia a **curve ellittiche**, una variante della crittografia asimmetrica che riesce ad ottenere lo stesso livello di sicurezza di algoritmi come RSA utilizzando chiavi molto più piccole, risultando quindi più efficiente. [Ant17]

La **chiave privata** di ogni utente viene scelta randomicamente in un intervallo definito, e la **chiave pubblica** viene derivata tramite un'operazione matematica detta **moltiplicazione scalare** su una curva ellittica. La curva adottata da Bitcoin è la **secp256k1**, che opera su un campo finito di 256 bit, scelta per la sua semplicità ed efficienza. [Cer10]

La sicurezza del sistema si basa sulla difficoltà computazionale di risalire alla chiave privata di determinato utente conoscendo solo quella pubblica, ossia risolvere il **problema del logaritmo discreto su curve ellittiche**.

Durante una firma, l'algoritmo genera una coppia di valori partendo dal messaggio e dalla chiave privata del nodo in questione, questi due numeri costituiscono quindi la firma digitale. Chiunque può verificarne la veridicità utilizzando la chiave pubblica del mittente.

ECDSA garantisce l'autenticità e la non ripudiabilità delle transazioni, rendendo possibile un sistema di scambio decentralizzato.

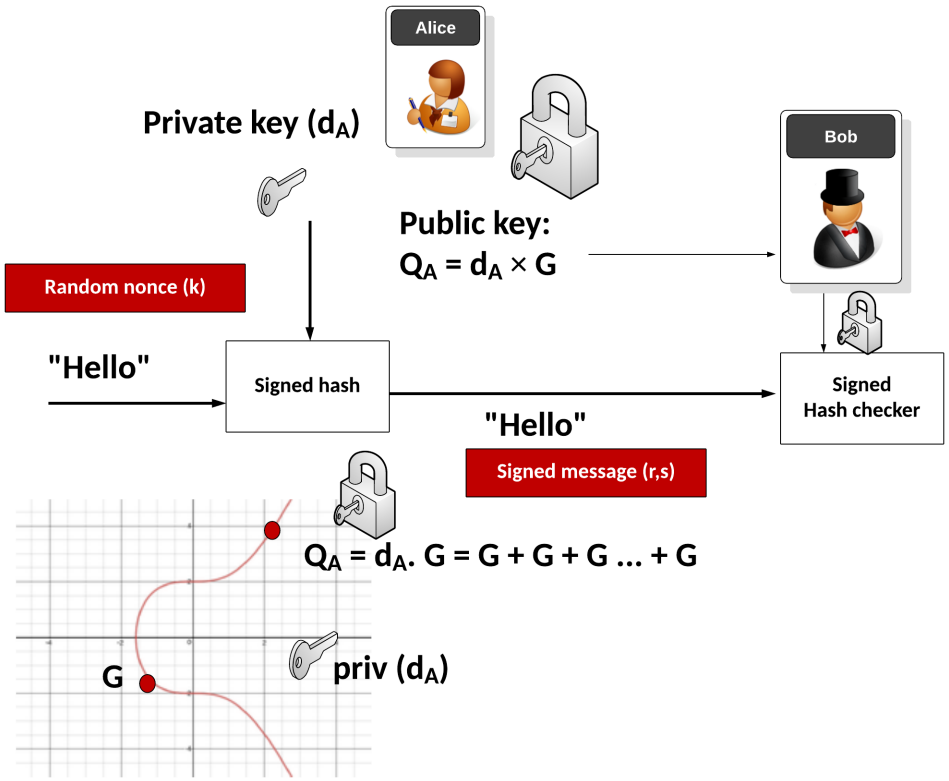

Figura 2.1: Schema logico della firma digitale ECDSA. La chiave privata firma l'hash del messaggio; la chiave pubblica permette la verifica senza rivelare il segreto. (Fonte: rielaborazione da Scrypt Platform)

2.2.2 Firme multiple

Per implementare un sistema complesso come il Lightning Network richiede meccanismi avanzati e di conseguenza le firme singole non bastano più. Per gestire la proprietà condivisa dei canali sono richieste le **firme multiple**, o **multisig**, che permettono di vincolare i fondi ad un gruppo di persone.

In questa tipologia di schemi viene impostata una condizione per cui nello script di output vengono registrate k chiavi pubbliche ed almeno t di queste devono fornire una firma valida per spendere i fondi. Schemi come questo vengono definiti come t-of-k. Per esempio in una firma multipla 2-of-3 ci sono tre potenziali chiavi pubbliche, ed almeno due di queste devono essere usate per creare una firma valida per spendere i fondi. [Ant17]

Nel contesto del Lightning Network, viene utilizzato uno schema 2-of-2. Quando due utenti aprono un canale di pagamento, creano una transazione di finanziamento, la *Funding Transaction*, verso un indirizzo multisig controllato congiuntamente. Dal momento in cui il canale è stato creato, la regola imposta dalla blockchain è semplice e stretta: per muovere quei fondi è necessario il consenso di entrambe le parti, questo si traduce nella necessità che entrambe le parti firmino le modifiche allo stato dei fondi.

Questa architettura agisce come una garanzia di sicurezza preventiva:

- Nessuna delle parti può sottrarre i fondi all'altra, poiché mancherebbe la seconda firma necessaria.
- In caso di disaccordo, i fondi rimangono bloccati.

In realtà, per quanto riguarda questo ultimo punto, nel protocollo sono previsti meccanismi di uscita che permettono di risolvere la disputa, come i *timelock*, che verranno descritti successivamente. [PD16]

In sintesi, il Multisig rende i fondi vincolati in un canale condivisi, costringendo le parti a cooperare per aggiornare lo stato del canale.

2.3 Vincoli temporali nelle transazioni

Nei sistemi di pagamento tradizionali l'obiettivo è rendere il denaro derivante da una transazione immediatamente spendibile; nei sistemi complessi come il Lightning Network, invece, è necessario che i fondi derivati da una transazione non siano spendibili immediatamente, questo per evitare truffe e frodi.

Verrà chiarito in seguito come il non permettere di spendere subito i fondi derivanti da una transazioni sia una misura di sicurezza che permette alla controparte del canale di realizzare di aver subito una truffa e di agire di conseguenza.

Diventa quindi necessario gestire il tempo all'interno delle transazioni. Quest'ultimo nella blockchain di Bitcoin non è misurato prevalentemente in secondi, ma in **altezza di blocco**, *Block Height*, sebbene sia possibile utilizzare anche il timestamp Unix. Poiché un blocco viene minato in media ogni 10 minuti, definire un intervallo di 144 blocchi equivale a circa 24 ore.

Entrano quindi in gioco i **timelock**. Ne esistono due tipi:

- **Timelock assoluti:** rendono l'output di una transazione spendibile solo dopo un determinato punto nel futuro, ad esempio dopo il blocco 850.000. A livello di script, questi sono implementati tramite l'opcode `OP_CHECKLOCKTIMEVERIFY` (CLTV), introdotto con il BIP 65 [Tod14]. Questi vengono utilizzati per i **rimborsi**. Come detto precedentemente, quando due nodi aprono un canale bloccano dei fondi con la *Funding Transaction*, questa rende tali fondi comuni ed impone la necessità della firma di entrambe le parti per fare un qualsiasi movimento. Se un nodo, una volta creato il canale, smette di rispondere, o va offline perennemente, l'altro, dopo un determinato periodo, può recuperare i fondi che aveva bloccato grazie a una transazione che diviene valida solo allo scadere del timelock. [Ant17]

- **Timelock relativi:** introdotti con il BIP 112 [FBL15] tramite l'opcode `OP_CHECKSEQUENCEVERIFY` (CSV), essi non guardano una data fissa nel calendario, ma l'età dell'output che si sta spendendo. Questo tipo di timelock impone la seguente condizione: una transazione X che spende un output Y è valida se e solo se sono passati n blocchi dalla conferma di Y sulla blockchain. Vengono utilizzati nel caso di **dispute**, e sono l'elemento centrale del meccanismo di sicurezza del Lightning Network. Tenendo bloccati i fondi per un certo numero di blocchi, un'eventuale vittima può accorgersi di essere stata truffata e creare una finestra di contestazione. La vittima ha tempo a sufficienza per accorgersi del tentativo di frode e, utilizzando una chiave di revoca, prelevare l'intera somma prima che il timelock scada e il truffatore possa incassare. [Ant17]

Capitolo 3

Bitcoin come livello base di sicurezza

Nel capitolo precedente sono state analizzate le primitive crittografiche necessarie per la costruzione di contratti condizionali e canali di pagamento. Tuttavia, per ottenere un sistema funzionante, sicuro e soprattutto *trustless*, gli elementi appena trattati, da soli, non bastano. Sorge quindi la necessità di un'infrastruttura sottostante, in grado di interpretarli ed eseguirli in modo inappellabile.

Il protocollo originale di Bitcoin, a questo punto, non è più utilizzato solo come valuta digitale, ma assume un ruolo cruciale, diventando un arbitro imparziale per le *dispute*. Come teorizzato nel whitepaper di Satoshi Nakamoto [Nak08], la blockchain risolve il problema della doppia spesa attraverso una catena di prove crittografiche basate sul lavoro computazionale. Per le soluzioni di secondo livello come il Lightning Network, questa proprietà di immutabilità è fondamentale: la blockchain principale diventa l'ente a cui le parti si rivolgono solo in caso di controversia o per finalizzare gli scambi avvenuti privatamente.

Come detto nei capitoli precedenti, è importante sottolineare che le transazioni eseguite nel Lightning Network non sono alternative a Bitcoin, ma sono vere e proprie transazioni Bitcoin valide, la differenza è che queste vengono trattenute dai nodi e non sono trasmesse immediatamente alla rete globale. La sicurezza di questi scambi *off-chain* deriva interamente dalla certezza che, in qualsiasi momento, lo stato corrente del canale possa essere pubblicato e confermato sulla blockchain

principale.

Questo capitolo analizzerà le componenti strutturali di Bitcoin che rendono possibile tale architettura. In primo luogo, verrà esaminata la struttura delle transazioni, successivamente, si discuterà il ruolo della Proof-of-Work non solo come meccanismo di consenso, ma come orologio universale per la gestione dei vincoli temporali. Infine, verrà trattata l'evoluzione del protocollo con l'introduzione di Segregated Witness (SegWit), un aggiornamento critico che, risolvendo la vulnerabilità della *transaction malleability*, ha reso possibile l'implementazione sicura ed efficiente dei canali di pagamento moderni.

3.1 Struttura delle transazioni

Le **transazioni** permettono ai nodi di scambiarsi i Bitcoin [Con19], la possibilità di attuare tali operazioni è un aspetto fondamentale per una moneta elettronica. Senza transazioni i vari proprietari di monete non potrebbero scambiarsele tra loro e si andrebbe dunque a perdere il concetto di moneta stessa.

Ogni utente possessore di bitcoin può trasferire una certa quantità di valuta ad un altro utente andando ad apporre la propria firma digitale sull'hash della transazione precedente e la chiave pubblica del nuovo proprietario.

Sorge però un problema: un qualsiasi utente non può verificare che un possessore di bitcoin che ha appena effettuato una transazione non abbia già speso quelle monete altre volte, ossia non abbia effettuato un **double-spending**. Di fatto il denaro digitale può essere duplicato senza alcuna fatica, proprio come qualsiasi altro oggetto digitale.[Con19]

Bisogna quindi trovare un modo per dimostrare ai beneficiari di una transazione che le monete che stanno ricevendo non sono già state spese. Il sistema di moneta digitale introdotto da David Chaum nel 1983, noto come eCash [Cha83], è il primo a risolvere il problema della doppia spesa, ma in un ambiente centralizzato, di conseguenza Bitcoin è il primo metodo di pagamento digitale in grado di risolvere questo problema in maniera totalmente decentralizzata.

Per fare ciò tutti i nodi della rete devono essere a conoscenza di tutte le transazioni che sono state eseguite fino a quel momento e devono essere tutti d'accordo

sull'ordine cronologico in cui sono avvenute, dunque tutte le transazioni devono essere pubbliche.

Per accordare ogni nodo sull'ordine cronologico si utilizza un **timestamp server** che prende in input un insieme di dati e ne calcola l'hash per poi pubblicarlo. Rendendo l'hash pubblico chiunque può verificare che quei dati dovevano già esistere al momento della pubblicazione. Ogni nuovo timestamp include anche l'hash del precedente in modo tale da creare una blockchain.

In sintesi il timestamp server è usato per creare una prova temporale immutabile dell'esistenza dei dati. [Nak08]

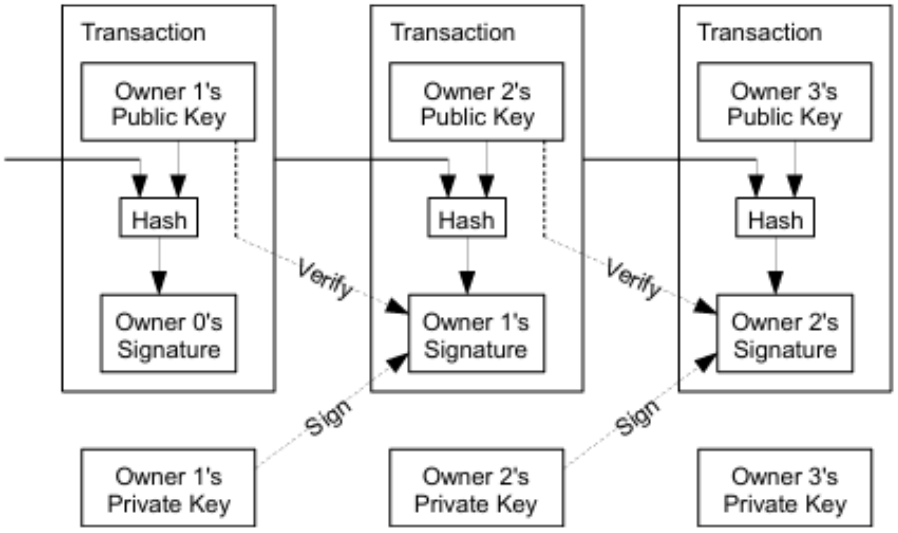

Figura 3.1: Struttura di una transazione Bitcoin composta da input e output. Ogni input fa riferimento a un output non speso di una transazione precedente. Fonte: Nakamoto[Nak08].

3.1.1 I satoshi

I **satoshi** sono l'unità più piccola utilizzabile nel protocollo originale di Bitcoin [Ant17]. La relazione che intercorre tra bitcoin e satoshi è la seguente:

$$1 \text{ sat} = 10^{-8} \text{ BTC} = 0,00000001 \text{ BTC} \tag{3.1}$$

L'evoluzione verso soluzioni di secondo livello, come il Lighting Network, ha reso necessaria una granularità ancora maggiore, introducento così i **millisatoshi**, o **msat**. questa suddivisione è fondamentale per permettere il calcolo di commissioni di routing estremamente basse [Lig16]. I *msat* seguono la relazione sottostante:

$$1 \text{ msat} = 10^{-3} \text{ sat} = 10^{-11} \text{ BTC} \tag{3.2}$$

3.1.2 Il modello UTXO

A differenza dei sistemi bancari tradizionali o di altre tipologie di monete virtuali, Bitcoin non utilizza un modello basato sui conti o sul portafoglio. Il software dà l'impressione che le monete vengano spostate da un portafoglio ad un altro, quando in realtà queste si muovono di transazione in transazione. Ognuna di esse spende i **satoshi** ricevuti, che sono gli output di una o più transazioni passate. L'output si ogni transazioni può essere suddiviso in più parti, per permettere di spedire i fondi a nodi differenti, ma ognuna di queste parti è e deve essere spendibile una sola volta [Ant17].

Nello specifico, ogni output è univocamente identificato tramite il **Transaction Identifier** (TXID), ovvero l'hash della transazione firmata da cui ha origine. Poiché ogni singolo output può essere speso una sola volta, l'insieme di tutti gli output registrati sulla blockchain può essere categorizzato in due insiemi disgiunti: gli output già spesi e gli output non ancora spesi, definiti **Unspent Transaction Outputs (UTXO)**. Una transazione risulta valida se e solo se essa utilizza esclusivamente UTXO esistenti come propri input.

Eccezion fatta per le transazioni di *coinbase*, ovvero le transazioni create dei miner per introdurre nuova moneta del sistema, il protocollo impone un rigoroso bilancio: se il valore totale degli output di una transazione eccede quello dei suoi input, la transazione viene considerata invalida e rigettata dalla rete. Al contrario, qualora il valore degli input sia superiore a quello degli output, la differenza viene interpretata come commissione di transazione, detta *transaction fee*, e può essere riscossa dal miner che include la transazione all'interno di un blocco.

Questo modello presenta proprietà fondamentali per i protocolli di secondo livello. In particolare:

- Ogni stato di un canale di pagamento può essere rappresentato come una possibile spesa futura degli UTXO bloccati nella transazione di funding.
- La spesa concorrente di uno stesso UTXO è impossibile per costruzione, eliminando alla radice il problema del double-spending.
- Gli UTXO permettono di vincolare fondi tramite script arbitrari, consentendo la realizzazione di contratti condizionali.

Come osservato da Poon e Dryja [PD16], la possibilità di bloccare e successivamente ridistribuire UTXO condivisi è il meccanismo alla base dei canali di pagamento.

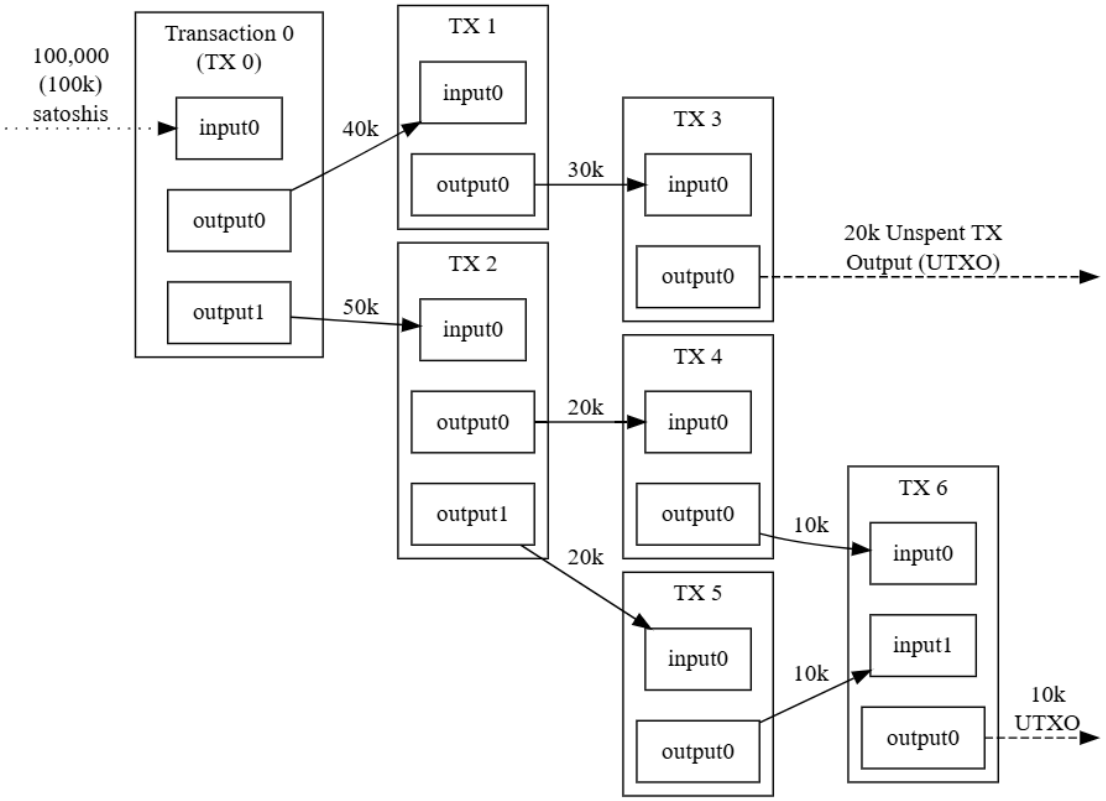

Figura 3.2: Propagazione e validazione di una transazione nella rete Bitcoin. Ogni nodo verifica la correttezza della transazione prima di propagarla ulteriormente. Fonte: Bitcoin Developer Documentation [Bit25].

3.1.3 Transazioni e script di spesa

In Bitcoin è importante distinguere tra la struttura di una transazione e le condizioni che ne regolano la spendibilità. Una transazione definisce quali UTXO vengono consumati e quali nuovi output vengono creati, ma non contiene logica esecutiva complessa. Le regole che determinano chi e quando può spendere un output sono invece espresse tramite **Bitcoin Script**, un linguaggio di scripting basato su uno stack, abbastanza limitato. [Ant17].

Questa scelta progettuale riduce la superficie di attacco del protocollo e garantisce che il comportamento delle transazioni sia deterministico e facilmente verificabile dai nodi della rete. Ogni output contiene uno script di blocco, lo *scriptPubKey*, mentre per spendere tale output è necessario fornire uno script di sblocco, *scriptSig*, che soddisfi le condizioni imposte.

Nel Lightning Network, Bitcoin Script viene utilizzato per costruire contratti condizionali che combinano firme multiple, vincoli temporali e segreti crittografici. Sebbene il linguaggio non permetta smart contract complessi, la sua semplicità consente di implementare canali di pagamento sicuri senza compromettere la decentralizzazione del layer base [PD16].

3.2 Struttura dei blocchi

Ogni blocco della blockchain di Bitcoin è costituito da due sezioni:

- **Block header** → l'intestazione del blocco.
- **Block body** → l'elenco delle transazioni appartenenti al blocco.

Le informazioni contenute nell'intestazione sono necessarie a collegare i vari blocchi tra loro e a verificarne la validità. Nella figura 3.3 possiamo vedere quali sono gli elementi che compongono l'header di un tipico blocco della blockchain di Bitcoin:

- **Version** → 4 bytes, indica la versione corrente del blocco e dunque le regole di validazione che esso sta seguendo.

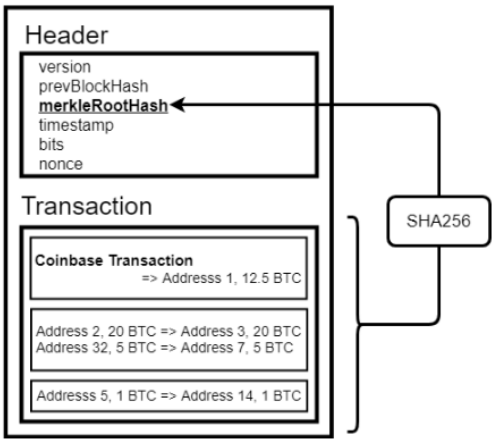

Figura 3.3: Struttura di un blocco Bitcoin. Le transazioni sono organizzate in un Merkle tree, il cui hash radice è incluso nell'header del blocco. Fonte: Provino, rielaborazione da [Pro20].

- **PrevHashBlock**→ 32 bytes, rappresenta il doppio hash del header del blocco precedente, necessario per rendere tale struttura una catena. Garantisce che nessun blocco precedente possa essere modificato senza modificare anche l'intestazione di questo. Ovviamente l'unico blocco che non ha un predecessore è il primo, che viene chiamato **genesis block**.

- **MerkleRootHash** → 32 bytes, è il doppio hash della root dell'albero di Merkle composto da tutte le transazioni appartenenti al blocco in questione. Modificare una transazione significa anche modificare questo elemento del blocco.

- **Timestamp** → 4 bytes, timestamp approssimato della creazione del blocco. Deve essere maggiore del timestamp medio degli 11 blocchi precedenti.

- **Bits/Target** → 4 bytes, codifica compatta della difficoltà richiesta per trovare un hash valido per risolvere il blocco.

- **Nonce** → 4 bytes, numero arbitrario trovato dai miners che rende l'hash del blocco valido, ossia inferiore al target. Se tutti i valori a 32 bit vengono tentati si può aggiornate il timestamp o la Merkle root.

I Merkle tree permettono di rappresentare in modo compatto l'insieme delle transazioni incluse in un blocco. Attraverso una struttura ad albero basata su funzioni hash, è possibile verificare l'appartenenza di una singola transazione al blocco senza dover scaricare l'intero contenuto, proprietà fondamentale per i nodi leggeri. [Pro20]

Per quanto riguarda invece il body del blocco si nota che il primo elemento all'interno dell'elenco delle transazioni e la cosiddetta **coinbase transaction**, ossia i nuovi bitcoin che sono stati generati e dati come ricompensa al miner che ha risolta il blocco in questione. Questo meccanismo, noto come **block reward**, rappresenta l'unico modo con cui vengono emessi nuovi bitcoin nel sistema. [Bit25]

La struttura dei blocchi e la loro concatenazione tramite Proof-of-Work forniscono una nozione condivisa di tempo e finalità, che viene sfruttata direttamente dai protocolli di secondo livello per gestire vincoli temporali e risolvere eventuali dispute.

3.3 Proof-of-Work: meccanismo di consenso e arbitraggio

Nel protocollo originiale di Nakamoto [Nak08], la **Proof of Work** è il meccanismo adottato per garantire la sicurezza dalla rete, impedire il *double spending* e confermare le transazioni attraverso un processo di competizione tra i miner che richiede un enorme potenza di calcolo e di energia. La funzione principale della PoW è quindi quella di rendere estremamente costoso e impraticabile per un malintenzionato manomettere la rete.

Per il lighting network questo meccanismo è fondamentale perché garantisce sicurezza e veridicità sulla blockchain, e la rende immutabile. Inoltre tale sistema viene anche utilizzato come orologio centrale, dato che esso genera un blocco ogni dieci minuti. [AOP22]

I **miner** sono computer, tipicamente dotati di molta potenza, che convalidano le transazioni risolvendo i vari blocchi ed aggiungendoli alla blockchain. Per fare ciò competono per risolvere problemi crittografici complessi e colui che risolve il problema per primo viene ricompensato con nuove unità di Bitcoin. [CGN14]

La **Proof of Work** consiste nel trovare un valore di *Nonce* che permetta all'hash del blocco di iniziare con un numero predefinito di zeri. Tale numero viene rappresentato dal valore *Target* nell'header di ogni blocco. La quantità di lavoro richiesta per risolvere il problema è esponenziale rispetto al numero di zeri richiesti, ma, una volta trovata la soluzione, ogni nodo della rete può verificare con facilità la validità del blocco. [Nak08]

Ogni miner deve dunque risolvere il seguente problema:

$$H = \mathrm{SHA256}\big(\mathrm{SHA256}(\mathrm{BlockHeader}(N))\big)$$

Trovare N (nonce) tale che $H < T$ (target)[CGN14]

Come detto nei capitoli precedenti una delle caratteristiche che rende così gettonate le funzioni di hash è l'impossibilità di risalire all'input conoscendo il risultato. È proprio questa la proprietà su cui si basa la Proof of Work, infatti, quello di trovare un double hash per il blocco che contenga un determinato numero di zeri iniziali è un problema che può essere risolto solo a tentativi. Nessun miner è in grado di costruire artificialmente un *nonce* in grado di risolvere il blocco. Una volta che la soluzione è stata trovata però tutti i nodi sono in grado di calcolare l'hash del blocco e verificare che esso rispetti il numero di zeri richiesto. [Mil17]

La **PoW** risolve anche un altro problema fondamentale: quello di determinare in modo affidabile la rappresentanza nella decisione di maggioranza. La fiducia nella **blockchain** si basa sul **consenso distribuito**, infatti un nuovo blocco può essere aggiunto alla catena solo quando la maggioranza dei partecipanti della rete concorda sulla sua validità. Se per rappresentare la maggioranza venisse utilizzato un meccanismo *one-IPaddress-one-vote* il sistema sarebbe nella mani di chi è in grado di allocare in maggior numero di indirizzi IP. Invece la PoW si basa su un meccanismo ***one-CPU-one-vote***, quindi la catena considerata valida è la catena più lunga, ovvero quella che incorpora la maggior quantità di lavoro computazionale. Se la maggioranza della potenza delle CPU è rappresentata da nodi onesti vuol dire che la loro catena crescerà più velocemente rispetto a catene fasulle generate da nodi non onesti. Per modificare un blocco passato della catena corrente un malintenzionato dovrebbe ricalcolare la PoW di tutti i blocchi successivi e riuscire anche a superare la crescita della blockchain originale. [Nak08]

Per compensare all'aumento di velocità, la potenza hardware e le variazioni nel numero di nodi attivi nel tempo, la **difficoltà** della proof-of-work viene regolata tramite una **media mobile** che mantiene costante il numero medio di blocchi generati per ora. Se i nuovi blocchi vengono prodotti troppo rapidamente, la difficoltà verrà aumentata. Essa rappresenta il *numero di zeri iniziali richiesti nel doppio hash* e questo valore è memorizzato nel campo **nBits/Target**.

Ogni nodo conosce la difficoltà corrente leggendone il valore dall'header dell'ultimo blocco valido della blockchain. Tipicamente la difficoltà viene aggiornata ogni 2016 blocchi, per mantenere una media costante di un blocco ogni 10 minuti, fondamentale per il Lightning Network. [CGN14]

Il campo **nBits/Target**, presente nell'header di ciascun blocco, rappresenta una *codifica compatta* del valore di **target**, ovvero la soglia numerica a cui l'hash doppio SHA-256 del blocco deve risultare inferiore o uguale per essere considerato valido. Il valore `nBits` è composto da:

- una **mantissa** (m), che occupa i 3 byte meno significativi,
- un **esponente** (e), rappresentato dal byte più significativo.

Il target effettivo si ottiene espandendo tale valore compatto secondo la formula:

$$\text{target} = m \times 2^{8(e-3)}.$$

Più piccolo è il valore del target, maggiore sarà la difficoltà richiesta per trovare un hash valido, questo perché un target più basso richiede che l'hash si trovi in una porzione sempre più piccola dello spazio di output di SHA-256 [CGN14].

La **Proof-of-Work** è un meccanismo di sicurezza robusto ma poco efficiente, che ha garantito la sopravvivenza e l'integrità di Bitcoin a scapito della sua sostenibilità [PD16]. Di seguito verranno trattate le principali problematiche:

Consumo energetico Il limite principale della Proof-of-Work è l'elevato consumo energetico. Il mining richiede una potenza di calcolo sempre maggiore per mantenere il tempo medio di generazione dei blocchi, comportando enormi sprechi di energia elettrica. Col tempo questo ha sollevato preoccupazioni ambientali e portato alla ricerca di modelli e strutture alternative. Di fatti l'**hash rate** di

Bitcoin è dell'ordine di 10^{20} hash al secondo e, considerando tutti i miner, il consumo energetico totale per l'elaborazioni è vicino a quello di un paese con oltre *45 milioni* di abitanti[Tse22] . Sebbene una percentuale significativa di tale energia provenga da fonti rinnovabili, si tratta comunque di una grossa spesa. [KÖ19]

Scalabilità limitata I blocchi possono contenere solo una ridotta quantità di transazioni ciascuno e il tempo medio di generazione di un singolo blocco è di circa *10 minuti*. Questo rappresenta un vincolo importante che impedisce a Bitcoin di competere o sostituire i tradizionali metodi di pagamento [Tse22].

Queste limitazioni, insite nel *Layer 1*, non rappresentano un fallimento del protocollo, ma una scelta progettuale per massimizzare la sicurezza. Tuttavia, esse rendono indispensabile lo spostamento della logica dei micropagamenti verso soluzioni di secondo livello, come il **Lightning Network**, che però sfrutta questo meccanismo a suo vantaggio. Il fatto che esso renda economicamente proibitivo riscrivere la storia della blockchain garantisce che gli stati dei canali pubblicati on-chain abbiano una finalità temporale affidabile, rendendo sicuri i meccanismo di timelock e di penalizzazione su cui si basa il protocollo.

3.4 Transaction malleability e introduzione di SegWit

La **transaction malleability** è una vulnerabilità strutturale introdotta nel protocollo Bitcoin dalle proprietà dell'algoritmo di firma ECDSA. Sebbene una firma garantisca che i fondi non possono essere spesi senza autorizzazione, essa non è univoca: è possibile alterare leggermente i dati della firma senza invalidarla [Ant17, LLW15].

Questo dettaglio crea problemi nelle operazioni in cui sono necessarie le multifirme, quindi molte delle procedure che fanno parte del protocollo del Lightning Network, permettendo ad una delle parti generare delle firme alternative, cambiando, ad esempio, il TXID.

Di fatto è già stato detto che, quando due parti, A e B, aprono un canale stanziano dei fondi con la Funding Transaction e da quel momento in poi per eseguire qualche operazione con tali fondi è necessaria la firma di entrambi. La parte A decide di spendere parte di quei soldi, quindi viene creata una transazione TX0, che necessita la firma di entrambi, che divide i fondi in due output, il primo sono i soldi spesi, e il secondo sono i soldi che rimangano comuni nel canale, per evitare che i soldi rimangono bloccati. Per evitare di bloccare i fondi per sempre viene firmata una transazione TX1 che divide nel modo corretto i fondi del secondo output di TX0, dopodiché verrà firmata TX0 [Ant17].

Viene utilizzata questa tecnica perché se firmassero solo TX0 e A sparisse, i fondi di B rimarrebbero bloccati per sempre, perché necessitano di A per essere mossi. Con TX1 che punta al TXID di TX0 B può comunque trasmettere la transazione e usare TX1 come chiave per riottenere la propria parte dei fondi [PD16].

Da qui nasce un problema fondamentale: nel protocollo originale il **TXID** è il risultato del doppio hash SHA-256 di tutti i campi della transazione, inclusi gli script di firma *scriptSig*. Poiché l'algoritmo ECDSA permette di generare firme valide ma bit-a-bit differenti, e dato che le funzioni di hash modificando un solo bit generano un output totalmente diverso, una delle parti potrebbe alterare la firma di TX0 dopo che TX1 è stata creata. Se una delle due parti dopo aver firmato la transazione di uscita TX1 decidesse di modificare la propria firma di TX0 e pubblicarla sulla blockchain, il TXID di quest'ultima cambierebbe. Di conseguenza, il riferimento contenuto in TX1 diventerebbe errato, puntando ad una transazione inesistente sulla blockchain. La controparte si troverebbe quindi con una transazione di rimborso non valida, perdendo la possibilità di recuperare i fondi stanziati nel canale [Ant17, DW15].

Già dal 2011 gli sviluppatori sapevano come risolvere questo problema, l'idea era quella di non includere gli script di firma dei delle due parti negli input utilizzati per generare il TXID di una transazione, mettendo le due firme dentro una nuove struttura chiamata *witness*, tale idea è chiamata **segregated witness**, o **segwit** [Ant17].

Modifiche come questa creano problemi di retrocompatibilità con i blocchi vecchi, e solo nel 2015 si è trovato un approccio che rendesse il segwit compatibile con

i vecchi blocchi. Per fare ciò bisogna che i nodi nuovi, che implementano la nuova modifica, non accettino nella blockchain blocchi che i nodi vecchi senza modifica non accetterebbero. Però i nuovi nodi possono non accettare nodi che i vecchi accetterebbero, avendo così la possibilità di applicare le nuove regole di consenso [Ant17].

Con la nuova struttura ogni transazione avrà due diversi ID: il TXID sarà il doppio hash SHA-256 dei dati, senza però i witness, mentre viene definito un nuovo ID, chiamato *WTXID*, che è il doppio hash SHA-256 di tutti i dati inclusi i witness [LLW15].

3.5 Bitcoin come livello base per il Lightning Network

Capitolo 4

Architettura e funzionamento dei canali di pagamento

I canali di pagamento permettono a due utenti di scambiarsi fondi più volte senza registrare ogni operazione sulla blockchain.

4.1 Apertura, gestione e chiusura di un canale

4.2 Aggiornamento dello stato e meccanismi di sicurezza

4.3 Gestione dei nodi offline e ruolo delle watchtower

Capitolo 5

Dal singolo canale di pagamento alla rete Lightning

Collegando più canali tra loro è possibile costruire una rete di pagamenti globale.

5.1 Pagamenti attraverso nodi intermedi

5.2 Routing dei pagamenti e diffusione delle informazioni

5.3 Aspetti di privacy nel Lightning Network

Capitolo 6

Limiti del Lightning Network e sviluppi futuri

Nonostante i vantaggi, il Lightning Network presenta ancora diversi limiti e criticità.

6.1 Topologia della rete e problemi di centralizzazione

6.2 Vulnerabilità e attacchi

6.3 Proposte di miglioramento e linee di ricerca

Conclusioni

In questa tesi è stato analizzato il Lightning Network come possibile soluzione al problema della scalabilità...

Bibliografia

[Als21] Eric Alston. Constitutions and blockchains: Competitive governance of fundamental rulesets. In James Caton, editor, *Handbook on Blockchain and Cryptocurrencies*. Routledge, 2021. Original work published 2019.

[Ant17] Andreas M Antonopoulos. *Mastering Bitcoin: Programming the Open Blockchain*. O'Reilly Media, 2nd edition, 2017.

[AOP22] Andreas M. Antonopoulos, Olaoluwa Osuntokun, and Rene Pickhardt. *Mastering the Lightning Network: A Second Layer Blockchain Protocol for Instant Bitcoin Payments*. O'Reilly Media, 1st edition, 2022. Fondamentale per l'architettura dei canali e degli HTLC.

[Bit25] Bitcoin Developers. Blockchain — bitcoin developer reference, 2025. Consultato il 23 ottobre 2025.

[But21] Vitalik Buterin. Why sharding is great: Demystifying the scalability trilemma. `https://vitalik.eth.limo/general/2021/04/07/sharding.html`, April 2021. Accesso: 30 Gennaio 2025.

[Cer10] Certicom Research. Sec 2: Recommended elliptic curve domain parameters. Technical report, Standards for Efficient Cryptography Group (SECG), 2010. Version 2.0.

[CGN14] Nicolas T. Courtois, Marek Grajek, and Rahul Naik. The unreasonable fundamental incertitudes behind bitcoin mining. *arXiv preprint arXiv:1310.7935*, 2014.

[Cha83] David Chaum. Blind signatures for untraceable payments. In *Advances in Cryptology*, Lecture Notes in Computer Science, pages 199–203. Springer, 1983.

[Con19] Marco Conoscenti. *Capabilities and Limitations of Payment Channel Networks for Blockchain Scalability*. PhD thesis, Politecnico di Torino, Torino, Italia, 2019. Tesi di dottorato, accesso aperto.

[DW15] Christian Decker and Roger Wattenhofer. A fast and scalable payment network with bitcoin duplex micropayments. In *International Conference on Networked Systems*, pages 3–12. IEEE, 2015. Analisi fondamentale sulla malleabilità e i canali di pagamento.

[FBL15] Mark Friedenbach, BtcDrak, and Eric Lombrozo. Bip 112: Checksequenceverify. `https://github.com/bitcoin/bips/blob/master/bip-0112.mediawiki`, 2015. Standard per timelock relativi.

[KÖ19] Sinan Küfeoğlu and Mahmut Özkuran. Energy consumption of bitcoin mining. 2019. Studio di riferimento per l'impatto ambientale e il dispendio elettrico del mining.

[Lig16] Lightning Network Developers. Bolt #1: Base protocol. `https://github.com/lightning/bolts/blob/master/01-protocol.md`, 2016. Specifica tecnica del protocollo Lightning Network, v1.0.

[LLW15] Eric Lombrozo, Johnson Lau, and Pieter Wuille. Bip 141: Segregated witness (consensus layer). `https://github.com/bitcoin/bips/blob/master/bip-0141.mediawiki`, 2015. Specifica tecnica fondamentale per la risoluzione della malleabilità.

[Mil17] Spencer Miller. Mmh32 from scratch: An introduction to hash functions. Technical report, University of Chicago, 2017. REU Program in Mathematics.

[Nak08] Satoshi Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2008. Whitepaper disponibile su bitcoin.org.

[NC17] Arvind Narayanan and Jeremy Clark. Bitcoin's academic pedigree. *Communications of the ACM*, 60(12):36–45, 2017.

[PD16] Joseph Poon and Thaddeus Dryja. The bitcoin lightning network: Scalable off-chain instant payments, January 2016. DRAFT Version 0.5.9.2.

[Pro20] Andrea Provino. Merkle tree e blockchain, 2020. Consultato il 23 ottobre 2025.

[Sta17] William Stallings. *Cryptography and Network Security: Principles and Practice*. Pearson, 7th edition, 2017.

[Tod14] Peter Todd. Bip 65: Op_checklocktimeverify. `https://github.com/bitcoin/bips/blob/master/bip-0065.mediawiki`, 2014. Standard per timelock assoluti.

[Tse22] David Tse. Lecture 15: From proof-of-work to proof-of-stake. EE 374: Blockchain Foundations, Stanford University, 2022. Analisi della transizione dai modelli PoW a protocolli di consenso energeticamente efficienti.