COESIONE ITALIA 21-27 PIEMONTE

Cofinanziato dall'Unione europea

REGIONE PIEMONTE

3 - Finanza decentralizzata e blockchain

Concetti di Economia e Finanza Decentralizzata

b.f. 2023-2025

Docente:

Enrico Zimuel

Intervento realizzato da ITS TECNOLOGIE DELL'INFORMAZIONE E DELLA COMUNICAZIONE

Programma

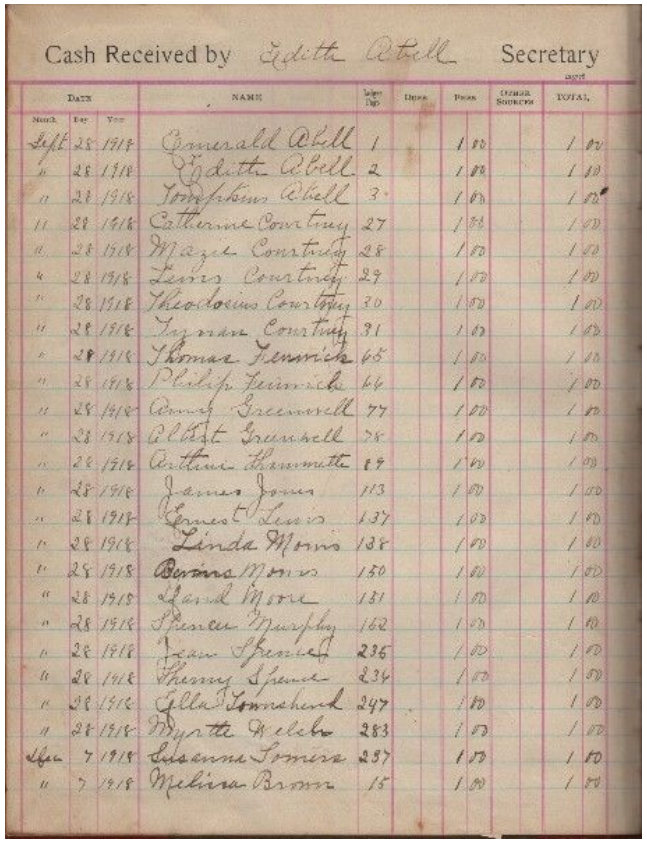
- Che cos'è la blockchain?
- Catena di blocchi
- Peer-to-peer
- Cenni storici: il movimento cypherpunk
- Com'è fatto un blocco in una blockchain?
- Funzioni hash
- Merkle tree
- Algoritmo di consenso Proof of Work
- Mining
- Crittografia a chiave pubblica
- Firma delle transazioni
- Indirizzi Bitcoin
- Double spending
- Validazione dei blocchi

Che cos'è la blockchain?

La **blockchain**, letteralmente una catena (chain) di blocchi (block) è un **registro, condiviso** e **immutabile**

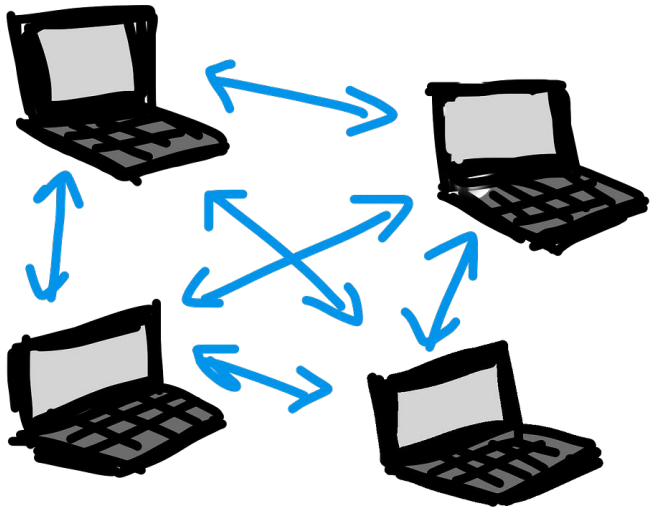
Registro

- Un **registro** è un archivio nel quale vengono memorizzate informazioni in ordine temporale
- In una blockchain le informazioni sono memorizzate in **blocchi** di dimensioni prestabilite (es. 1 MB)
- Ogni blocco viene memorizzato nella blockchain con una data (timestamp)



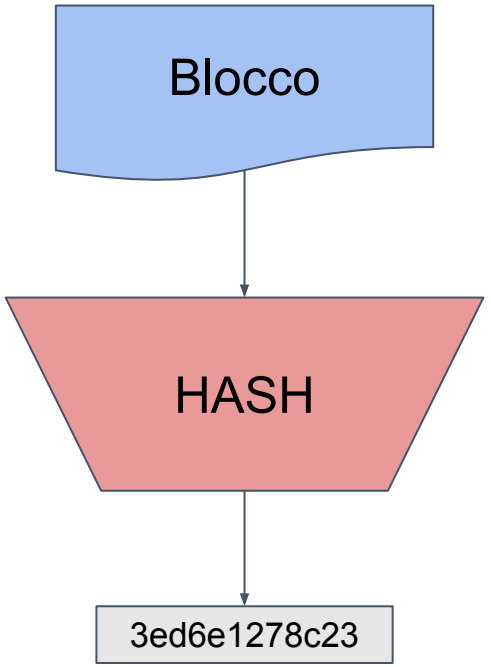
Condiviso

- Questo registro è **condiviso** pubblicamente tra tutti i nodi della rete blockchain
- Ogni nodo detiene la **stessa copia** del registro
- La rete dei nodi è di tipo **peer-to-peer**

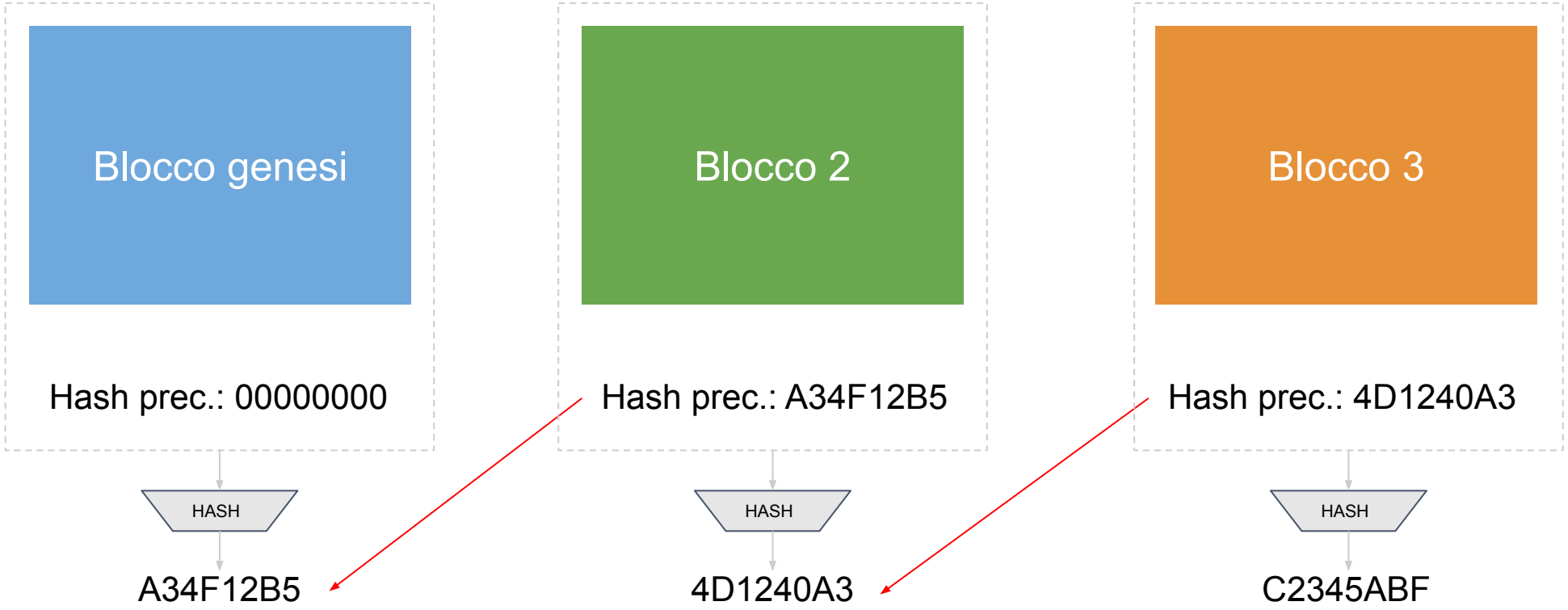


Immutabilità

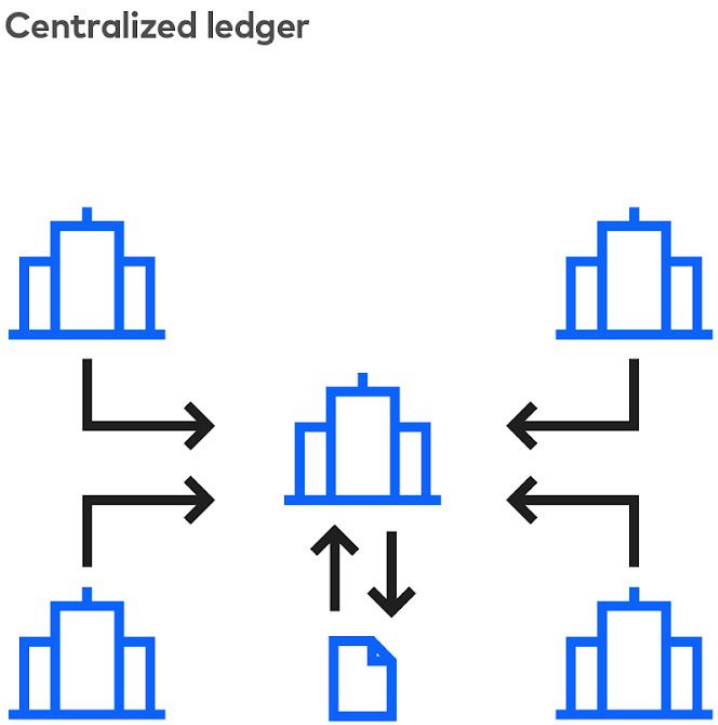
- Questo registro condiviso è **immutabile**,una volta che un blocco di dati viene memorizzato, non può essere più modificato
- I blocchi sono collegati tra di loro attraverso delle **funzioni hash**

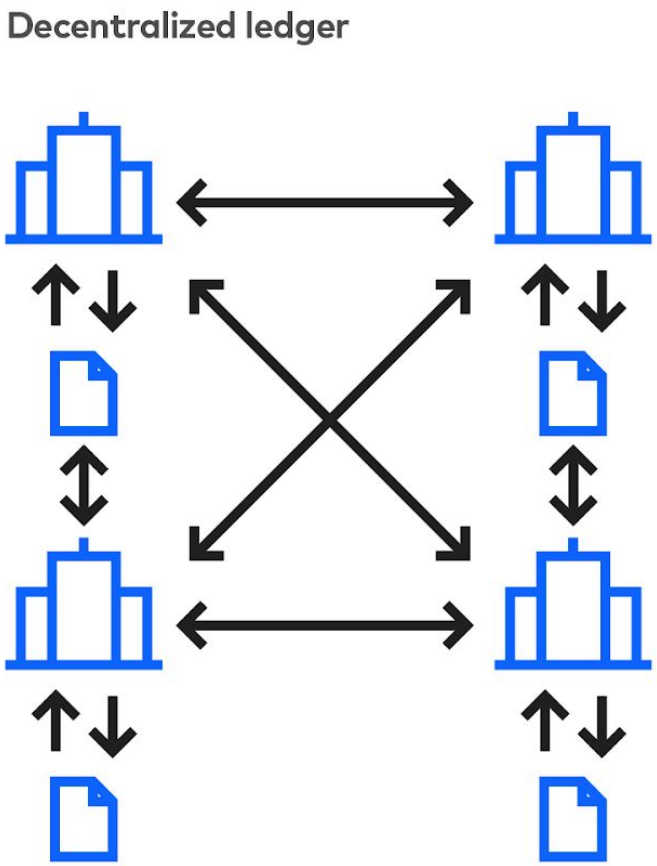


Catena di blocchi

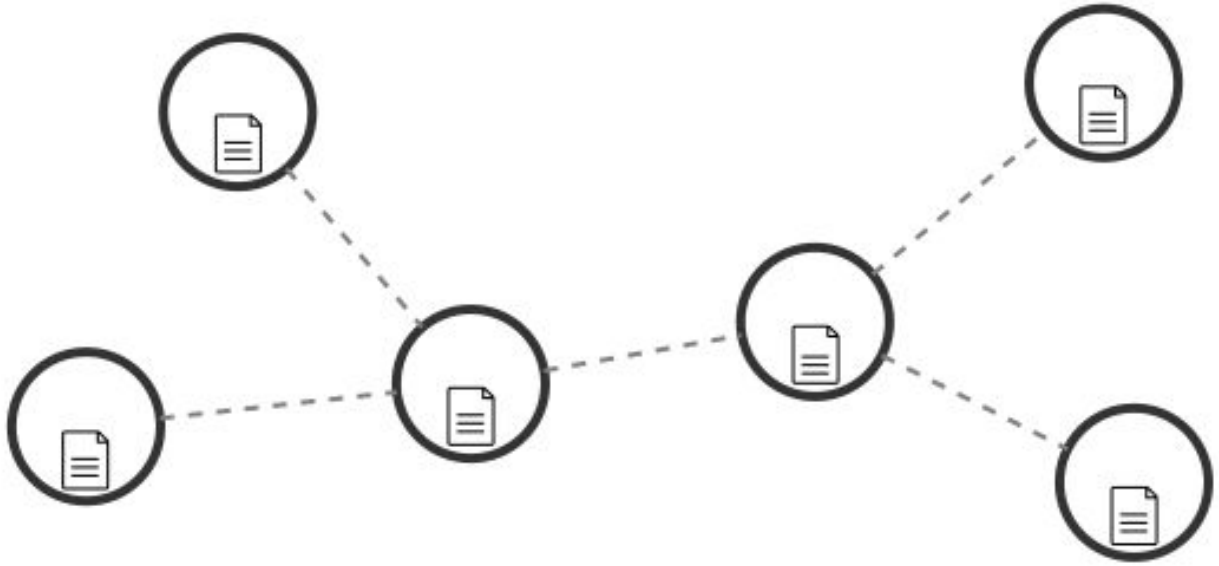


Registro decentralizzato





Peer-to-peer



L’8 Maggio 2024 2023 la blockchain di Bitcoin aveva una dimensione di circa 570 GB

Gossip

- Uno degli algoritmi peer-to-peer più utilizzati è l'algoritmo di **gossip**
- Ogni nodo sceglie a caso un altro nodo e invia le informazioni a quest'ultimo
- E' possibile dimostrare che se *n* è il numero di nodi di una rete, sono necessari mediamente *log(n)* iterazioni per nodo per raggiungere tutti i nodi della rete
- Simulatore dell'algoritmo gossip: https://ctufaro.github.io/GossipPlot/index.html

Origini della blockchain

- L'idea della **blockchain** nasce nel 2008 con l'articolo Bitcoin: A Peer-to-Peer Electronic Cash System di Satoshi Nakamoto (pseudonimo)
- Nakamoto utilizza il termine "block chain" per indicare il registro nel quale vengono memorizzate le transazioni di bitcoin
- L'articolo di Nakamoto è rivoluzionario perché introduce il concetto di **Proof Of Work** (PoW) per risolvere il problema del ***double spending*** su una rete **decentralizzata**

Prima di Bitcoin

- L'idea di creare una catena (chain) di informazioni collegate tramite hash non è stata ideata da Nakamoto
- L'idea originale appare per la prima volta nell'articolo How to Time-Stamp a Digital Document di Stuart Haber e Scott Stornetta del 1991
- Erano state proposte diverse monete elettroniche prima di Bitcoin come DigiCash (1982), B-Money (1998), Bit gold (1998)

Cypherpunk

- Movimento nato agli inizi degli anni ‘80 per promuovere l’utilizzo della crittografia come strumento di liberazione, per la protezione della privacy
- Mailing list **cypherpunk** nata nel 1992 per opera di Eric Hughes, Timothy C. May e John Gilmore
- *"Privacy is necessary for an open society in the electronic age. ... We cannot expect governments, corporations, or other large, faceless organizations to grant us privacy ... We must defend our own privacy if we expect to have any. ... Cypherpunks write code. We know that someone has to write software to defend privacy, and ... we're going to write it."*
  A Cypherpunk's Manifesto, Eric Hughes, 1993

bitcoin

- **bitcoin** è la prima criptovaluta ideata nel 2008
- Un bitcoin (₿) vale circa $ 62’132*
- Market cap $ 1,23* trilioni
- E’ la criptovaluta più importante; da sola rappresenta più del 50% del mercato
- Sito ufficiale del progetto: bitcoin.org

* Dato dell’8 Maggio 2024

Come funziona Bitcoin

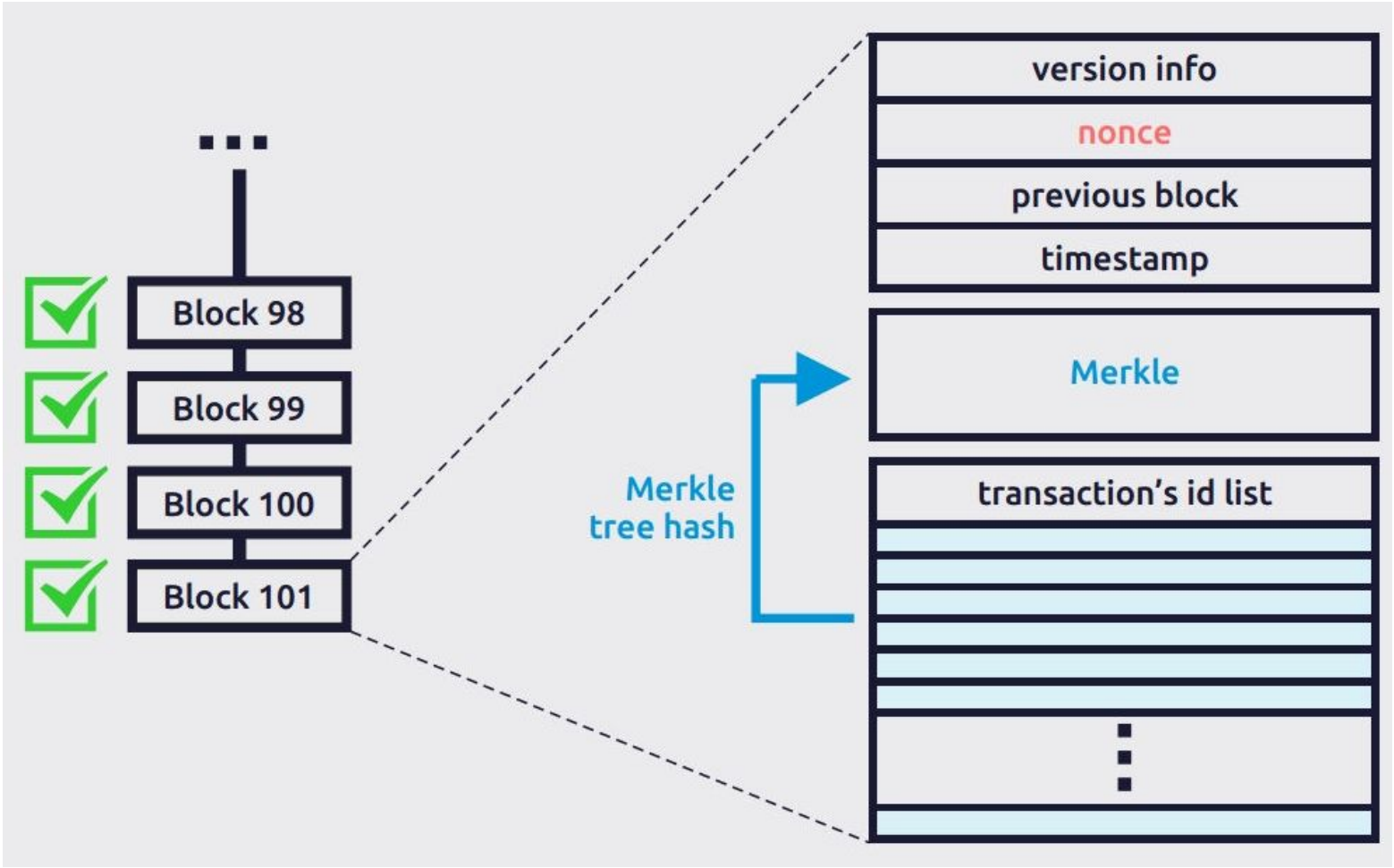
- In **Bitcoin**, come in tutte le criptovalute, ogni persona può avere uno o più indirizzi
- Questi indirizzi sono **anonimi**, non contengono informazioni sulla persona
- E' possibile trasferire bitcoin da un indirizzo a un altro (**transazione**)
- Le transazioni sono pubbliche, chiunque può verificare che un indirizzo A ha trasferito x bitcoin ad un indirizzo B in una certa data (per questo motivo si dice che Bitcoin garantisce un semi-anonimato)

Come funziona Bitcoin (2)

- I bitcoin vengono generati dai **miner**, ossia i nodi della rete che convalidano la creazione dei blocchi (ogni 10 minuti)
- Il protocollo Bitcoin è progettato in modo tale che nuovi bitcoin sono creati ad una velocità fissa
- E' previsto un numero massimo di bitcoin in circolazione, **21 milioni** (attualmente 19,7* milioni)

* Dato dell'8 maggio 2024

Blockchain



← **HASH** del blocco precedente

← **Merkel tree**

← **Transazioni**:
A invia 1₿ a B
D invia 0.05₿ a E
...

Fonte: https://devopedia.org/blockchain

Hash

- La blockchain è una **catena di dati** collegate tra di loro tramite l'utilizzo di una funzione **hash**
- Un hash è una **funzione non invertibile** che mappa una stringa di lunghezza arbitraria in una stringa di lunghezza predefinita (es. 32 bytes)
- Esempio: **SHA256("Hello World")** = a591a6d40bf420404a011733cfb7b190d62c65bf0bcda32b57b277d9ad9f146e
- Partendo dal risultato della funzione hash deve essere computazionalmente impossibile determinare la stringa originale

Proprietà di una funzione hash

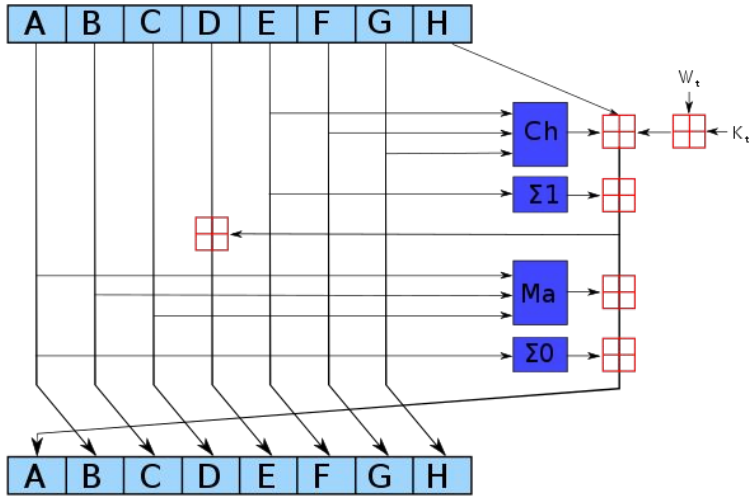
Una funzione hash deve rispettare le seguenti proprietà:

- È facile calcolare l'hash di un messaggio
- È computazionalmente impossibile calcolare il messaggio che ha generato un dato hash (**non invertibilità**)
- È computazionalmente difficile modificare un messaggio senza modificare il relativo hash (**resistenza debole alle collisioni**)
- Dato un messaggio è computazionalmente difficile trovarne un altro con lo stesso hash (**resistenza forte alle collisioni**)

SHA256

- Il messaggio viene suddiviso in blocchi di **512 bit**
- Questi blocchi vengono iterati con una funzione di compressione su un insieme di 8 valori da 32 bit (A, B, C, D, E, F, G, H)
- Il risultato è la concatenazione di questi blocchi (8 x 32 = 256 bit)
- Bitcoin utilizza **SHA256²** ossia SHA256(SHA256(x))

Funzione di compressione SHA-2

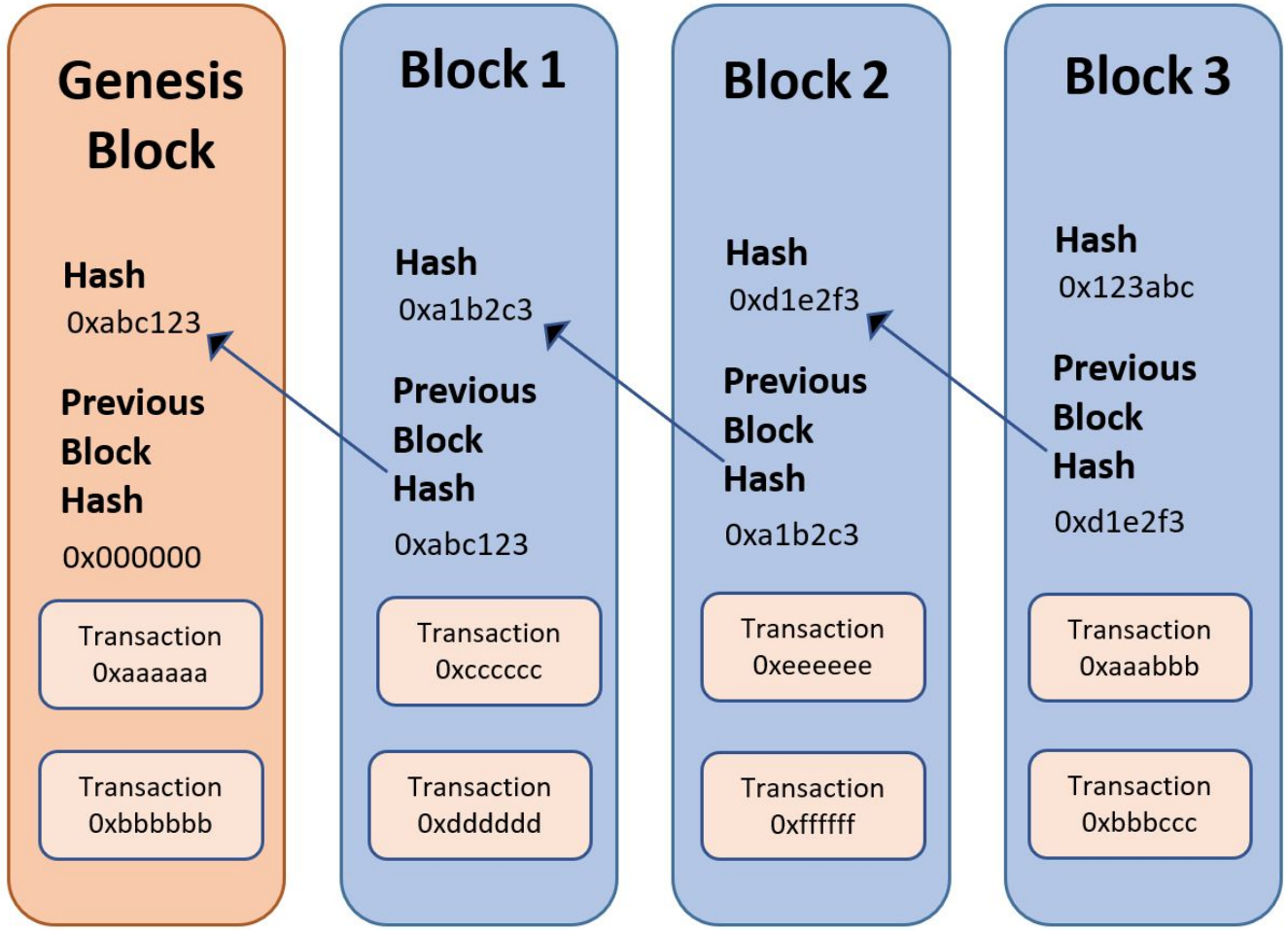


$$\mathrm{Ch}(E,F,G) = (E \wedge F) \oplus (\neg E \wedge G)$$
$$\mathrm{Ma}(A,B,C) = (A \wedge B) \oplus (A \wedge C) \oplus (B \wedge C)$$
$$\Sigma_0(A) = (A \ggg 2) \oplus (A \ggg 13) \oplus (A \ggg 22)$$
$$\Sigma_1(E) = (E \ggg 6) \oplus (E \ggg 11) \oplus (E \ggg 25)$$

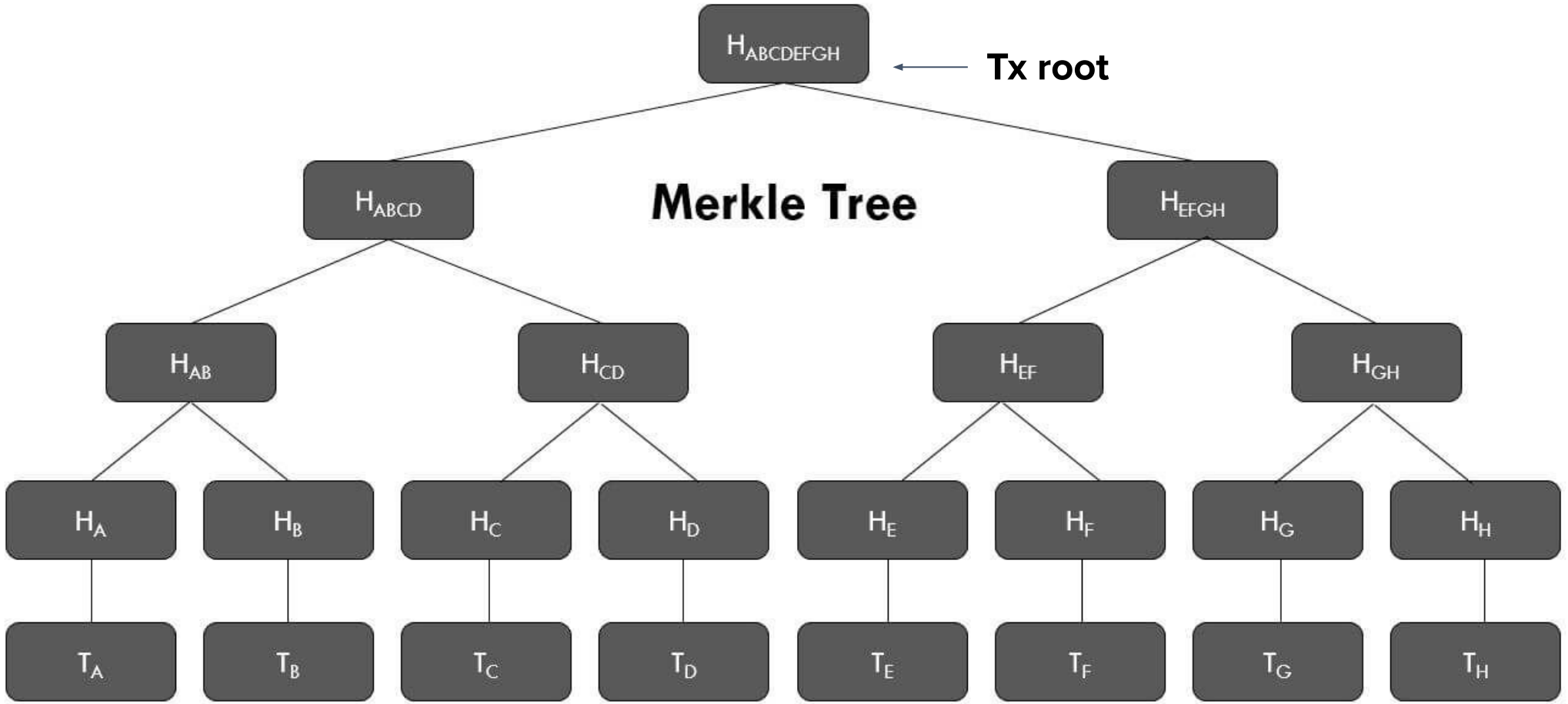
Esercizio

- Scrivere un programma in Python che sia in grado di leggere un file e calcolare le seguenti funzioni hash sul contenuto del file:
  - SHA256
  - $SHA256^2$

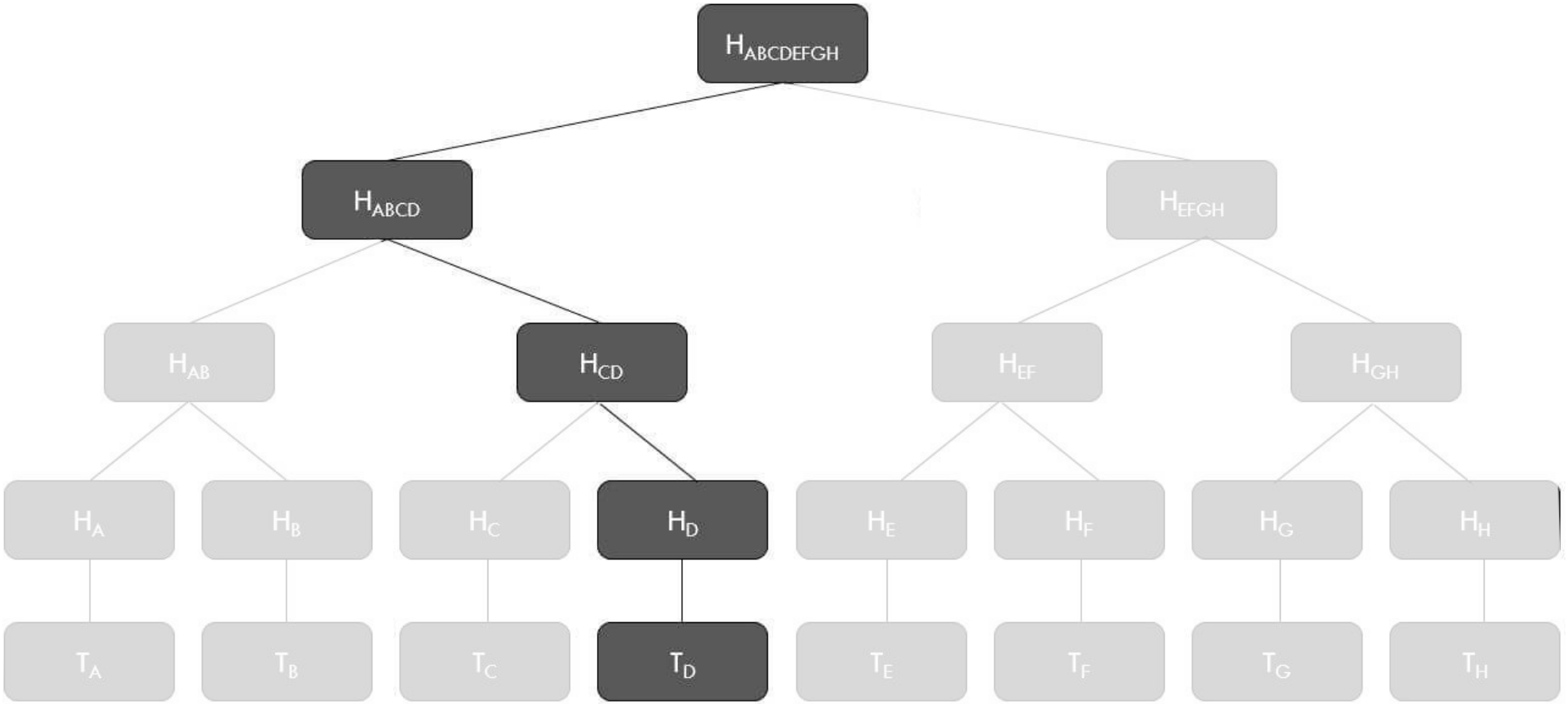
Blockchain



Merkle tree

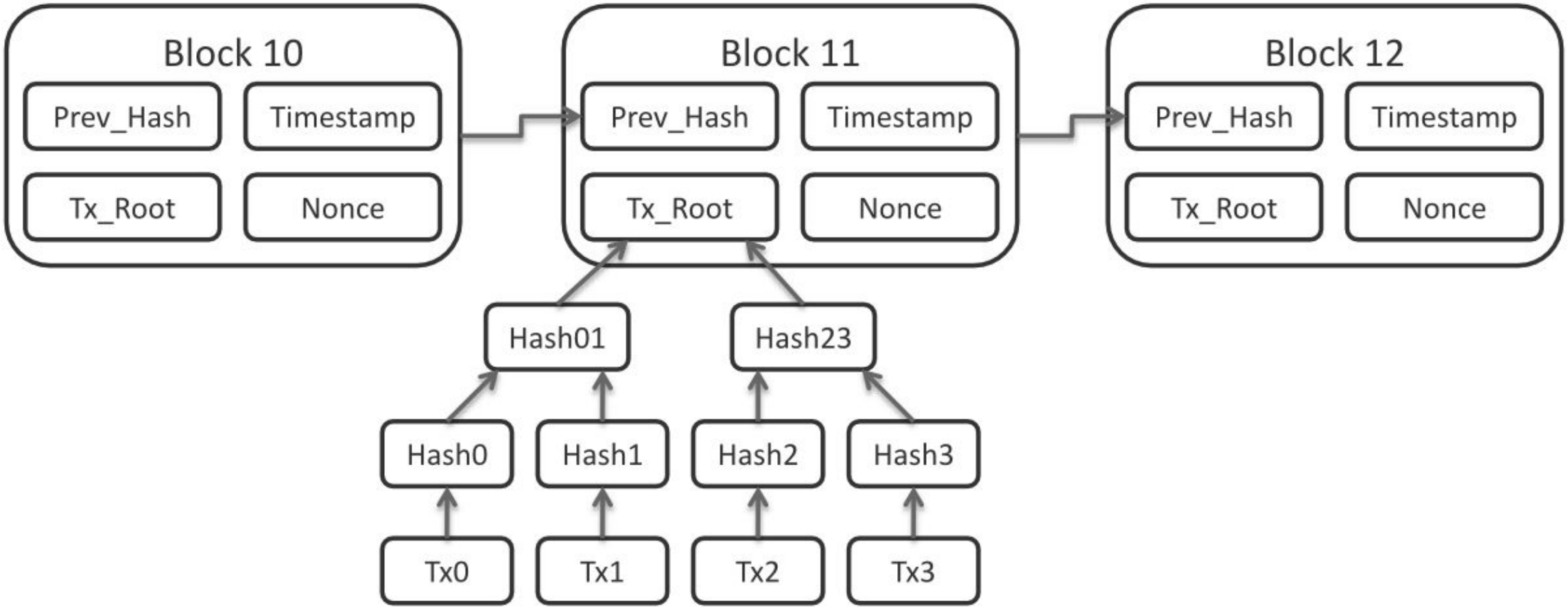


Verifica di una transazione



Se *n* è il numero di hash, ne servono solo ***log(n)*** per una verifica

Blockchain



Fonte: https://commons.wikimedia.org/wiki/File:Bitcoin_Block_Data.png

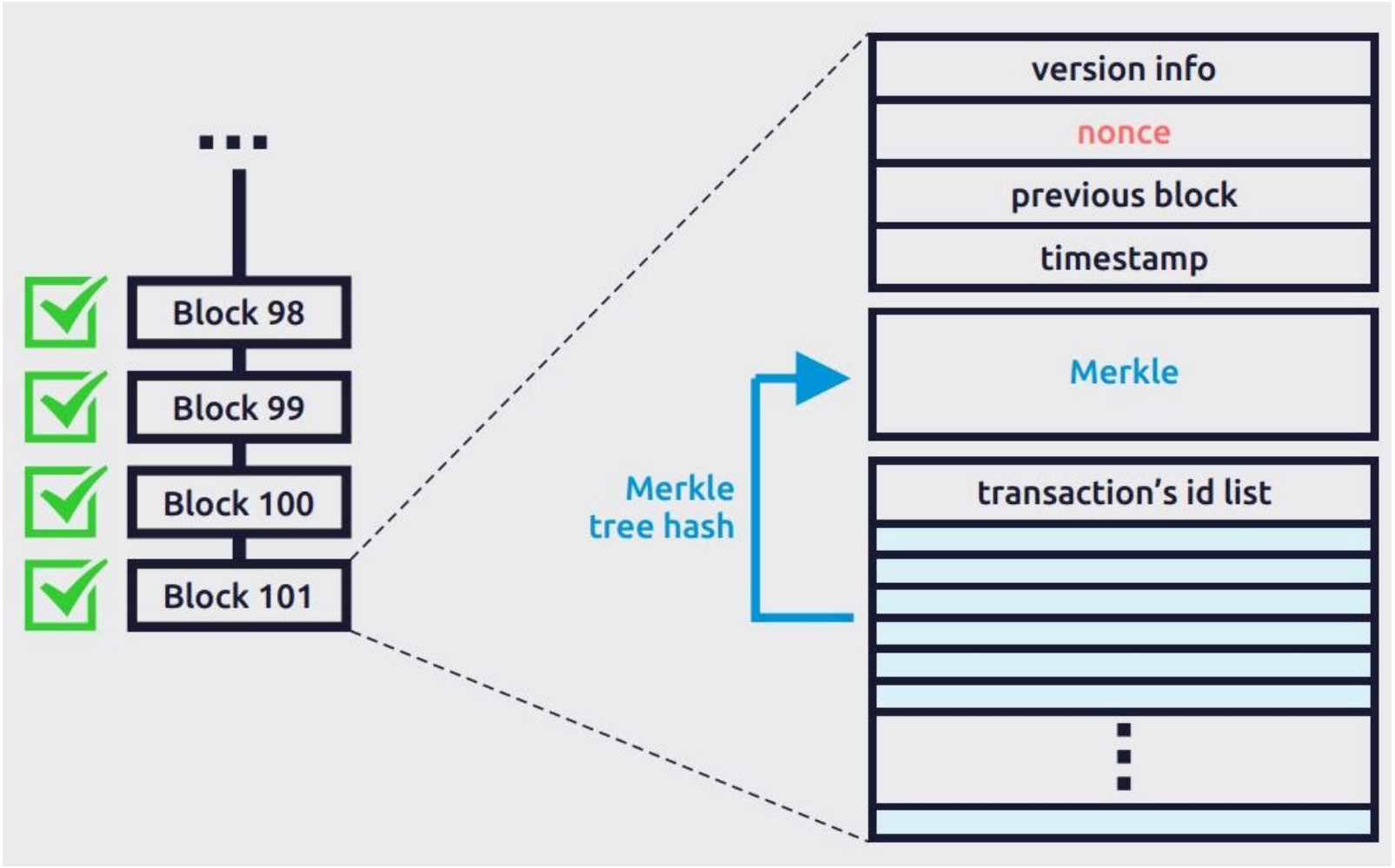
Esercizio

- Scrivere un programma in Python che sia in grado di leggere un insieme di file e calcolare il Merkle tree root utilizzando la funzione hash SHA256
- Nel caso in cui il numero dei file o dei nodi intermedi del Merkle tree non sia pari è necessario copiare l'ultimo nodo e aggiungerlo all'albero (in modo da avere sempre un numero pari di nodi su cui lavorare)

Proof of Work

- Modificare una blockchain deve essere un'operazione computazionalmente (molto) difficile
- Il **Proof of Work** (PoW) è un'operazione che richiede diverso tempo computazionale (es. un puzzle matematico)
- Il campo **nonce** in un blocco è un esempio di **PoW**, ossia è il valore che deve essere concatenato all'input per ottenere un hash < target
- Bitcoin utilizza l'algoritmo Hashcash

Nonce



Nonce: è il valore che concatenato alla stringa di partenza da come risultato un hash < target

Esempio di PoW

- Stringa in input “Hello, world!”
- Trovare il valore di **nonce** tale che **SHA256(“Hello, world!” + nonce) < 2^{240}**

```
nonce=-1
while (hash >= 2^240)
   nonce++
   hash = SHA256("Hello, world!" + nonce)
```

Esempio di PoW (2)

"Hello, world!0" => 1312af178c253f84028d480a6adc1e25e81caa44c749ec81976192e2ec934c64 = 2^252.253458683
"Hello, world!1" => e9afc424b79e4f6ab42d99c81156d3a17228d6e1eef4139be78e948a9332a7d8 = 2^255.868431117
"Hello, world!2" => ae37343a357a8297591625e7134cbea22f5928be8ca2a32aa475cf05fd4266b7 = 2^255.444730341
...
"Hello, world!4248" => 6e110d98b388e77e9c6f042ac6b497cec46660deef75a55ebc7cfdf65cc0b965 = 2^254.782233115
"Hello, world!4249" => c004190b822f1669cac8dc37e761cb73652e7832fb814565702245cf26ebb9e6 = 2^255.585082774
"Hello, world!4250" => 0000c3af42fc31103f1fdc0151fa747ff87349a4714df7cc52ea464e12dcd4e9 = 2^239.61238653

- nonce = **4250**
- Il target in questo esempio è 2^240
- Bitcoin resetta periodicamente il livello di difficoltà, ogni 2016 blocchi (circa 2 settimane), per tenere il tempo di creazione di un blocco ogni 10 minuti

Esercizio

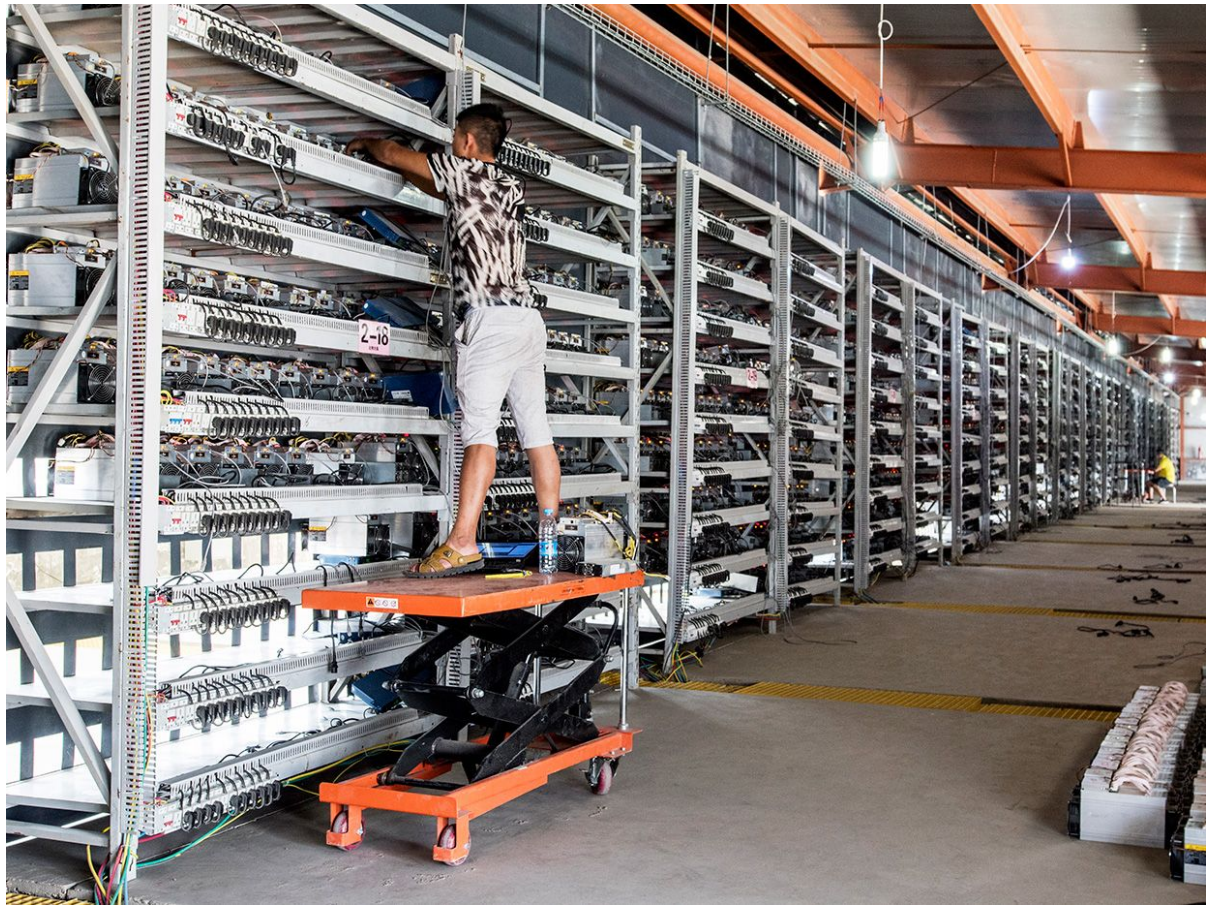
- Scrivere un programma in Python che sia in grado di calcolare il **nonce** di un blocco utilizzando l'algoritmo PoW presentato nelle slide precedenti.
- In particolare, questo programma deve accettare in **input** una stringa da verificare e il valore **target**
- Il nonce dovrà essere un valore tale che:
  SHA256(input || nonce) < target
  dove || è l'operazione di concatenazione di stringhe.

Mining

- I **miner** sono i nodi che risolvono i PoW
- Ogni volta che viene risolto un puzzle il nodo viene ricompensato (es. in bitcoin)
- Costo del mining (hardware + energia elettrica)
- Esempio:
  - Hardware: BITMAIN Antminer S17e (64Th), potenza di 2880W, costo ≈ $1000
  - Costo energia: 0.1 USD/kWH
  - Ricavo: $ 17.06 USD/giorno

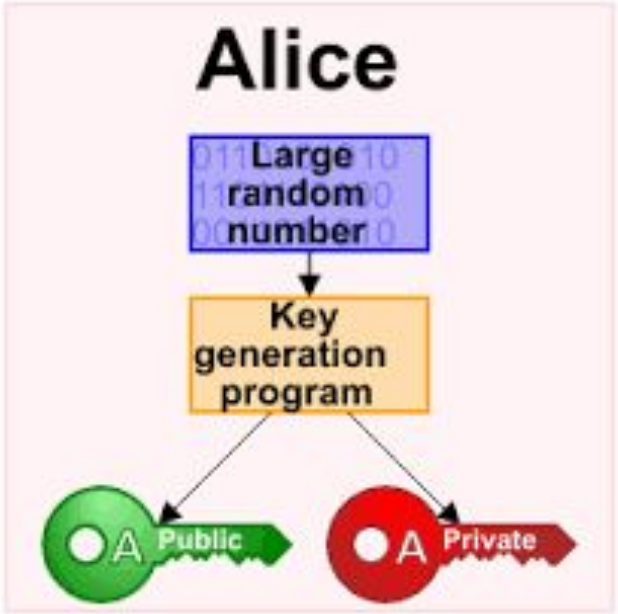
Fonte: Nicehash profitability calculator

Mining farm



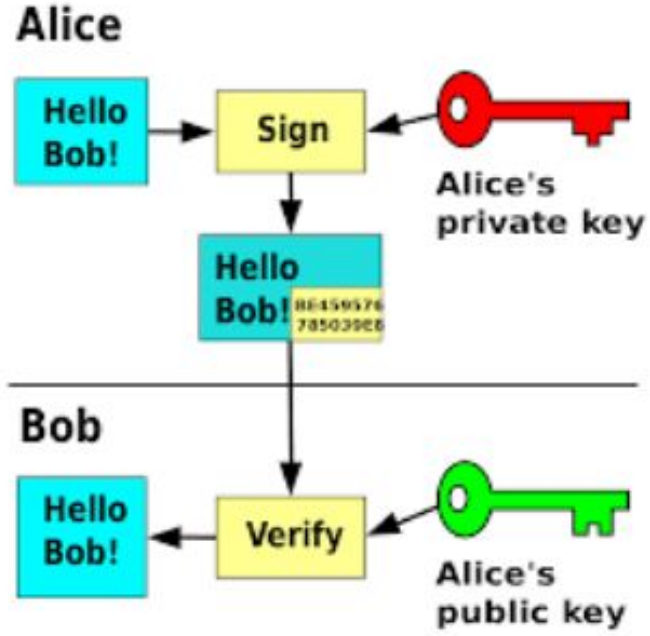
Crittografia a chiave pubblica

- Per garantire l'autenticità e il non ripudio dei dati memorizzati in una blockchain si utilizza la **crittografia a chiave pubblica**
- Ogni utente genera una coppia di chiavi: **privata** e **pubblica**
- La crittografia a chiave pubblica può essere utilizzata sia per proteggere (**cifrare**) che per autenticare informazioni (**firmare**)



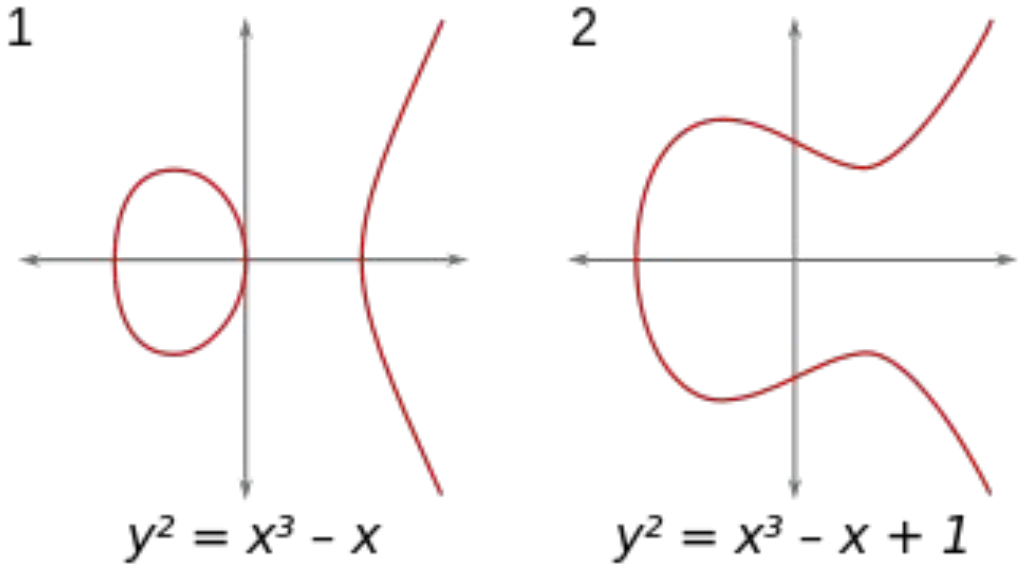
Firma

- La **chiave privata** viene utilizzata per **firmare** un dato
- La **chiave pubblica** viene utilizzata per **verificare** la firma
- In Bitcoin, un indirizzo è calcolato con l'hash della chiave pubblica, in particolare: ***ripemd160(SHA256(Public-key))***



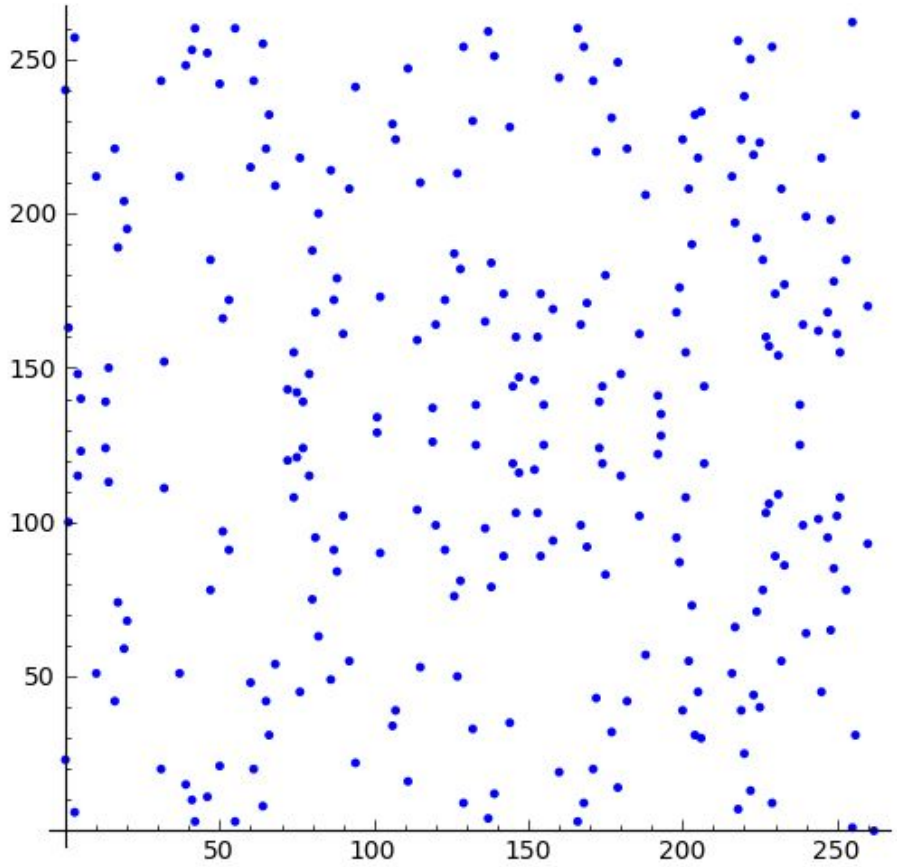
Curva ellittica

- Una curva ellittica è una funzione matematica particolarmente utilizzata in crittografia
- Una curva ellittica è la funzione $\mathbf{y^2 = x^3 + ax + b}$



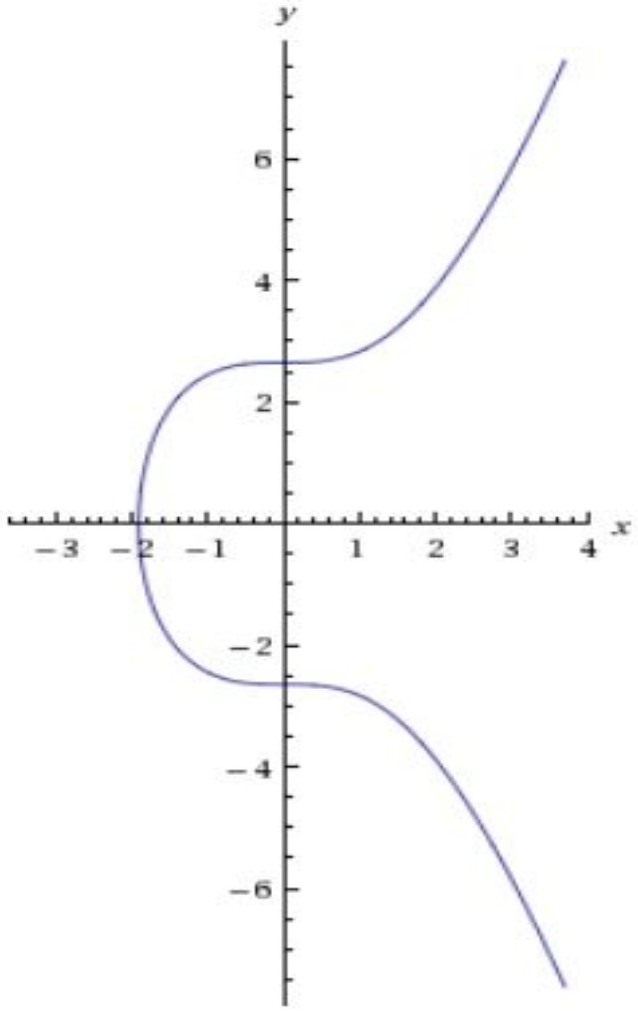
Crittografia ellittica

- La crittografia ellittica (ECC) utilizza le curve ellittiche su un campo $\mathbb{Z}_n$
- Nelle tecnologie blockchain l'algoritmo che viene utilizzato per firmare le transazioni è l'**ECDSA**, Elliptic Curve Digital Signature Algorithm

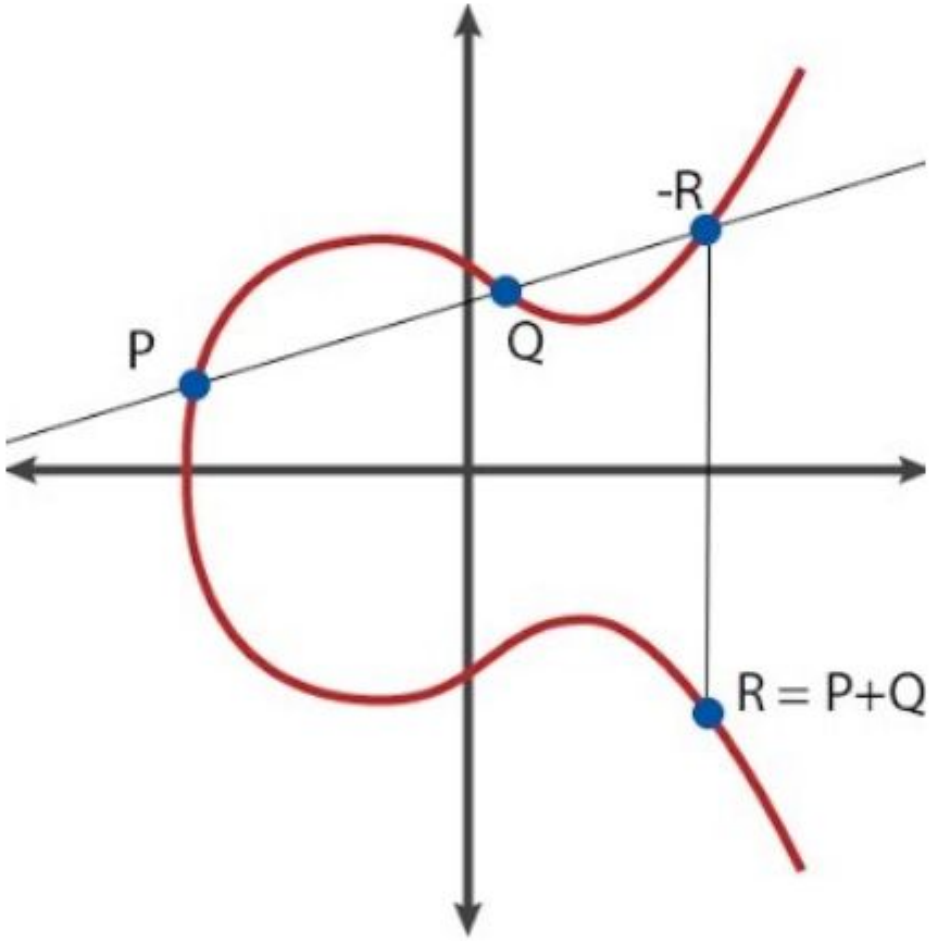


Secp256k1

- Una delle curve più utilizzate nelle tecnologie bLockchain (es. Bitcoin) è la **Secp256k1**, dove a=0 e b=7, ossia la curva $\mathbf{y^2 = x^3 + 7}$
- Bitcoin utilizza $\mathbb{Z}$ definita sul campo $\mathbb{Z}_{2^{256}-2^{32}-977}$, nel quale le coordinate x e y sono interi a 256-bit modulo un primo molto grande



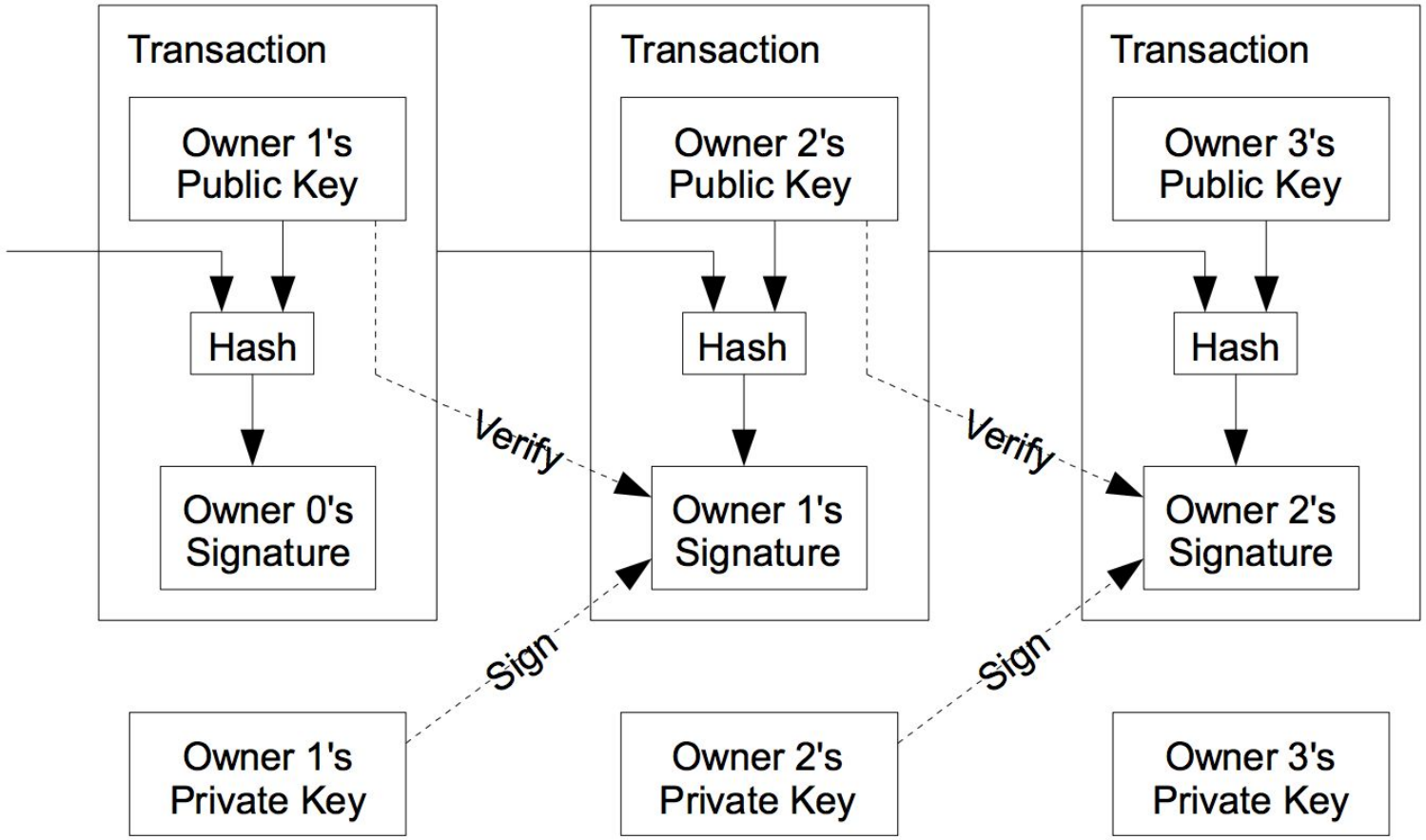
Somma di due punti



Sicurezza dell'ECC

- Avendo definito la somma di due punti è possibile chiedersi il seguente problema: dati due punti **P** e **Q** di una curva ellittica trovare il numero intero **k** tale che **P = Q ∗ k**
- Questo è un problema computazionalmente difficile, su questo si basa la sicurezza dei sistemi ECC

Firma delle transazioni



Generazione di un indirizzo Bitcoin

- Utilizzando una chiave pubblica *Pub* generata con la curva Secp256k1 si può calcolare un indirizzo (*Address*) Bitcoin nel modo seguente:

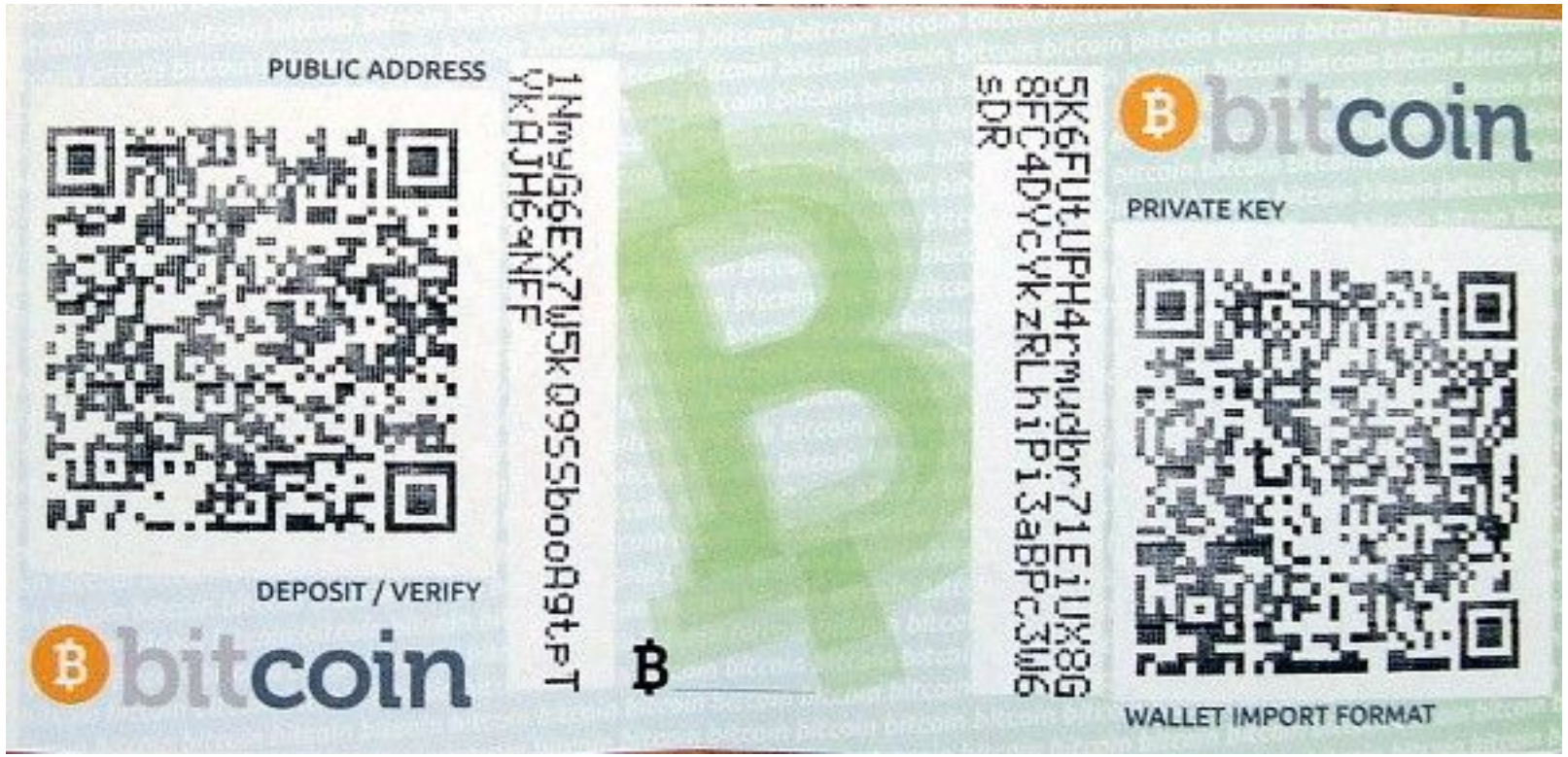
$$hash = 00 \parallel \mathrm{RIPEMD160}(\mathrm{SHA256}(04 \parallel Pub))$$

$$checksum = \mathrm{SHA256}^2(hash)$$

$$Address = hash \parallel checksum[:4]$$

- dove | | è l'operatore di concatenazione e [:4] indica i primi 4 byte.

Esempio di Paper Wallet (in disuso)



Esercizio

- Scrivere un programma in Python che sia in grado di utilizzare la crittografia a curva ellittica e nello specifico l'algoritmo **ECDSA** con la curva **Secp256k1**
- Provare a generare una coppia di chiavi: privata e pubblica
- A partire dalla chiave pubblica generata, provare a costruire un indirizzo Bitcoin **P2PKH** seguendo l'articolo: https://medium.com/coinmonks/bitcoin-address-generation-on-python-e267df5ff3a3

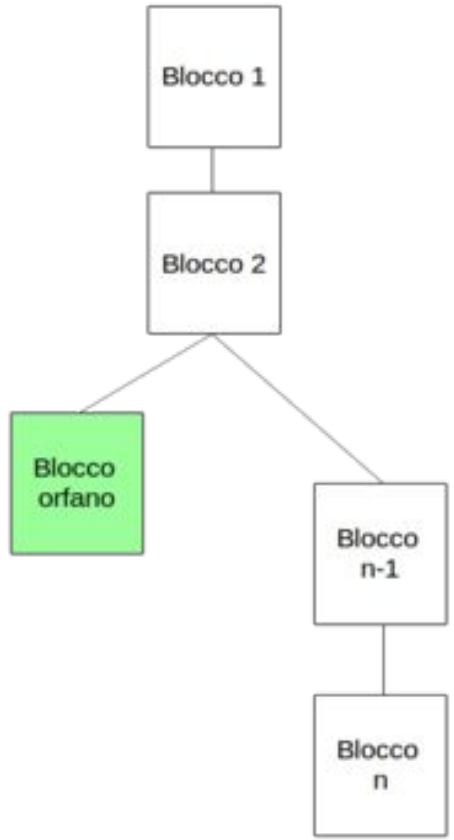
  Potete verificare la validità dell'indirizzo utilizzando questo sito: https://thomas.vanhoutte.be/tools/validate-bitcoin-address.php

Double spending

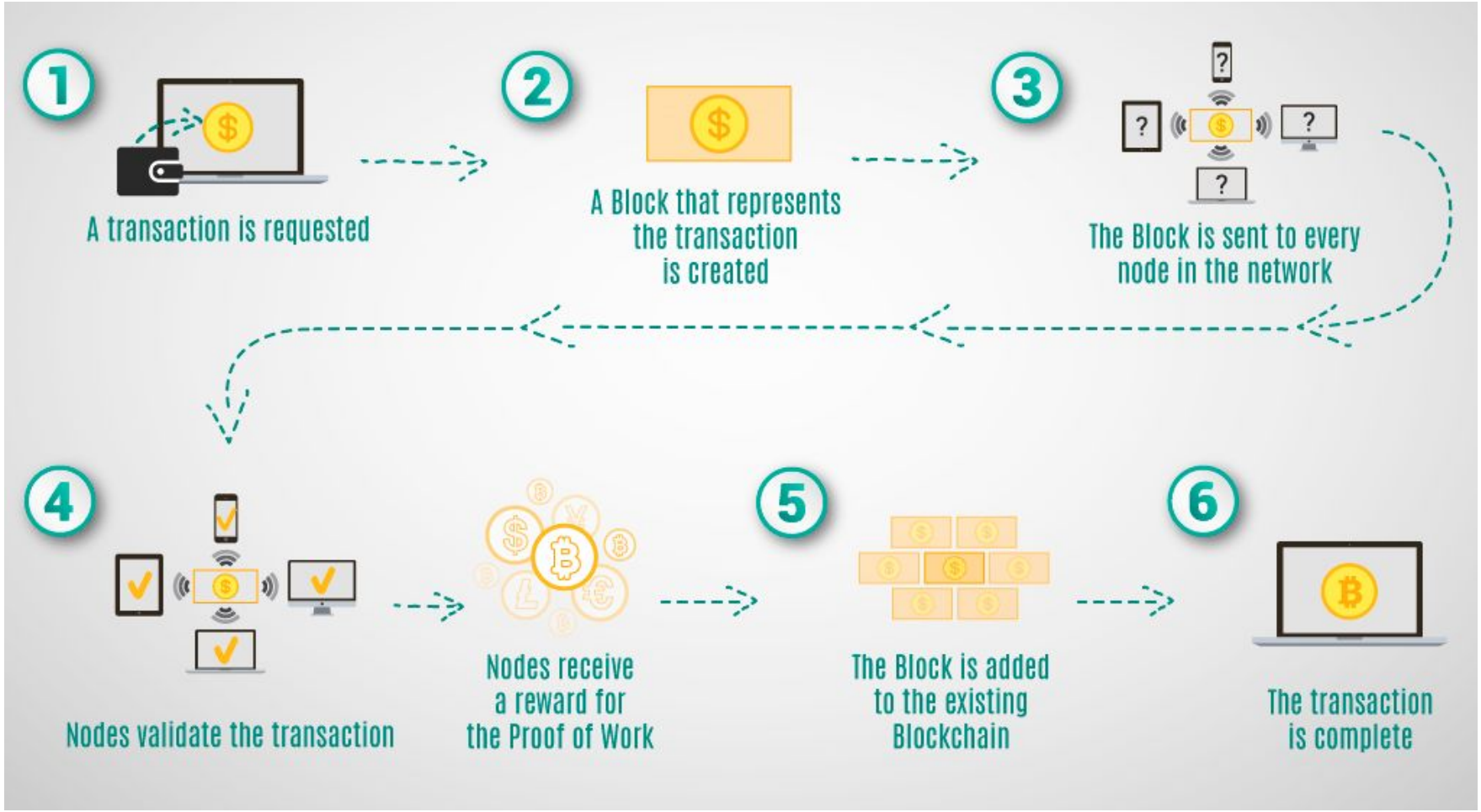
- Utilizzando una moneta elettronica bisogna risolvere il problema del **double spending**
- Dal momento che nel digitale è molto facile copiare una transazione, come faccio ad impedire che A invii a B x bitcoin e contemporaneamente invii gli stessi x bitcoin a C?
- Bitcoin e le altre criptovalute basate su Blockchain utilizzano un **meccanismo di validazione** per prevenire il double spending

Validazione dei blocchi

- Per poter accettare un blocco è necessaria la validazione del PoW
- I blocchi vengono inseriti in sequenza
- La sequenza più lunga è quella che vince
- In caso di pari lunghezza viene effettuata una scelta casuale
- Se la maggioranza dei nodi (oltre il 50%) non è malevola la sicurezza della blockchain è garantita



Esempio di transazione Bitcoin



Bibliografia

- Demers et al. (1987), Epidemic algorithms for replicated database maintenance, PODC '87: Proceedings of the sixth annual ACM Symposium on Principles of distributed computing
- Satoshi Nakamoto, Bitcoin: A Peer-to-Peer Electronic Cash System, Bitcoin whitepaper
- Narayanan et al. (2016), Bitcoin and Cryptocurrency Technologies: A Comprehensive Introduction, Princeton University Press
- Andreas M. Antonopoulos (2017), Mastering Bitcoin: Programming the Open Blockchain: Unlocking Digital Cryptocurrencies, O'Reilly Media, Inc.
- S. Haber, S. Stornetta (1991), How to Time-Stamp a Digital Document, Journal of Cryptology, Vol. 3. Num. 2
- David Chaum, Blind Signatures for Untraceable Payments, Advances in Cryptology Proceedings of Crypto 82
- Jordan Baczuk, How to Generate a Bitcoin Address — Step by Step, Medium blog , 2018
- Beatrice Bertani, La crittografia nel sistema di moneta digitale Bitcoin, Tesi di Laurea in Matematica, Università di Bologna

Grazie!

Per informazioni: enrico.zimuel@its-ictpiemonte.it

Intervento realizzato da ITS TECNOLOGIE DELL'INFORMAZIONE E DELLA COMUNICAZIONE