



La blockchain di Bitcoin e i wallet

Y2Y: BLOCKCHAIN E SMART CONTRACT

Organizzatori: Massimiliano Sala, Andrea Gangemi e Christopher Spennato

Speaker: Giorgio Rasetto

26 Ottobre e 2 Novembre, 2023

BitPolito

- Bitcoin
 - Struttura del blocco
 - Indirizzi
 - Nodi
 - Wallets
 - Le transazioni e la loro verifica
 - Mining e Proof of Work
 - Fork
 - Creazione di nuovi Bitcoin
 - Difetti
 - Lightning Network
- Approfondimenti
 - Crittografia di Bitcoin
 - Considerazioni sui wallet
 - Le transazioni P2PKH e P2SH
 - Segregated Witness e Taproot

Bitcoin

Bitcoin è una criptovaluta creata nel 2009 da un inventore anonimo, noto con lo pseudonimo Satoshi Nakamoto, che ha presentato l'idea alla fine del 2008 ([Bitcoin: A Peer-to-Peer Electronic Cash System](#)).

La *bitcoin* è una criptovaluta digitale, (pseudo)-anonima e distribuita, gestita da una rete *peer-to-peer* di utenti, che non ha bisogno di fare affidamento su autorità centrali. Il suo valore è determinato unicamente dall'offerta e dalla domanda.

Bitcoin registra tutte le transazioni tramite una blockchain pubblica, condivisa e gestita direttamente dai nodi di una rete *peer-to-peer*.

Un blocco è composto da due parti: l'intestazione (*header*) e il corpo (*body*). Le transazioni sono incluse nel corpo del blocco, mentre l'intestazione contiene diversi campi per gestire il blocco stesso.

Struttura della blockchain

In una blockchain, **ogni blocco include l'hash del blocco precedente**, creando così il collegamento tra due blocchi. L'iterazione di questa procedura forma la catena e garantisce l'integrità del blocco precedente, fino al blocco genesi, che è il primo blocco della blockchain.

A volte, due blocchi possono essere prodotti contemporaneamente, causando una **fork** (o biforcazione) nella blockchain.

Ciascuna blockchain ha un algoritmo specifico per gestire le fork. I blocchi che non vengono selezionati per essere inclusi nella catena vengono chiamati *stale blocks* (o blocchi orfani). Questi blocchi sono invalidi, e quindi anche le transazioni al loro interno sono invalide.

Struttura del blocco

Field	Description	Size
Magic no	value always 0xD9B4BEF9	4 bytes
Blocksize	number of bytes following up to end of block	4 bytes
Blockheader	consists of 6 items	80 bytes
Transaction counter	positive integer <code>VI = VarInt</code>	1 - 9 bytes
transactions	the (non empty) list of transactions	<Transaction counter>-many transactions

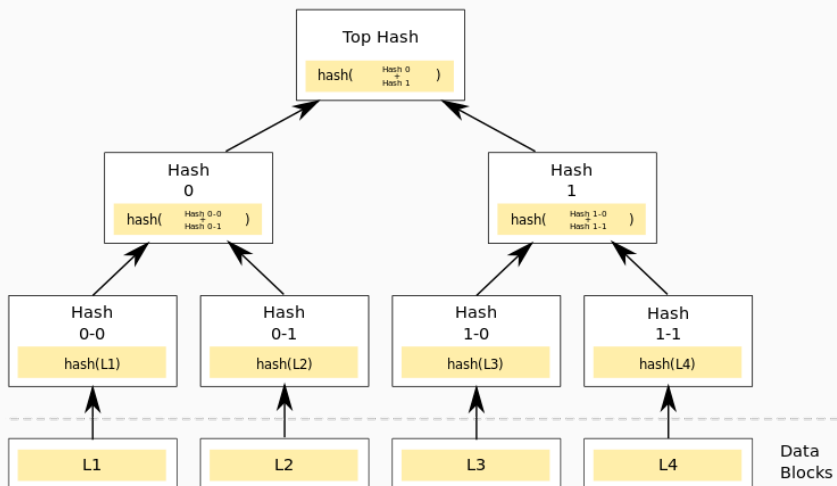
Il numero esadecimale D9B4BEF9 identifica il blocco come parte della *timechain* (o *blockchain*) di Bitcoin.

Intestazione del blocco

Field	Purpose	Size (Bytes)
Version	Block version number	4
hashPrevBlock	256-bit hash of the previous block header	32
hashMerkleRoot	256-bit hash based on all of the transactions in the block	32
Time	Current block timestamp as seconds since 1970-01-01T00:00 UTC	4
Bits	Current target in compact format	4
Nonce	32-bit number (starts at 0)	4

- **hashMerkleRoot** è l'hash di tutti gli hash di tutte le transazioni nel blocco.
- **Nonce** è un valore che viene inserito casualmente fino a quando il *digest* (calcolato eseguendo SHA-256 due volte di fila) del blocco è inferiore al *target* (campo **bits** nella figura sopra). Solo in questo caso il blocco viene inserito nella blockchain. Questo lavoro è svolto dai **miners**.

Merkle trees I



L'hash in cima all'albero è comunemente chiamato **Merkle root**.

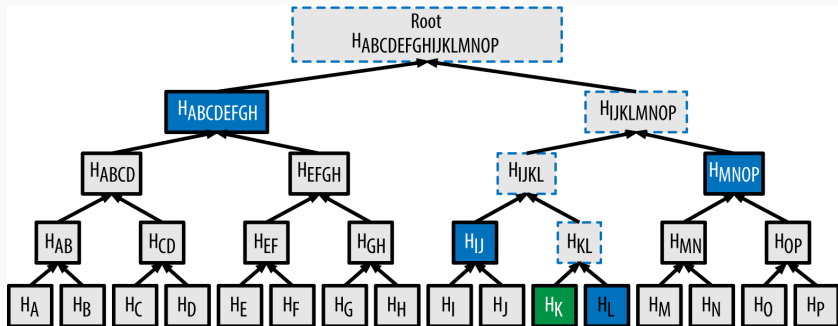
Merkle trees II

I *Merkle trees* vengono utilizzati per verificare rapidamente se una transazione è presente in un certo blocco, utilizzando una quantità limitata di informazioni.

Per costruzione, tutto ciò che serve è la transazione “vicina” a quella da verificare, insieme agli hash sul percorso dalla transazione da verificare fino alla radice dell’albero.

Tale percorso ha una lunghezza proporzionale al logaritmo del numero di transazioni nel blocco e, pertanto, l’algoritmo di verifica utilizzando un *Merkle tree* è estremamente più efficiente rispetto a una semplice verifica sequenziale su tutte le transazioni nel blocco.

Merkle trees III



Come esempio, per dimostrare la presenza della transazione H_K nel blocco, possiamo semplicemente utilizzare la transazione H_L e gli hash evidenziati, quindi verificare che possiamo calcolare nuovamente la corretta *Merkle root* del blocco.

Osserviamo che nella struttura del blocco non c'è l'hash del blocco precedente ma solo l'**hash dell'header del blocco precedente**.

Gli hash dei blocchi sono calcolati dai miners, e devono essere inferiori al *target* affinché il blocco venga inserito, ma non vengono effettivamente registrati nella blockchain.

Questo non crea un problema di sicurezza, perché grazie all'effetto valanga delle funzioni hash, se ci fosse una modifica anche di un solo bit in una transazione del blocco, allora la *Merkle root* cambierebbe. La radice è memorizzata nell'*header*, e di conseguenza cambierebbe l'hash dell'*header*.

C'è anche un altro modo per identificare i blocchi, in base alla loro **altezza**.

L'altezza del blocco indica la sua distanza dal blocco genesi, che per definizione ha altezza zero. Il primo blocco estratto dopo il blocco genesi ha altezza 1, il successivo ha altezza 2 e così via.

Attualmente (27 settembre 2023 09:30 a.m.), l'ultimo blocco estratto nella blockchain è il blocco con altezza 809542.

I blocchi di Bitcoin hanno una dimensione massima di 1MB e ogni blocco contiene in media tra 2000 e 3000 transazioni. Al momento, la rete Bitcoin gestisce in media circa 4 transazioni al secondo.

Solo a titolo di confronto, Visa ha una velocità massima di circa 24.000 transazioni al secondo, e PayPal quasi 200.

La blockchain di Bitcoin è pubblica, ma non così comoda da visualizzare. Ci sono molti siti dedicati a rendere più semplice l'esplorazione della blockchain. Ogni *explorer* ha le sue funzioni e peculiarità.

Ecco una lista dei più utilizzati:

- <https://blockstream.info>;
- <https://mempool.space/it/>;
- <https://www.blockchain.com/explorer>;
- <https://blockchair.com/bitcoin>;
- <https://live.blockcypher.com>.

Bitcoin non ha il concetto di saldo associato a ciascun utente.

La blockchain di Bitcoin è una lunga lista di transazioni, dove le transazioni relative a un singolo utente non possono essere facilmente riconosciute e raggruppate insieme.

Vediamo ora in generale cosa accade quando si effettua una transazione su Bitcoin. Ad esempio, supponiamo che Alice abbia ricevuto 1 bitcoin (BTC) in una transazione precedente e voglia trasferirlo a Bob.

Transazioni Bitcoin: una panoramica

La procedura (semplificata) è la seguente:

- Alice informa Bob che gli vuole trasferire 1 BTC.
- Bob crea una coppia di chiavi privata/pubblica utilizzando i crittosistemi basati su curve ellittiche, e calcola l'indirizzo Bitcoin Add_B associato alla chiave pubblica.
- Bob invia ad Alice il nuovo indirizzo Bitcoin Add_B .
- Alice scrive la transazione impostando l'indirizzo di output della transazione Add_B . Gli input della transazione invece appartengono ad Alice e contengono almeno 1 BTC.
- Alice invia a Bob la transazione firmata digitalmente.
- Bob verifica la correttezza della firma digitale ed è quindi certo che Alice abbia accettato di pagare 1 BTC.

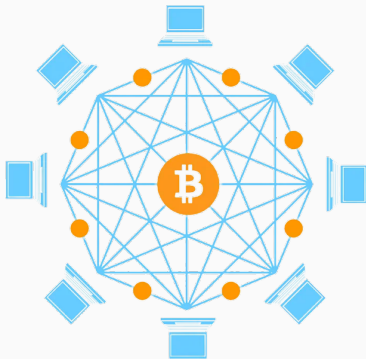
Non si utilizza direttamente la chiave pubblica come l'indirizzo a cui si trasferiscono i bitcoin, ma si usa il cosiddetto **indirizzo Bitcoin**, che è più breve e viene derivato dalla chiave pubblica.

Semplificando, la chiave pubblica viene hashata due volte, prima usando SHA256, poi usando RIPEMD160. Successivamente, viene calcolato un **checksum** che è concatenato al *digest* precedente.

Ci sono alcuni vantaggi nell'utilizzare l'indirizzo Bitcoin invece della chiave pubblica direttamente:

- Avere un controllo interno (il checksum) rende molto difficile inviare bitcoin a indirizzi errati.
- Avere una codifica evita errori di scrittura classici (ad esempio, confondere la lettera O con il numero 0).
- Generalmente, solo gli indirizzi Bitcoin sono nella blockchain, non le chiavi pubbliche che rimangono sostanzialmente segrete fino a quando i bitcoin non vengono spesi di nuovo (cioè, quando si firma digitalmente).
- Grazie alle funzioni di hash utilizzate nel protocollo, non è possibile risalire dall'indirizzo Bitcoin alla chiave pubblica.

Full nodes e light nodes



I nodi della rete Bitcoin possono essere suddivisi in:

- **Full nodes.**
- **Light nodes.**
- **Miners.**

Attivare un Bitcoin *full node* significa unirsi alla comunità Bitcoin a tutti gli effetti. Ogni full node memorizza l'intera blockchain (cioè tutte le transazioni di tutti gli utenti dall'inizio di Bitcoin) e gestisce tutti gli aspetti del protocollo Bitcoin, compresa la verifica di tutte le transazioni inviate alla rete. Per attivare un full node sono necessari:

- Desktop o laptop con una versione recente di Windows, Mac OS X o Linux.
- 8 GB+ di memoria (RAM).
- 500+ GB di spazio su disco libero.
- Connessione Internet ad alta velocità.

Light nodes

Un *light node* non memorizza l'intera blockchain, ma solo gli *header* di tutti i blocchi, risparmiando oltre il 99,99% di spazio, e richiede una potenza computazionale minima.

Nonostante queste limitazioni, può comunque svolgere un ruolo nella rete Bitcoin. Può **inviare transazioni** alla rete e può **verificarne la correttezza formale**. Tuttavia, non può verificare se il destinatario di un pagamento ha effettivamente ricevuto i bitcoin. Questo richiede l'intera blockchain.

Un light node può comunque verificare autonomamente e rapidamente una transazione affidandosi a un full node, utilizzando il meccanismo di **verifica del pagamento semplice** (*Simple Payment Verification*, o SPV).

Verifica del pagamento semplice (SPV) I

Questo meccanismo permette di verificare, senza trasferimenti significativi di dati, che una transazione sia stata inclusa in un blocco (e quindi accettata dalla rete), richiedendo a un full node tutti gli hash intermedi del *Merkle tree* e la *Merkle root*.

Utilizzando la transazione e gli hash intermedi, il light node può ricreare tutti i calcoli necessari e verificare se ottiene come hash finale la *Merkle root* del blocco (che è presente nell'*header* del blocco, e il light node possiede gli *header* di tutti i blocchi).

Verifica del pagamento semplice (SPV) II

Il full node non può ingannare il light node, poiché è (quasi) impossibile creare hash intermedi falsi che conducano dalla transazione alla *Merkle root*. Pertanto, il light node è certo che la transazione si trova in quel blocco della blockchain.

Il light node non può verificare autonomamente la correttezza di una transazione, ma **può verificare la sua presenza in un blocco**, assumendo che se è in un blocco, è stata accettata dalla rete ed è quindi valida.

Tutto ciò può essere realizzato trasferendo pochissimi byte tra il full node e il light node, grazie alla struttura del *Merkle tree*, dove gli hash intermedi sono dell'ordine del logaritmo del numero di transazioni in un blocco.

Il terzo protagonista della rete Bitcoin è il **miner**. I miners sono coloro che utilizzano la propria potenza di calcolo per garantire l'integrità della rete, e allo stesso tempo inserire un nuovo blocco alla catena.

I miners, per poter effettuare il loro lavoro, hanno bisogno di conoscere la storia di tutta la blockchain.

Tuttavia, essi non hanno necessariamente bisogno di scaricarsi un full node: **possono unirsi a una mining pool, ovvero un'azienda che raduna molti miners, e farsi fornire da loro il blocco che devono tentare di validare.**

Come abbiamo appena osservato, un miner non è necessariamente un full node.

Neanche il viceversa è vero: un utente potrebbe avere scaricato un full node, e quindi l'intera blockchain, sul proprio computer, ma non essere comunque interessato all'attività di mining.

Wallet

Per essere in grado di acquistare e rivendere Bitcoin (nonché altre criptovalute), è necessario disporre di un software (o hardware) dedicato, chiamato **wallet** (portafoglio), che conserva tutti i dati relativi alle transazioni (indirizzi Bitcoin, chiavi pubbliche, ecc.).

Ci sono molti tipi di wallet e possono essere divisi in varie categorie, a seconda della piattaforma utilizzata o del grado di autonomia nell'interazione con la rete Bitcoin.

Un aspetto importante è che un wallet **non contiene** i bitcoin (o qualunque altra criptovaluta): essi sono registrati sulla blockchain, sotto forma di output di transazione.

Le cinque categorie sono le seguenti:

- Desktop wallet.
- Mobile wallet.
- Web wallet.
- Hardware wallet.
- Paper wallet.

I primi tre sono anche chiamati **hot wallets**, mentre gli ultime due sono detti **cold wallets**.

Desktop wallet

I desktop wallets sono software che si installano sul computer. Esistono versioni per Windows e Mac OS con molte funzionalità e controllo. Lo svantaggio è la sicurezza, che dipende anche dal computer stesso.

Mobile wallet

Questo è il tipo di wallet più comune e diffuso. Si installano su smartphone (Android o iOS), e sono molto leggeri e facili da usare. Come nel caso precedente, la sicurezza dipende dal dispositivo su cui sono installati.

Entrambi i tipi di wallet precedenti possono prevedere la memorizzazione delle chiavi di transazione localmente (su un PC o smartphone) o sul server del fornitore di servizi.

Web wallet

I web wallet sono accessibili tramite un browser web e conservano le informazioni chiave delle transazioni su un server di proprietà di un terzo. Pertanto, non è consigliabile utilizzare questo tipo di wallets per conservare grandi quantità di criptovaluta.

Hardware wallet

Si tratta di dispositivi mobili che si collegano al PC tramite la porta USB e sono appositamente progettati per gestire un wallet. Sono considerati molto sicuri e adatti anche per l'archiviazione di grandi quantità di criptovaluta.

Paper wallet

Le chiavi segrete e pubbliche e gli indirizzi Bitcoin possono anche essere stampati per l'archiviazione a lungo termine: tale sistema viene di solito chiamato paper wallet. Si tratta di un metodo non tecnologico, ma molto sicuro.



Figure 1: Trezor hardware wallet

C'è un'altra importante distinzione che dovrebbe essere fatta tra i wallet, tra servizi che ci consentono di detenere le chiavi e servizi che non lo fanno.

- **Custodial wallet:** sono quei wallets in cui la custodia delle chiavi è responsabilità del fornitore del servizio.
- **Non Custodial wallet:** sono quei wallets in cui la custodia delle chiavi è responsabilità dell'utente.

Nel primo caso, l'utente non ha realmente il possesso delle proprie criptovalute, poiché non possiede le chiavi!

Una terza distinzione tra i wallet riguarda come scelgono di generare le chiavi. Possiamo distinguere tra **wallet deterministici e non deterministici**. Cominciamo descrivendo un wallet non deterministico.

- In un wallet non deterministico, ogni chiave viene generata in modo indipendente, quindi non c'è collegamento tra una chiave e un'altra all'interno del wallet.
- Sono necessari backup continui di tutte le chiavi, altrimenti, se per qualche motivo il wallet diventa inaccessibile, tutti i bitcoin collegati a queste chiavi vengono persi.
- Questa idea di un wallet entra quindi in conflitto con il principio di utilizzare un nuovo indirizzo ogni volta che si riceve un bitcoin da una transazione (vedremo meglio questo punto in seguito), poiché generare più indirizzi significa conservare più chiavi private che richiedono backup continui.

In un wallet deterministico, ogni chiave è derivata da una **master key**, che a sua volta è derivata da un **seed** (seme). Tutte le chiavi possono essere generate nuovamente semplicemente conoscendo il *seed*.

- Il metodo più utilizzato per derivare le chiavi utilizza una struttura ad albero ed è chiamato **Hierarchical Deterministic** (HD) wallet.
- Per rendere il *seed* facile da memorizzare, all'utente viene chiesto di ricordare un elenco di parole in lingua inglese da cui è possibile recuperarlo facilmente.
- Per esportare le chiavi da un wallet a un altro, il *seed* è quindi sufficiente.

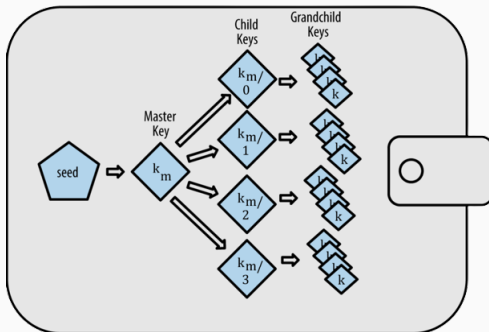


Figure 2: HD wallet

Transazioni

Transazioni multi-input e multi-output

In generale, una transazione Bitcoin può avere più input e più output.

Se Alice deve pagare 1 BTC a Bob, non è necessario che abbia ricevuto in passato una transazione esattamente di 1 BTC. Può specificare nella transazione i propri indirizzi di input in modo che, tutti insieme, abbiano ricevuto più di 1 BTC.

Se la somma di tutti questi input supera 1 BTC, può gestire facilmente la transazione specificando due indirizzi di output: uno di Bob per accreditare 1 BTC e uno suo per accreditare la differenza, il cosiddetto **indirizzo di resto**.

Transazione Coinbase

La prima transazione di ogni blocco è leggermente diversa da tutte le altre. Non ha alcun input, e ha come output l'indirizzo del miner che ha inserito il blocco in cui è contenuta la transazione coinbase.

Questa è l'unica modalità per generare bitcoin. I nuovi bitcoin vengono quindi creati per remunerare i miners per il loro lavoro per la comunità.

E' possibile effettuare transazioni anche per una frazione di bitcoin, fino all'ottava cifra decimale.

In effetti, il **satoshi** è stato definito come centomilionesima parte di un bitcoin, cioè ci vogliono cento milioni di satoshi per ottenere un bitcoin (BTC).

I mattoni fondamentali di una transazione Bitcoin sono i cosiddetti **transaction output**. I full nodes (ne parleremo più avanti) tengono traccia di tutti i *transaction output* non spesi, spesso abbreviati come UTXO. Chiamiamo **UTXO set** l'insieme di tutti gli UTXO.

- Ogni transazione rappresenta una transizione di stato nell'UTXO set: gli UTXO utilizzati come input vengono eliminati, mentre gli output diventano nuovi UTXO, che vengono aggiunti all'UTXO set.
- Quando un utente riceve bitcoin, significa che il suo wallet ha individuato un UTXO che può essere speso con una delle sue chiavi private.
- Il saldo di un wallet è la somma di tutti gli UTXO che possono essere spesi da quel wallet.

Commissioni I

La **fee** (commissione) è l'importo che il mittente della transazione è disposto a pagare per far elaborare la sua operazione.

Le commissioni sono liberamente determinate e pagate dal mittente della transazione, e maggiore è la commissione, più è probabile che la transazione venga inclusa rapidamente nella blockchain.

Le commissioni non sono esplicitamente indicate. La commissione è semplicemente la differenza tra la somma di tutti gli input e la somma di tutti gli output, e può persino essere zero.

Molti wallet Bitcoin suggeriscono automaticamente le commissioni all'utente considerando le commissioni pagate di recente dalle transazioni incluse nei blocchi più recenti e la lunghezza della transazione.

Le commissioni servono a remunerare ulteriormente i miners per il loro lavoro, ma anche a rendere la rete resistente a un attacco di Denial of Service.

Senza commissioni, chiunque potrebbe costruire numerose transazioni ordinarie e valide dal proprio indirizzo a un altro dei propri indirizzi, intasando la rete e impedendo agli altri di effettuare transazioni.

Con un costo di pochi euro in commissioni per ogni transazione, riempire un blocco con trasferimenti validi ma semplici da un indirizzo a un altro costerebbe all'attaccante decine di migliaia di euro.

Esempio di transazione

Transaction View information about a bitcoin transaction

0627052b6f28912f2703066a912ea577f2ce4da4caa5a5fbd8a57286c345c2f2

1Cdid9KFAaatwczBwBttQcwXYCpvK8h7FK (0.1 BTC - Output)



1GdK9UzpHBzqzX2A9JFP3Di4weBwqgmoQA
- (Unspent) 0.015 BTC
1Cdid9KFAaatwczBwBttQcwXYCpvK8h7FK -
(Unspent) 0.0845 BTC

97 Confirmations

0.0995 BTC

Summary

Size 258 (bytes)

Received Time 2013-12-27 23:03:05

Included In [277316](#) (2013-12-27 23:11:54 +9
Blocks minutes)

Inputs and Outputs

Total Input 0.1 BTC

Total Output 0.0995 BTC

Fees 0.0005 BTC

Estimated BTC Transacted 0.015 BTC

Output di una transazione 1

Uscite 1

Indice	0	Dettagli	non spesi
Indirizzo	1GdK9UzpH8zqzX2A9JFP3Di4weBwqgmoQA 	Valore	0.01500000 BTC
Pkscript	OP_DUP OP_HASH160 ab68025513c3dbd2f7b92a94e0581f5d50f654e7 OP_EQUALVERIFY OP_CHECKSIG		
<hr/>			
Indice	1	Dettagli	non spesi
Indirizzo	1Cdld9KFAaatwcz8wBttQcwXYCpvK8h7FK 	Valore	0.08450000 BTC
Pkscript	OP_DUP OP_HASH160 7f9b1a7fb68d60c536c2fd8aeea53a8f3cc025a8 OP_EQUALVERIFY OP_CHECKSIG		

Figure 3: Output di una transazione

Output di una transazione II

L'output di una transazione contiene vari campi: tuttavia, alcuni di questi sono creati dal client che interagisce con la blockchain.

- Le informazioni memorizzate sulla blockchain sono l'importo di bitcoin, l'indirizzo e lo **script di blocco** (*locking script*), una sorta di enigma crittografico (chiamato *Pkscript* nella Figura). Lo script di blocco contiene le condizioni necessarie per spendere l'output.
- Le rimanenti informazioni nella Figura sono *Indice*, che tiene traccia del numero totale di output, e *Dettagli*, che ci dice se un certo output è già stato speso o meno. Se non è stato ancora speso, fa parte dell'UTXO set.

Ogni UTXO è associato a uno script di blocco.

Input di una transazione 1

ingressi ⓘ


		Dettagli
Indirizzo	1Cdd9KFAaatwczBw8ttGcwXYCpvK8h7FK 	Valore 0.10000000 BTC
PKscript	OP_DUP OP_HASH160 7f9b1a7fb68d60c536c2fd8aaaa53a8f3cc025a8 OP_EQUALVERIFY OP_CHECKSIG	
Sigscript	3045022100884d142d86652a3f47ba4746ec719bbfd040a570b1deccbb6498c75c4ae24cb02204b9f039ff08df09cbe9f6addac960298cad530a863ea8f53982c09db8f6e381301 0484ecc0d46f1918b30928fa0e4ed99f16a0fb4fde0735e7ade8416ab9fa423cc5412336376789d172787ec3457eee41c04f4938de5cc17b4a10fa336a8d752adf	
Witness	N/A	

Figure 4: Input di una transazione

Input di una transazione II

- Un transaction input identifica un UTXO specifico, che viene speso tramite una prova di proprietà, chiamata **script di sblocco** (o *unlocking script*, *Sigscript* nella figura).
- Per ciascun UTXO utilizzato, il wallet crea un input sbloccabile tramite lo script di sblocco corrispondente.
- Le informazioni memorizzate sono simili a quelle fornite per l'output. In particolare, *pkscript* si riferisce allo script di blocco dell'UTXO a cui fa riferimento l'input in questione, *sigscript* rappresenta lo script di sblocco, mentre il campo *witness* verrà discusso in seguito.

Ogni input contiene uno script di sblocco, necessario per sbloccare il contenuto dell'UTXO a cui fa riferimento.

La maggior parte delle transazioni consuma output bloccati tramite uno script **Pay-to-Public-Key-Hash** (P2PKH). In questo caso, i bitcoin vengono inviati, come suggerisce il nome, all'hash di una chiave pubblica, cioè ad un address Bitcoin.

Bitcoin permette anche di scrivere semplici scripts, quali ad esempio transazioni multifirma, grazie a speciali transazioni chiamate **Pay-to-Script-Hash** (P2SH). In questo caso, i bitcoin sono bloccati sempre su un address, che però rappresenta l'hash di uno *script*.

- Gli script contenuti negli input e negli output di una transazione Bitcoin sono scritti in un linguaggio di *scripting* chiamato **Script**. È simile ai linguaggi di programmazione degli anni '60.
- Il linguaggio scelto è molto semplice e ha capacità limitate, ma offre comunque la possibilità di scrivere condizioni complesse. In particolare, Script non è *Turing completo*, il che significa che non è possibile creare cicli ma solo flussi condizionali (if). Ciò rende prevedibile il tempo di esecuzione di uno *script*.
- Quando una transazione viene convalidata, gli *script* vengono eseguiti in sequenza (iniziando sempre con lo *script* di sblocco) e viene convalidata se le condizioni dello *script* di blocco vengono soddisfatte. Ciascun input viene convalidato in modo indipendente.

Validazione e consenso

Prima che una transazione diventi effettiva, deve superare due fasi di verifica:

- **Verifica della transazione.**
- **Validazione del blocco contenente la transazione.**

Poiché tutti gli utenti sulla rete hanno la stessa copia della blockchain e lo stesso software per la verifica automatica delle transazioni, se tutto è in ordine, l'intera rete accetta la transazione come corretta e la transazione viene considerata **verificata**.

Il problema del double-spending

Questa verifica non è sufficiente. Infatti, se tutte le transazioni verificate dalla rete venissero automaticamente inserite in un blocco della blockchain, ci sarebbe un problema significativo dovuto alla natura asincrona della rete: **Il problema del double-spending**.

Supponiamo che Alice invii alcuni BTC a Bob. Se Alice dovesse pagare un altro utente, Carl, approssimativamente nello stesso momento in cui ha pagato Bob, la verifica della rete potrebbe avere successo per entrambe le transazioni. Alice finirebbe per pagare due persone con gli stessi bitcoin!

Questo è il motivo per cui, dopo la fase di verifica della transazione, c'è ancora la **fase di validazione del blocco**.

La validazione del blocco è il punto più critico di tutto il protocollo, poiché è lì che viene decisa la composizione del nuovo blocco da inserire nella blockchain e quindi quali transazioni diventano finalizzate.

Entrano in gioco i **miners**. Ciascun miner prende un certo numero di transazioni dal pool di transazioni verificate dalla rete (in base alle commissioni di transazione), e crea un nuovo potenziale blocco.

Ricordiamo che le commissioni sono determinate da coloro che effettuano le transazioni, quindi coloro che desiderano che le loro transazioni vengano incluse nella blockchain il prima possibile saranno disposti a pagare commissioni più elevate, rendendo le loro transazioni più attraenti per i miners.

Proof of Work I

Ogni miner, dopo aver selezionato le transazioni da includere nel corpo del blocco, calcola l'hash (applicando SHA-256 due volte di seguito) dell'intero blocco. Ognuno di loro varia il nonce finché l'output dell'hash è inferiore al *target*.

Il primo nodo che "risolve" il blocco, cioè ottiene un hash inferiore al *target*, lo trasmette alla rete, dove viene accettato come il prossimo blocco nella catena.

L'hash del suo *header* rappresenta in modo univoco il blocco e verrà utilizzato come riferimento dal blocco successivo.

La rete aggiorna periodicamente e automaticamente il valore del *target* per controllare il ritmo di inserimento dei nuovi blocchi.

Più precisamente, la rete aggiorna il *target* ogni **2.016 blocchi** in modo che, **in media, ogni blocco venga inserito ogni 10 minuti**.

Questo processo di validazione del blocco è chiamato **Proof of Work** (PoW) ed è il **protocollo di consenso** per l'aggiunta di nuovi blocchi alla blockchain di Bitcoin.

Il motivo di questa costruzione (che è altamente energivora e richiede computer potenti) è che se la decisione su quale blocco accettare fosse basata solo sulla maggioranza degli utenti, un attaccante potrebbe semplicemente registrarsi migliaia di volte, creando identità fittizie per controllare la rete (il cosiddetto **attacco Sybil**).

È possibile che due miners ottengano due blocchi validi diversi quasi contemporaneamente. Non possono essere aggiunti entrambi i blocchi, in quanto una volta che uno viene aggiunto, l'altro non è più considerato un blocco valido in quanto manca dell'hash dell'*header* dell'ultimo blocco aggiunto.

Non è operativamente fattibile accettare semplicemente chi arriva una frazione di secondo prima a causa dell'asincronia della rete. In questo caso si verifica una **fork** (biforcazione) della catena.

Quando si verifica una biforcazione, entrambi i blocchi sono considerati potenziali nuovi blocchi, e **i miners sono incoraggiati a continuare a minare su uno qualsiasi dei due rami.**

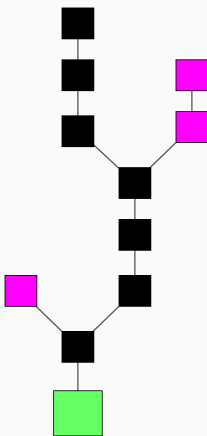


Figure 5: Blocchi della catena principale (nero), con il blocco genesis (verde), e blocchi abbandonati/inutilizzati (viola).

Non appena un nuovo blocco viene validato e aggiunto a uno dei due rami, i miners tendono a spostarsi sul ramo più lungo, che ha una probabilità maggiore di diventare quello definitivo. L'altro blocco (di solito chiamato blocco orfano) viene scartato.

Per questo motivo, i miners tendono fortemente a spostarsi sul ramo più lungo e **le biforcazioni tendono a risolversi rapidamente**.

Oltre alle solite fork causate (involontariamente) dalla convalida simultanea di più blocchi da parte di diversi miners, ci sono altre ragioni che possono portare a una biforcazione della blockchain.

Ad esempio, una parte degli utenti della blockchain potrebbe **proporre modifiche al protocollo**. Se la proposta viene accettata da tutta la rete, non si crea alcuna biforcazione.

Tuttavia, dato che la rete Bitcoin è vasta, raggiungere un accordo totale su un cambiamento, soprattutto uno significativo, è difficile. Quindi, potrebbero emergere gruppi contrapposti: quelli a favore del cambiamento e quelli che desiderano mantenere le vecchie regole.

Se gli innovatori non sono disposti a compromessi, hanno due approcci:

- **Soft Fork:** Adottare un protocollo che consenta a chi utilizza il vecchio software di continuare a funzionare normalmente (quindi le transazioni eseguite utilizzando il nuovo protocollo devono essere riconosciute come valide anche da chi utilizza il vecchio software).
- **Hard Fork:** Adottare un protocollo che non sia compatibile con quello vecchio.

Soft Fork e Hard Fork II

In sostanza, un soft fork non è una vera biforcazione nel senso che non causa una ramificazione della blockchain; il protocollo rimane unificato.

Dopo un hard fork, tuttavia, emergono due blockchain distinte, ognuna sviluppandosi indipendentemente e condividendo tutti i blocchi fino al momento della biforcazione.

Durante la sua storia, Bitcoin ha subito vari hard fork, ma quasi tutti hanno dato origine a una nuova blockchain e a una nuova comunità che è stata vitale solo per un breve periodo e poi è stata abbandonata o seguita molto poco.

Gli ultimi due aggiornamenti di Bitcoin (SegWit e Taproot) sono stati applicati tramite una soft fork.

Creazione di Bitcoin I

Come abbiamo menzionato, i bitcoin vengono creati quando ogni blocco viene aggiunto alla blockchain.

All'interno di ciascun blocco, ci sono numerose transazioni che trasferiscono la proprietà dei BTC da un utente a un altro. Tuttavia, c'è una transazione speciale, chiamata **coinbase**, che è diversa da tutte le altre.

L'output della transazione coinbase è l'indirizzo del miner. Il suo input è un indirizzo speciale chiamato **coinbase**, dove non sono stati effettuati depositi.

Inizialmente, la ricompensa per il miner era di **50 bitcoin**. Il protocollo Bitcoin stabilisce che questa ricompensa viene **dimezzata** ogni **210.000 blocchi** (dato che un blocco viene estratto in media ogni 10 minuti, ciò avviene circa ogni **4 anni**).

Il sistema è quindi progettato per ridurre geometricamente la disponibilità di nuove monete. Questa riduzione della ricompensa è chiamata **Halving**.

$$\frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \dots = \sum_{k=1}^{\infty} \left(\frac{1}{2}\right)^k = 1.$$

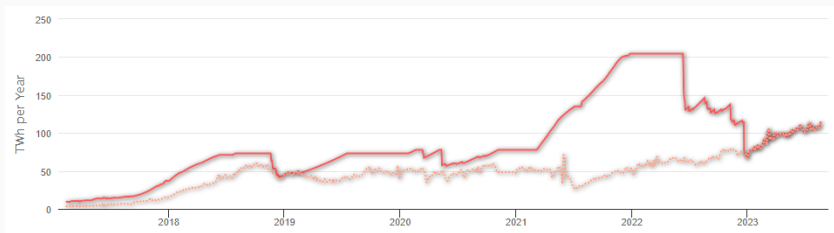
Il numero di bitcoin si avvicinerà asintoticamente a un valore totale di 21 milioni. È facile verificare che ciò accadrà nel gennaio 2140, oltre il quale **i miners non riceveranno più alcuna ricompensa per i blocchi estratti e quindi non saranno più creati ulteriori bitcoin.**

Limiti di Bitcoin

Le caratteristiche più contestate di Bitcoin sono:

- Il costo economico e ambientale del suo meccanismo di Proof of Work (PoW).
- La lentezza nella elaborazione delle transazioni rispetto ai sistemi più tradizionali.
- La tendenza all'emergere di pool di mining che rischiano di centralizzare la validazione dei blocchi.
- La visibilità di tutte le transazioni e il conseguente rischio per la privacy.

Consumo di Energia di Bitcoin



Nell'ultimo anno, i miners di Bitcoin hanno consumato circa 100 TWh.

Lentezza e Scarsa Scalabilità

Attualmente, tutti i blocchi di Bitcoin sono vicini alla dimensione massima di 1MB. Ricordiamo che, in media, un blocco viene aggiunto ogni 10 minuti secondo il protocollo.

Si potrebbe **ridurre l'intervallo di 10 minuti tra i blocchi**, il che comporta il rischio di creare biforcazioni più lunghe e potenziali rischi per la sicurezza, o **aumentare la dimensione del blocco**.

Aumentare la dimensione del blocco potrebbe migliorare la velocità, ma paradossalmente peggiorerebbe un altro problema di scalabilità: **la dimensione dell'intera blockchain**.

Se la blockchain cresce più velocemente, la sua dimensione potrebbe diventare così grande che non tutti possono permettersi di conservare l'intera blockchain, portando a una decentralizzazione ridotta.

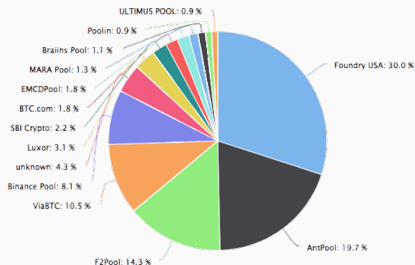
Due proposte per risolvere il problema della scalabilità sono *Bitcoin Cash* e *Bitcoin SV*, due hardfork dell'originale blockchain di Bitcoin.

Bitcoin Cash nacque dopo l'aggiornamento SegWit del protocollo Bitcoin. L'idea era quella di aumentare la dimensione di ogni blocco da 1MB a 4MB, in modo tale da migliorare la scalabilità della rete. Il difetto è che blocchi più grandi si propagano più lentamente.

Bitcoin SV (*Satoshi Vision*) è a sua volta una hard fork di Bitcoin Cash. L'idea è la stessa, ovvero aumentare ancora di più la dimensione di un blocco per aumentare di conseguenza il numero di transazioni processate.

Mining Pools

I *mining pools* sono gruppi di miners cooperanti che accettano di condividere i premi ottenuti in proporzione al loro potere di hash che hanno messo a disposizione. Far parte di un *mining pool* è un aspetto positivo per un singolo miner, in quanto otterrà remunerazioni più uniformi e prevedibili, ma l'aumento dei *mining pool* sta creando un livello di aggregazione che va contro la filosofia della blockchain.



Distribuzione hash rate - settembre 2023

Nella blockchain di Bitcoin sono registrate tutte le transazioni in chiaro. La privacy è assicurata dal fatto che nella blockchain sono indicati gli indirizzi bitcoin di input e di output di tutte le transazioni, ma non è noto a quali persone fisiche appartengono tali indirizzi.

Questo però non dà una sicurezza assoluta e ci sono studi sul livello di privacy di Bitcoin. Ci sono tre tipi di rischi per la privacy:

- **Locali:** rischi dovuti all'uso del PC/Smartphone.
- **Network:** rischi legati alla rete.
- **Blockchain:** rischi legati alla struttura della blockchain. In particolare:
 - alcuni utenti riusano gli stessi indirizzi più volte;
 - in genere, gli input di una transazione sono della stessa persona e l'output più piccolo anche (perchè rappresenta il *change adress*).

Ognuno di questi difetti è stato affrontato all'interno di Bitcoin, con studi e discussioni della community su possibili modifiche del protocollo mirate alla loro risoluzione o minimizzazione.

In molti casi questi difetti hanno portato anche alla **creazione di progetti di nuove criptovalute** con una natura a volte anche decisamente diversa, che nascono proprio con l'**obiettivo primario di eliminare o ridurre uno o più problemi evidenziati da Bitcoin**.

Ci sono poi anche dei progetti esterni che però non sono alternativi a Bitcoin, ma nascono a suo supporto. Uno dei più interessanti, nato con l'obiettivo di aumentare il numero di operazioni gestibili e la riduzione delle commissioni, è il **Lightning Network**.

Lightning Network

Il [Lightning Network](#) è un protocollo di pagamento di livello 2 (layer 2 protocol) che opera al di sopra della blockchain di Bitcoin.

L'obiettivo di Lightning Network è quello di consentire (micro)transazioni veloci e economiche, ed è stato proposto come soluzione al problema della scalabilità e delle commissioni troppo alte che non permettono transazioni di piccole dimensioni in bitcoin.

L'idea è quella di creare dei **canali bidirezionali di pagamento off-chain (payment channels)**, sui quali vengono effettuate le transazioni senza che vengano scaricate sulla blockchain.

Payment Channel (Canale di pagamento)

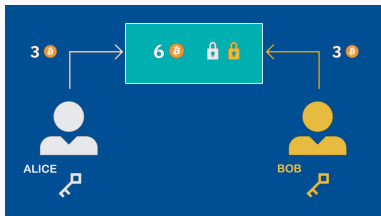
Supponiamo che Alice e Bob vogliano aprire un canale di pagamento. Allora, devono procedere nel seguente modo:

- **Transazione di apertura (del canale):** è una transazione multi-firma sulla blockchain che immobilizza dei fondi (nel senso che per spenderli serve sia la firma di Alice, sia quella di Bob). Questa transazione apre il canale;
- **Transazioni off chain:** Alice e Bob possono fare tutte le transazioni che vogliono tra di loro, senza che vengano inserite sulla blockchain (e quindi senza le relative commissioni);
- **Transazione di chiusura (del canale):** quando Alice e Bob decidono di chiudere il canale, calcolano il saldo di tutte le transazioni effettuate off chain ed eseguono una transazione di chiusura sulla blockchain.

Transazione di apertura di un canale di pagamento

Supponiamo che Alice e Bob si accordino per investire nel canale 3 BTC a testa.

Essi eseguono un transazione con 2 input (3 BTC da Alice, e 3 BTC da Bob), e con un singolo indirizzo bitcoin condiviso tra Alice e Bob come output; quest'ultimo, per essere usato come input in una futura transazione, necessiterà di entrambe le firme di Alice e di Bob.



Transazioni off-chain e transazione di chiusura

Da questo momento, Alice e Bob possono scambiarsi bitcoin liberamente e velocemente, in quanto nulla viene inserito sulla blockchain.

Supponiamo, per esempio, che Alice e Bob effettuino i seguenti scambi:

Alice → Bob 0.3 BTC

Alice → Bob 1.2 BTC

Bob → Alice 0.5 BTC

Alice → Bob 0.2 BTC

Bob → Alice 0.4 BTC

Se Alice e Bob decidessero di chiudere il canale, li basterebbe fare una transazione finale sulla blockchain di bitcoin, con input l'indirizzo multi-firma di apertura, e con output due address, un indirizzo bitcoin di Alice che riceverà 3.8 BTC, e un indirizzo bitcoin di Bob che riceverà 2.2 BTC, in modo da pareggiare i conti.

Hash Timelock Contracts (HTLCs)

Il sistema descritto funziona solo se Alice e Bob si conoscono e si fidano l'uno dell'altro. Per avere un protocollo che funzioni più in generale, bisogna utilizzare delle tecniche più sofisticate, ad esempio i **contratti Hash Timelock** (HTLC).

I *contratti Hash Timelock* (HTLCs) sono transazioni sulla rete Bitcoin che possono essere utilizzati per creare pagamenti condizionati: il destinatario, per utilizzare i fondi ricevuti, deve fornire un determinato segreto (mostrando un testo che ha una hash stabilita) oppure deve aspettare un certo intervallo di tempo.

Torniamo all'esempio di canale di pagamento tra Alice e Bob.

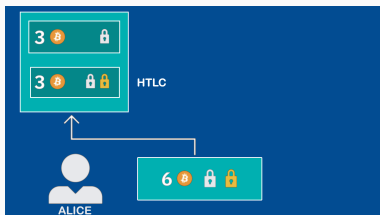
Protocollo sicuro per creare un canale di pagamento I

1. Alice e Bob generano casualmente dei testi segreti $secret_A$ e $secret_B$.
2. Alice e Bob si scambiano i *digest* $H(secret_A)$ e $H(secret_B)$.
3. Alice crea e manda a Bob una transazione HTLC con 2 output: 3 BTC verso un suo indirizzo bitcoin, e 3 BTC verso un indirizzo multi-firma 2-su-2 con Bob, firmata digitalmente (per ora non è finalizzata).
4. Bob analogamente crea e manda a Alice una transazione HTLC con 2 output: 3 BTC verso un suo indirizzo bitcoin, e 3 BTC verso un indirizzo multi-firma 2-su-2 con Alice, firmata digitalmente (per ora non è finalizzata).
5. Alice e Bob possono firmare la transazione di apertura del canale, a questo punto in modo sicuro.

Protocollo sicuro per creare un canale di pagamento II

Le transazioni ai punti 3 e 4 non sono per ora finalizzate, perché firmate da solo uno dei due utenti. Non serve mandarle alla blockchain, servono solo per poter recuperare i soldi nel caso l'altro utente non collabori. Entrambi però posso in qualsiasi momento finalizzare la transazione ricevuta dall'altro utente, firmandola digitalmente e mandandola alla blockchain di Bitcoin.

Entrambe le transazioni ai punti 3 e 4 sono HTLC e quindi sono soggette a vincoli di tempo e di conoscenza di un segreto.



Protocollo sicuro per creare un canale di pagamento III

Questa transazione è HTLC. L'output multi-firma della transazione, firmata da Alice e mandata a Bob, può essere spesa nelle seguenti condizioni:

1. **Alice ed Bob cooperano ed entrambi firmano.**
2. **da Bob, ma solo dopo un certo periodo di tempo (dovuto al timelock).**
3. **da Alice, ma solo se dimostra di conoscere $secret_B$.**

Analogamente, la transazione creata da Bob e mandata ad Alice ha i propri vincoli di tempo e di conoscenza del segreto.

Protocollo sicuro per creare un canale di pagamento IV

A questo punto, Alice e Bob possono firmare in perfetta sicurezza la transazione di apertura del canale di pagamento e mandarla alla blockchain.

Questo perché, se non c'è accordo nel gestire l'indirizzo multi-firma, ognuno può recuperare quanto ha investito grazie alle due transazioni non finalizzate che si sono scambiati i due utenti ai punti 3 e 4.

Se Alice e Bob cooperano, firmano entrambi l'output (multi-firma 2-su-2) della transazione di fondazione del canale e utilizzano i fondi in modo condiviso.

Se invece uno dei due si rifiuta di firmare l'output multi-firma, i soldi non rimangono bloccati per sempre, come nel caso semplificato.

Dopo avere aperto il canale di pagamento, supponiamo che Alice voglia trasferire 1 BTC a Bob. In tal caso:

- Alice e Bob condividono i loro segreti, in questo modo entrambi possono annullare le precedenti transazioni.
- Alice e Bob generano due nuovi segreti.
- Alice e Bob ripetono la procedura delle 2 transazioni firmate da uno solo e mandate all'altro, però questa volta Alice manda solo 2 BTC a se stessa e 4 alla multi-firma, mentre Bob ne invia 4 a se stesso e 2 alla multi-firma.
- A questo punto, la transazione di 1 BTC da Alice a Bob è stata formalizzata sul canale di pagamento.

Chiusura del canale di pagamento

Supponiamo che Alice voglia rimangiarsi l'impegno di aver trasferito 1 BTC a Bob.

In tal caso, Bob finalizza la transazione ricevuta da Alice e la manda alla blockchain, restituendo 2 BTC ad Alice e, atteso il tempo dovuto, si riprende i suoi 4 BTC, rendendo definitivo il passaggio di 1 BTC da Alice a Bob.

Anche Alice se vuole può chiudere il canale: firma la transazione ricevuta da Bob, la manda alla blockchain, trasferendo quindi 4 BTC a Bob e atteso il tempo dovuto si riprende i suoi 2 BTC, chiudendo il canale.

C'è anche un incentivo a tenere aperto un canale di pagamento: una piccolissima fee per ogni operazione gestita dal canale. Le commissioni molto basse del canale di pagamento permettono di gestire i micro pagamenti.

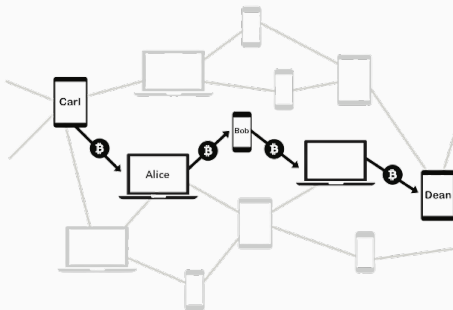
Una rete di canali di pagamento I

Un singolo canale di pagamento serve a poco, a meno che i due utenti non abbiano la necessità di fare tante transazioni tra di loro.

L'idea diventa molto più interessante quando ci sono **tante coppie di utenti che hanno aperto un canale tra di loro**, creando così un network che diventa diffuso e capillare, in modo tale che **un qualsiasi utente di bitcoin risulta legato con un cammino di canali a un qualsiasi altro utente**.

Supponiamo che Carl voglia fare un pagamento a Dean: Lightning Network è in grado di trovare il cammino meno costoso (cioè, con commissioni minori), e di effettuare i bilanciamenti tra tutti i canali coinvolti, nel nostro caso compreso il canale tra Alice e Bob, in modo da spostare off-chain la somma voluta.

Una rete di canali di pagamento II



Le commissioni sono stabilite liberamente da chi crea il canale di pagamento, ma fissarle troppo alte ha come conseguenza che pochi pagamenti tendono a passare da quel canale, che diventa quindi molto poco redditizio.

Viceversa, il canale che fissa commissioni molto basse servirà una grande quantità di transazioni, guadagnando molto poco per ogni transazione.

Difetti di Lightning Network

- Limitata capacità dei canali di pagamento: in ogni momento lo sbilanciamento del canale deve essere inferiore al capitale iniziale apportato da ognuno dei due creatori e l'intera rete è limitata dalla capacità dei singoli canali. Lightning Network è adatta solo a micropagamenti.
- I canali di pagamento sono soggetti a sbilanciamento: una volta che è stato raggiunto il massimo sbilanciamento in un senso, il canale funziona solo nell'altra direzione.
- I nodi che si disconnettono interrompono un canale e limitano la rete.
- Serve investire una grande liquidità: c'è il rischio che si creino dei grandi hub e quindi si vada verso la centralizzazione.

Velocità di Lightning Network

Con l'attuale rete di nodi di Lightning Network, il numero di operazioni al secondo gestibili dalla rete tende a essere sensibilmente più alto di quello di Bitcoin.

Se la rete si amplierà, sarà possibile in futuro avere anche una portata maggiore: tuttavia, per ora il Lightning Network non è ancora sufficiente per avvicinare Bitcoin agli altri sistemi di pagamento più tradizionali.

Pagamenti al secondo in diversi sistemi di pagamento



In questa slide riportiamo le risorse citate durante la lezione:

- [Bitcoin fork](#)
- [Bitcoin core GitHub repo](#)
- [Impatto ambientale Bitcoin](#)
- [StratumV2 \(mining pool\)](#)
- [Mastering Lightning Network](#)
- [Eltoo channel \(multi-channel\)](#)

Approfondimento: crittografia e address

Bitcoin utilizza la curva ellittica [Secp256k1](#).

Più in dettaglio:

- **Curva** : $y^2 = x^3 + 7$.
- **Campo** : \mathbb{Z}_q con $q = 2^{256} - 2^{32} - 2^9 - 2^8 - 2^7 - 2^6 - 2^4 - 1$.
- **Punto base** : $G = (x, y)$, $x =$
55066263022277343669578718895168534326250603453777594175500187360389116729240
 $y =$
32670510020758816978083085130507043184471273380659243275938904335757337482424
- **Numero di punti della curva** : $n =$
115792089237316195423570985008687907852837564279074904382605163141518161494337

q, x, y e n sono a 256 bit, circa 78 cifre decimali.

Quando Bob vuole creare una coppia chiave privata/chiave pubblica, sceglie semplicemente un valore intero molto grande d_B (chiave privata) in modo casuale, e quindi calcola la chiave pubblica corrispondente $P_B = d_B G = G + G + \dots + G$ (d_B volte).

La chiave pubblica non è un numero intero, ma un punto sulla curva, cioè una coppia di valori (x, y) che verifica l'equazione $y^2 = x^3 + 7$ modulo q .

Teniamo a mente che è possibile calcolare rapidamente la chiave pubblica a partire dalla chiave privata, ma il contrario è un problema molto difficile, noto come problema del logaritmo discreto sulle curve ellittiche (ECDLP).

Infine, si ottiene un indirizzo Bitcoin utilizzando la codifica *Base58Check* o *Bech32*.

Un address bitcoin viene calcolato a partire dalla chiave pubblica. Quest'ultima viene hashata due volte, e viene poi calcolato un checksum che è concatenato al *digest* precedente.

Infine, per ottenere un indirizzo Bitcoin, si utilizza la codifica *Base58Check* o *Bech32*.

Ulteriori informazioni sulle codifiche si possono trovare ai seguenti link:

- [Base58Check](#).
- [Bech32](#).

A titolo di esempio, vediamo come funziona la codifica *Base58*, prendendo in considerazione un address Bitcoin prima e dopo la sua codifica.

Prendiamo in considerazione la chiave pubblica (in esadecimale)

025c0de3b9c8ab18dd04e3511243ec2952002dbfadc...
...864b9628910169d9b9b00ec.

Il suo hash (in esadecimale) è

09c6e71118d8f12bec6b5c61884b35677c0a0ae3.

La codifica in *Base58* permette di utilizzare tutte le lettere minuscole e maiuscole e tutti i numeri tra 0 e 9, tranne un paio di coppie che si possono confondere facilmente: 0 e O, I e l.

L'hash dopo la codifica rappresenta l'address Bitcoin ed è

1thMirt546nngXqyPEz532S8fLwbozud8.

C'è anche uno svantaggio nell'utilizzare l'indirizzo Bitcoin invece della chiave pubblica direttamente. Due chiavi pubbliche diverse potrebbero avere lo stesso indirizzo Bitcoin, e quindi un utente diverso da quello a cui si desidera inviare la transazione potrebbe essere in grado di spendere questi bitcoin.

La probabilità che ciò accada è tuttavia trascurabile. Attualmente, ci sono in media 500.000 transazioni al giorno, quindi, mantenendo questo ritmo, nei prossimi 100 anni ci saranno circa 18 miliardi di transazioni. La probabilità di avere due indirizzi Bitcoin identici (cioè avere due chiavi pubbliche diverse il cui *digest* è lo stesso) sarebbe inferiore a 10^{-35} .

Approfondimento: wallet

In un *Hierarchical Deterministic* (HD) wallet, ogni chiave è derivata da un'altra chiave ottenuta in precedenza attraverso l'uso di funzioni hash.

- Lo standard più avanzato che descrive questo tipo di wallet è lo standard [BIP-32](#) (BIP è un acronimo e sta per Bitcoin Improvement Proposal). Da ogni “chiave genitore” è possibile derivare una o più “chiavi figlie” e così via, fino a una profondità potenzialmente illimitata.
- La struttura ad albero può essere utilizzata per una migliore organizzazione di tutte le chiavi.
- È possibile costruire una sequenza di chiavi pubbliche valide senza conoscere le chiavi private corrispondenti. Questo consente di generare chiavi anche su un server non sicuro.

La sicurezza di un wallet dipende dalla corretta implementazione degli algoritmi utilizzati. Esistono diversi standard, descritti dalle proposte di miglioramento di Bitcoin (BIP), che definiscono come dovrebbero essere implementate queste funzionalità.

- Lo standard [BIP-39](#) descrive il processo di creazione della sequenza di parole in lingua inglese da cui viene poi derivato il *seed*.
- Lo standard **BIP-32** permette la generazione di qualsiasi numero di chiavi a partire dal *seed* in modo assolutamente deterministico.
- Altri standard come [BIP-43](#) e [BIP-44](#) forniscono ulteriori dettagli su come organizzare le chiavi in un wallet.

Questi standard sono utilizzati dalla maggior parte dei wallet, rendendoli interoperabili: un utente può cambiare wallet semplicemente mantenendo a mente il proprio *seed*.

HD Wallet

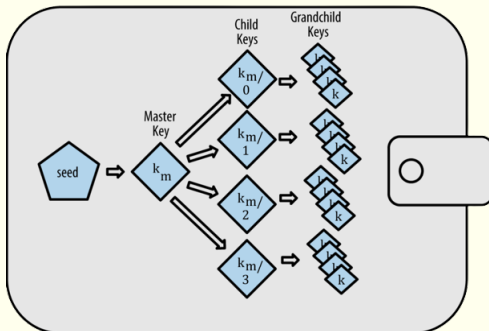


Figure 6: HD wallet

Approfondimento: le transazioni P2PKH e P2SH

- Gli *script* contenuti negli input e negli output di una transazione Bitcoin sono scritti in un linguaggio di *scripting* chiamato **Script**. È simile ai linguaggi di programmazione degli anni '60.
- Il linguaggio scelto è molto semplice e ha capacità limitate, ma offre comunque la possibilità di scrivere condizioni complesse. In particolare, *script* non è *Turing completo*, il che significa che non è possibile creare cicli ma solo flussi condizionali (if). Ciò rende prevedibile il tempo di esecuzione di uno *script*.
- Quando una transazione viene convalidata, gli *script* vengono eseguiti in sequenza (iniziando sempre con lo *script* di sblocco) e viene convalidata se le condizioni dello *script* di blocco vengono soddisfatte. Ciascun input viene convalidato in modo indipendente.

- Il linguaggio di *scripting* Bitcoin si basa sull'uso di *pila*. Sono possibili due operazioni: *push*, che inserisce un oggetto in cima alla pila, e *pop*, che rimuove l'oggetto in cima alla pila (struttura dati LIFO).
- Gli *operatori* sono indicati come *OP_X*, dove X indica l'operazione da eseguire sugli elementi in cima alla pila. Non appena un operatore viene inserito in una pila, l'operazione a cui si riferisce deve essere eseguita immediatamente.
- Ad esempio, *OP_ADD* prende i due elementi in cima alla pila, li somma e inserisce il nuovo risultato in cima alla pila, mentre *OP_EQUAL* confronta i due elementi in cima alla pila e inserisce *TRUE* o *FALSE* a seconda del caso.
- Lo *script* di blocco viene soddisfatto solo se alla fine rimane un unico valore nello stack, ovvero *TRUE*.

La maggior parte delle transazioni consuma output bloccati tramite uno *script* **Pay-to-Public-Key-Hash** (P2PKH). Questo *script* di blocco ha la seguente struttura:

OP_DUP OP_HASH160 < Indirizzo >

OP_EQUALVERIFY OP_CHECKSIG.

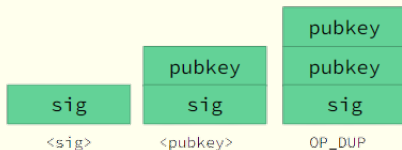
- Lo *script* di blocco descritto può sempre essere soddisfatto da uno *script* di sblocco con la seguente struttura:

Firma ChiavePubblica.

Pay-to-Public-Key-Hash II

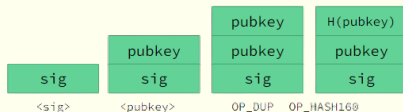
Verifichiamo la correttezza del processo di sblocco:

- Innanzitutto, inseriamo i contenuti dello *script* di sblocco nello stack. Solo allora possiamo passare ai vari operatori dello *script* di blocco. OP_DUP duplica l'elemento in cima allo stack, che al primo passo sarà quindi:



Pay-to-Public-Key-Hash III

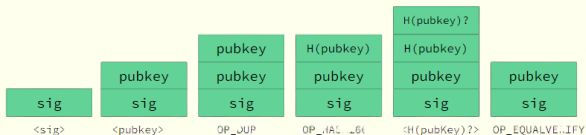
- Il secondo operatore nello *script* di blocco è OP_HASH160, che prende l'oggetto in cima allo stack e applica in sequenza le funzioni di hash SHA256 e RIPEMD160. Se l'oggetto in questione è una chiave pubblica, otteniamo un address:



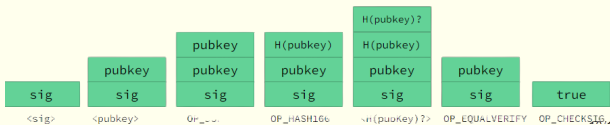
Procedendo con il locking *script*, a questo punto inseriamo l'address in cima alla pila.

Pay-to-Public-Key-Hash IV

- Se sono uguali, vengono rimossi entrambi dallo stack.



- Infine, `OP_CHECKSIG` verifica che la firma inserita nello stack sia valida, utilizzando la chiave pubblica fornita in input. Se lo è, restituisce `TRUE`.



Il protocollo Bitcoin offre la possibilità di scrivere transazioni con uno *script* di blocco più complesso. Un esempio è lo *script*

Pay-to-Script-Hash (P2SH).

- P2SH è stato introdotto nel 2012 e semplifica notevolmente l'uso di *script* di transazione più complessi. Viene solitamente utilizzato per la creazione di *script* multi-firma (fino a $N = 3$).
- Questi *script* possono essere considerati come gli smart contract di Bitcoin. A differenza dei più famosi smart contract di Ethereum, sono più semplici ma anche molto più sicuri.
- In uno *script* P2SH, lo *script* di blocco ha la seguente struttura:

OP_HASH160 < digest dello script >, OP_EQUAL.

Pay-to-Script-Hash I


Indice	0	Dettagli	Trascorso
Indirizzo	3P14159f73E4gFr7JterCCQh9QjiTjZrG 	Valore	0.03000000 BTC
Pkscript	OP_HASH160 e9c3dd0c07aac76179ebc76a6c78d4d67c6c160a OP_EQUAL		

Figure 7: Script di output P2SH


Indice	0	Dettagli	Produzione
Indirizzo	3P14159f73E4gFr7JterCCQh9QjiTjZrG 	Valore	0.03000000 BTC
Pkscript	OP_HASH160 e9c3dd0c07aac76179ebc76a6c78d4d67c6c160a OP_EQUAL		
Sigscript	OP_0 304402201c6e4fbdcf8c312c1818a4326390ccf34cde63fa35277b8a88c6ed80f84f509022068d61f4fa85d7835698c5a64dc5f2a7d056021b249006f566be926f5c408a1fb01 5141042f90074d7a5bf30c72cf3a8dfd1381bdbc30407010e878f3a11269d5f74a58788505cdca22ea6eab7cfb40dc0e07aba200424ab0d79122a653ad0c7ec9896bdf51ae		
Witness	N/A		

Figure 8: Script di input P2SH

Per poter sbloccare il contenuto di un UTXO, è necessario fornire lo *script* corretto, il cui hash è esattamente uguale a <digest dello script>.

- In questo caso, lo *script* di sblocco ha la seguente struttura:

OP_0 Sig1 ... SigM < script > .

- OP_0 è inserito per risolvere un bug nelle regole di consenso, mentre i successivi M campi contengono le firme digitali. Infine, l'ultimo campo contiene lo *script* cercato, il cui hash restituisce <digest dello script>.

Segregated Witness I

- **Segregated Witness** (SegWit) è un aggiornamento delle regole di consenso del protocollo Bitcoin, attivato a partire dal 1° agosto 2017.
- In generale, per Bitcoin, qualsiasi soluzione in grado di soddisfare la condizione imposta da un UTXO è chiamata *witness* (testimone).
- L'idea di SegWit è separare il testimone dallo *script* di sblocco a cui appartiene, spostando questo campo in una struttura dati speciale che accompagna la transazione, chiamata appunto *witness*.

Segregated Witness II

L'aggiornamento offre diversi vantaggi per la scalabilità, la sicurezza e le prestazioni di Bitcoin:

- *Versione del programma*: con l'introduzione di SegWit, ciascuno *script* di blocco è preceduto da un *numero di versione*. L'aggiunta di questo campo semplifica gli aggiornamenti futuri.
- *Scalabilità della rete*: la firma digitale occupa una grande quantità di spazio all'interno di una transazione. Spostandola altrove si guadagna spazio, migliorando così la scalabilità.

Segregated Witness NON è un cambiamento a livello di transazione, ma piuttosto un cambiamento a livello di output, poiché modifica come gli UTXO individuali possono essere spesi.

La codifica utilizzata per gli indirizzi è *Bech32*.

L'aggiornamento **Taproot**, introdotto il 14 novembre 2021, è composto da tre diversi BIP:

- *BIP340 - Schnorr*: protocollo di firma digitale.
- *BIP341 - Taproot*: migliora la privacy, l'efficienza e la flessibilità del linguaggio di *scripting* di Bitcoin.
- *BIP342 - Tapscript*: aggiornamento del linguaggio di *scripting*.

Esattamente come SegWit, Taproot utilizza la codifica *Bech32* per i suoi indirizzi.

Taproot II

L'idea chiave dietro l'aggiornamento è quella di combinare i due *script* P2PKH e P2SH in un nuovo, singolo *script*, chiamato **Pay-to-Taproot** (P2TR).

Taproot deve essere utilizzato in concomitanza con:

- **Firma di Schnorr**, perché facilita la creazione di firme digitali multi-firma.
- **Merkalized Alternative Script Tree (MAST)**, una costruzione crittografica derivata dai *Merkle trees* che consente di rivelare solo alcune parti di uno *script*.

I bitcoin protetti da uno *script* di blocco P2TR possono essere spesi in due modi: soddisfacendo gli *script* rivelati, o fornendo una firma digitale valida.