📖 Introduzione al corso

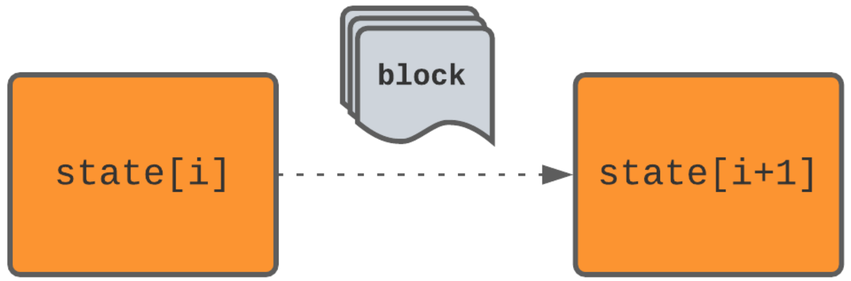
- Blockchain e macchine a stati
- Come funziona una blockchain: crittografia e consenso
- Ethereum: introduzione alla EVM e agli smart contracts
- Ethereum today: Dapps, DeFi, De-*
- Ethereum tomorrow: Eth2.0, L2s, scalability e altre buzzwords
- Q&A

Blockchain e macchine a stati

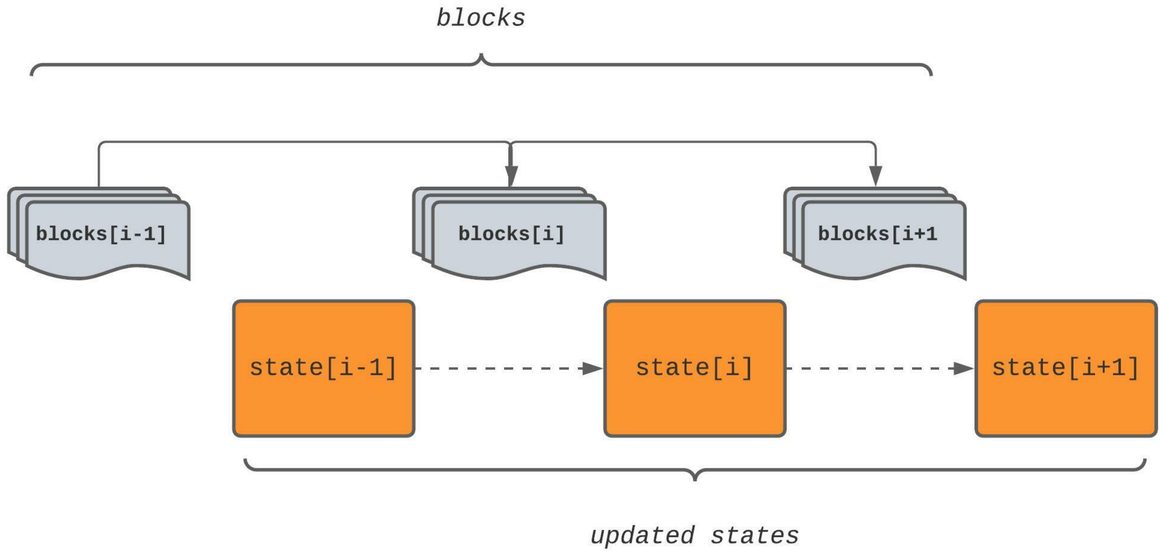
Una blockchain è una **macchina a stati** basata su transazioni **crittograficamente verificate**. Lo stato della macchina (*world state*) è **pubblico** e **distribuito** tra i nodi di una rete peer-to-peer.

Le transazioni vengono pubblicate da attori esterni (account o wallet) che interagiscono tra loro tramite l'utilizzo di primitive crittografiche.

Tali transazioni vengono accumulate in **blocchi di transazioni**.



Blockchain e macchine a stati



Lo stato esposto da una rete blockchain è lo **stato aggregato** applicando le trasformazioni dettate dai blocchi accettati dalla rete stessa.

I nodi della rete conservano e gestiscono tutti i blocchi prodotti.

Consenso

Per causare il passaggio della macchina a stati da uno stato all'altro, un blocco deve essere considerato valido.

Il processo che porta alla validazione di un blocco e alla successiva propagazione dello stesso è noto come **consenso**.

Esistono vari **algoritmi di consenso**, applicabili a vari contesti:
- Proof of Work (basato sull' hashing del blocco)
- Proof of Stake (basato sul *peso* di un nodo all'interno della rete)
- Proof of Authority (basato su KYC o qualunque altro agreement tra i client della rete)

Transazioni e wallet

Le transazioni avvengono tramite l’utilizzo di **crittografia asimmetrica**.

Ogni wallet viene identificato tramite una **chiave pubblica** che può essere utilizzata per inviare transazioni verso di esso.

Ogni transazione viene firmata dal mittente con la **chiave privata** associata alla propria **chiave pubblica** (il che permette inoltre di verificare l’appartenenza di tale chiave pubblica ad un account).

Una transazione firmata dal proprietario di una chiave privata verrà sempre associata alla sua chiave pubblica, anche se non propagata dal proprietario.

Blockchain e cryptovalute

Una cryptovaluta è una valuta virtuale realizzata con l'utilizzo della tecnologia blockchain.

Nick Szabo descrive nel 1998 una valuta virtuale chiamata *bit gold*. I partecipanti della rete di *bit gold* avrebbero dovuto risolvere puzzle crittografici che sarebbero stati assegnati alla loro chiave pubblica in un *Byzantine fault-tolerant public registry*.

Satoshi Nakamoto nel 2008 presenta *Bitcoin*. Con l'avvento di Bitcoin vengono introdotti concetti che saranno fondamentali per le reti blockchain (**proof of work** per la risoluzione del consenso bizantino, SPV, basi per la cryptoeconomics)

Ethereum

Nel 2013 Vitalik Buterin presenta Ethereum.

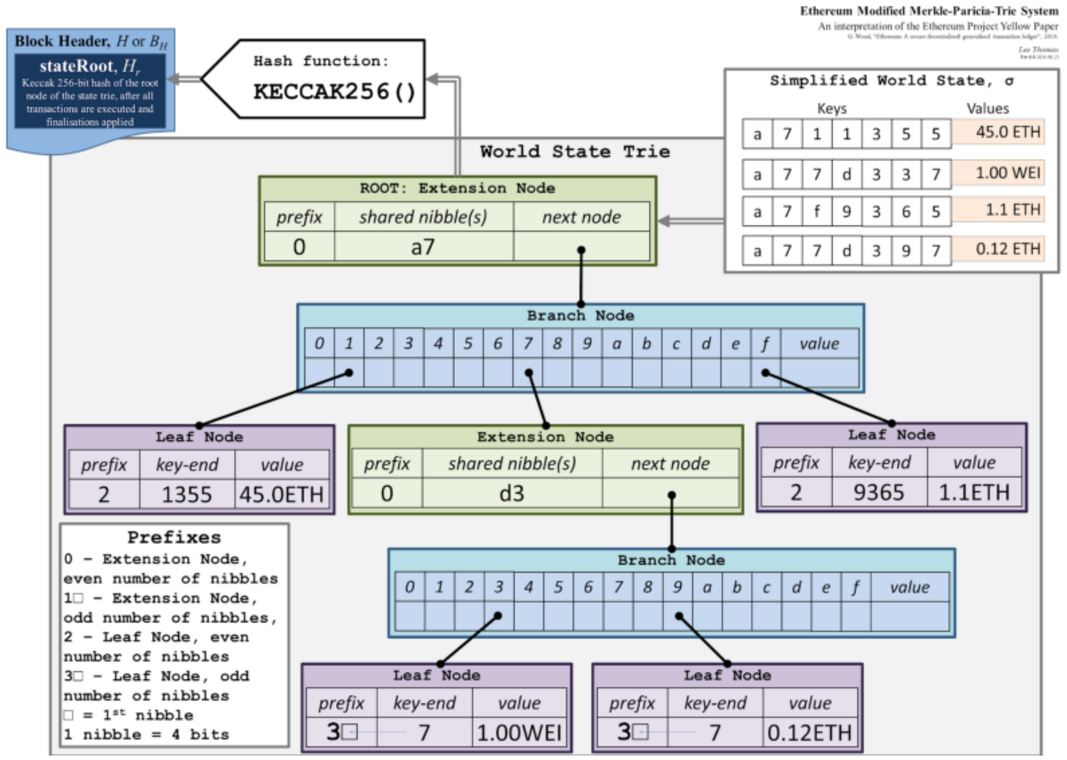
Ethereum è una **state machine deterministica**, costituita da un singolo stato globale che viene mutato eseguendo codice contenuto in delle routine chiamate **smart contract**.

Gli smart contract vengono memorizzati nello stato globale mantenuto da Ethereum e vengono eseguiti dalla EVM (Ethereum Virtual Machine).

La EVM permette l’esecuzione di applicazioni complesse (quasi-Turing completa) in modo decentralizzato e verificabile.

Ethereum: world’s state trie

Ethereum conserva lo stato globale in una struttura dati in grado di memorizzare i dati relativi agli utenti come un **key-value store** (world’s state trie).



Ethereum: EOA e smart contracts

In Ethereum abbiamo due tipi di account:

- externally-owned accounts (EOAs)
- smart contracts accounts

La cosa che differenzia i due tipi di account è la presenza di codice all'interno dello stato globale.

Quando il destinatario di una transazione è un contratto, essa causa l'**esecuzione** del contratto.

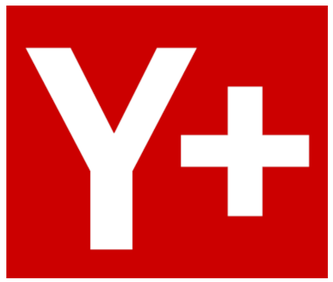
Uno smart contract non può avviare una transazione (può però eseguire altri contratti una volta avviata la sua esecuzione)

EVM: linguaggi di programmazione

La EVM è una virtual machine stack-based che esegue **bytecode.**

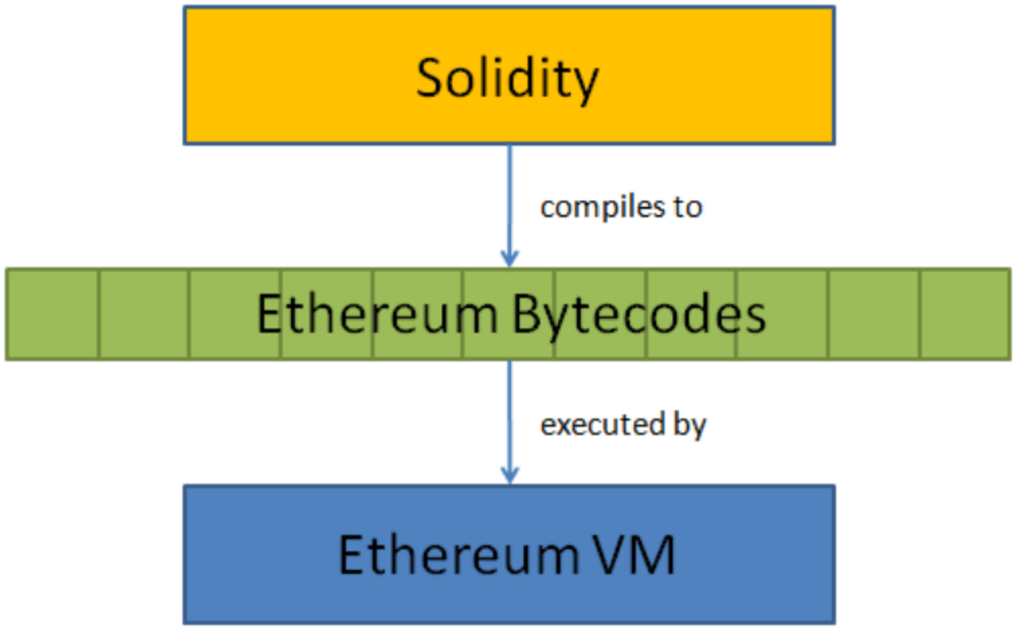
Esistono vari linguaggi di programmazione per la EVM tra cui:

- Solidity, ispirato a Javascript
- Vyper
- Yul+



EVM: linguaggi di programmazione

Tali linguaggi vengono compilati in **bytecode**, che viene poi memorizzato nello smart contract account una volta rilasciato sulla blockchain lo smart contract.



EVM: risolvere l’halting problem con gas e stack depth

La capacità di Ethereum di eseguire un programma arbitrario tramite una virtual machine lo rende una Universal Turing Machine

L’impossibilità di determinare se lo smart contract eseguito terminerà la sua esecuzione o meno è un problema (*halting problem*)

L’esecuzione (potenzialmente infinita) di codice sulla EVM viene limitata applicando dei costi di esecuzione e dei limiti alla profondità a cui le chiamate possono andare (stack depth)

EVM: risolvere l’halting problem con gas e stack depth

Ogni istruzione eseguita dalla EVM ha un costo in **unità di gas.**

Quando una transazione richiede l’esecuzione di uno smart contract, essa deve specificare un **limite di gas** che desidera raggiungere durante l’esecuzione.

Questa quantità rappresenta l’upper bound per l’esecuzione della transazione.

Terminato il gas messo a disposizione della EVM, l’esecuzione viene terminata.

Il **gas** è espresso in Ether (la currency di Ethereum).

Ethereum today: Dapps, DeFi, de-*

Il modello di esecuzione di smart contract in modo trustless permette la creazione di DApps (Decentralized Applications).

Una DApp è spesso costituita da un front-end statico (spesso ospitato da un servizio decentralizzato come IPFS) che interagisce con gli smart contracts.

Una DApp è una applicazione che utilizza lo stato di Ethereum come backend.

Ethereum today: Dapps, DeFi, de-*

L’interazione con le DApp da parte degli utenti avviene tramite l’utilizzo di wallet come Metamask, Frame e altri.

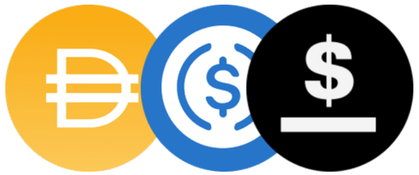
Tali wallet fungono come vero e proprio gateway tra gli utenti e la blockchain.

Ethereum today: Dapps, DeFi, de-*

RWA Oracles

Collaboration platforms

Decentralized DID



Stablecoins

Lending Markets

Decentralized Exchanges

P2P Code Collaboration Tools

Social Media building blocks

Ethereum L2s: Scaling Ethereum

Parte del problema della scalabilità in Ethereum trova soluzione in protocolli che usano la mainnet per registrare una forma compatta di un gran numero di transazioni che vengono effettuate altrove (Layer 2)

Protocolli come zkSync si basano su un concetto chiamato **ZK-rollup** che permette di rappresentare in modo compatto un gran numero di transazioni la cui esecuzione viene crittograficamente verificata da un *prover*

Il protocollo Optimism si basa invece sul concetto di **Optimistic Rollup** che consiste in transazioni che contengono le informazioni minime e un meccanismo grazie al quale entità off-chain possono segnalare eventuali transazioni malevole avvenute sul L2

q&a