Sommario

[Blockchain: concetti di base 3](#_Toc131504964)

[Blocco 3](#_Toc131504965)

[DLT – Distributed Ledger System 4](#_Toc131504966)

[Fork 4](#_Toc131504967)

[Versioni 5](#_Toc131504968)

[Transazione 7](#_Toc131504969)

[Tipi di transazioni 9](#_Toc131504970)

[Criptovalute 10](#_Toc131504971)

[Tipi principali 10](#_Toc131504972)

[Wallet 16](#_Toc131504973)

[Crittografia nelle blockchain 19](#_Toc131504974)

[Mining dei blocchi 24](#_Toc131504975)

[Perché fare mining 25](#_Toc131504976)

[Come si realizza il mining 26](#_Toc131504977)

[Considerazioni sul mining 27](#_Toc131504978)

[Algoritmi di consenso 27](#_Toc131504979)

[Proof of Work 27](#_Toc131504980)

[Proof of Stake 31](#_Toc131504981)

[Altre tipologie di consenso 35](#_Toc131504982)

[Tipi di blockchain 43](#_Toc131504983)

[Sidechain 46](#_Toc131504984)

[Blockchain trilemma 49](#_Toc131504985)

[Scalabilità 51](#_Toc131504986)

[Soluzioni di scalabilità 66](#_Toc131504987)

[Sicurezza 69](#_Toc131504988)

[Blockchain: concetti avanzati 78](#_Toc131504989)

[Token 79](#_Toc131504990)

[Tipologie di token 81](#_Toc131504991)

[Standard per i token 85](#_Toc131504992)

[Fungible Tokens 90](#_Toc131504993)

[NFT (Non Fungible Tokens) 91](#_Toc131504994)

[Tokenizzazione 95](#_Toc131504995)

[Cosa tokenizzare, tipi e benefici 96](#_Toc131504996)

[Ambiti applicativi 98](#_Toc131504997)

[Come creare un token 100](#_Toc131504998)

[Smart contract 101](#_Toc131504999)

[Come funzionano 105](#_Toc131505000)

[Dove vengono usati 105](#_Toc131505001)

[Tipologie 108](#_Toc131505002)

[Pro e contro 109](#_Toc131505003)

[Considerazioni legali e casi d’uso 110](#_Toc131505004)

[Possibili problemi 111](#_Toc131505005)

[Linguaggi per smart contract: Solidity 117](#_Toc131505006)

[Esempi di codice e passi base 117](#_Toc131505007)

[Solidity in a nutshell 126](#_Toc131505008)

[Framework JavaScript per interazione blockchain 128](#_Toc131505009)

[Ethers.js 128](#_Toc131505010)

[web3.js 137](#_Toc131505011)

[Vulnerabilità di Solidity e degli smart contract 153](#_Toc131505012)

[Riferimenti usati 166](#_Toc131505013)

# Blockchain: concetti di base

La blockchain è un protocollo software che definisce come trasferire soldi ed asset. Esso è un registro digitale che registra e archivia le informazioni in modo sicuro e trasparente. Si immagini un quaderno digitale a cui possono accedere e aggiornare molte persone allo stesso tempo, ma una volta scritto qualcosa non può essere cancellato o modificato. Il sistema è definito *peer-to-peer (p2p),* in quanto trasferisce valore tra due utenti senza un intermediario. Definiamo alcune caratteristiche di base:

* Essa è *decentralizzata*, in quanto non ci sono terze parti coinvolte: banche, cloud, amministratori. L'innovazione di una blockchain è che garantisce la fedeltà e la sicurezza di un record di dati e genera fiducia senza la necessità di una terza parte fidata. Essendo un registro pubblico, garantisce la trasparenza di tutte le transazioni e ne consente la tracciabilità. Se da un lato ciò può consentire una controprova delle informazioni, dall'altro rende difficile la loro falsificazione.
* Le informazioni vengono registrate in modo anonimo e immutabile. Ciò significa che una volta aggiunti al registro, i dati non possono essere rimossi o alterati. Né il fornitore può essere identificato nominativamente.

## Blocco

I dati sono strutturati a blocchi, ciascuno contenente un set di informazioni. Essi hanno alcune capacità di memoria e, quando riempiti, sono chiuso e collegati al blocco precedente. Tutte le nuove informazioni che seguono il blocco appena aggiunto vengono compilate in un blocco di nuova formazione che verrà aggiunto alla catena una volta riempito.

* Un database di solito struttura i suoi dati in tabelle, mentre una blockchain, come dice il nome stesso, struttura i suoi dati in pezzi (blocchi) che sono collegati tra loro. Questa struttura di dati rende intrinsecamente irreversibile la cronologia dei dati quando è implementata in modo decentralizzato. A ogni blocco della catena viene assegnato un *timestamp* esatto quando viene aggiunto alla catena.
* Per cominciare, i nuovi blocchi sono sempre memorizzati in modo lineare e cronologico. Cioè, vengono sempre aggiunti alla "fine" della blockchain. Dopo che un blocco è stato aggiunto alla fine della blockchain, è estremamente difficile tornare indietro e modificarne il contenuto, a meno che la maggioranza della rete non abbia raggiunto un consenso in tal senso.
* Questo perché ogni blocco contiene il proprio hash, insieme all'hash del blocco precedente e al già citato timestamp. I codici hash sono creati da una funzione matematica che trasforma le informazioni digitali in una stringa di numeri e lettere. Se queste informazioni vengono modificate in qualche modo, anche il codice hash cambia. Inoltre, ogni blocco contiene:
  + Dei dati (data);
  + Un nonce (number used only once), numero generato casualmente quando il blocco è creato
  + Quando viene creato il primo blocco di una catena, un nonce genera l'hash crittografico. I dati contenuti nel blocco sono considerati firmati e legati per sempre al nonce e all'hash, a meno che non vengano estratti.

## DLT – Distributed Ledger System

Come detto, l’informazione non deve essere alterata; per questo sono anche noti come *distributed ledger technology (DLT)* (in italiano *libro mastro distribuito*), tecnologia alla base delle blockchain.

* In poche parole, un libro mastro distribuito è un database che non ha bisogno di una terza parte per garantire che le transazioni registrate siano valide e oneste. Molti utenti hanno diversi livelli di accesso al database. Le modifiche ai dati vengono registrate automaticamente e non possono essere modificate da un utente senza autorizzazione. In alcuni libri mastri distribuiti, nessuno può apportare modifiche; in altri, le modifiche possono essere apportate, ma sono tracciate e l'utente che le ha apportate è noto.
* La Distributed Ledger Technology (DLT) consente di archiviare le informazioni in modo sicuro e preciso utilizzando la crittografia. L'accesso ai dati può avvenire tramite "chiavi" e firme crittografiche. Una volta memorizzate, le informazioni possono diventare un database immutabile; le regole della rete, scritte nella codifica della programmazione del database, governano il libro mastro.
* Essendo decentralizzati, privati e crittografati, i libri mastri distribuiti sono meno soggetti alla criminalità informatica, in quanto tutte le copie memorizzate nella rete devono essere attaccate simultaneamente perché l'attacco abbia successo. Inoltre, la condivisione e l'aggiornamento peer-to-peer dei record rendono l'intero processo molto più veloce, efficace ed economico.
* Ogni dispositivo di una rete di ledger distribuiti memorizza una copia del libro mastro. Questi dispositivi sono chiamati *nodi*: la rete può avere un numero qualsiasi di nodi. Qualsiasi modifica al libro mastro, come lo spostamento di dati da un blocco all'altro, viene registrata su tutti i nodi. Poiché ogni nodo possiede una copia del libro mastro, ognuno di essi pubblica la propria versione con le ultime transazioni.
* Si noti comunque che di base la decentralizzazione ne è il fattore comune, ma blockchain e DLT differiscono:
  + Le DLT hanno dati concatenati (dunque, non usano blocchi), possono essere criptate, private e con permessi appositi; possono essere immutabili
  + Le blockchain sono memorizzate in blocchi e sono sempre criptate. Sono generalmente pubbliche e senza permessi (permissionless), ma alcune hanno permessi appositi. Sono inoltre sempre immutabili.

## Fork

Esistono le cosiddette fork, cioè le modifiche al protocollo/regole di base delle blockchain. La rete è un software open source e il codice è liberamente disponibile. Ciò significa che chiunque può proporre miglioramenti e modificare il codice. La possibilità di sperimentare sul software open source è una parte fondamentale delle criptovalute e facilita anche gli aggiornamenti del software della blockchain.

* + In ogni momento, migliaia di minatori sono in competizione per creare un nuovo blocco. Con un'attività di mining così intensa, a volte due o più minatori creano un nuovo blocco nello stesso momento. Quando ciò accade, si crea un *fork accidentale*. Il problema si risolve quando vengono aggiunti nuovi blocchi a una delle catene. In questo caso, la rete continua a lavorare sulla catena più lunga e abbandona quella più corta.
  + Quando si crea una biforcazione intenzionale, la rete non si riconverte su una singola catena. Questo tipo di fork viene utilizzato dagli sviluppatori di blockchain per implementare modifiche al protocollo. Ad esempio, gli sviluppatori possono utilizzare una fork intenzionale per aumentare le dimensioni dei blocchi, ridurre il tempo di blocco o persino implementare un algoritmo di consenso completamente nuovo.
    - In questi fork distinguiamo gli *hard fork*, in cui i membri della comunità (utenti e minatori) devono prendere una decisione. Possono aggiornare il loro nodo e passare alla nuova catena forzata, oppure possono continuare a utilizzare il vecchio software. In ogni caso, possiedono criptovalute su entrambe le catene: hanno monete sulla catena precedente e possono rivendicare le criptovalute del nuovo protocollo sulla nuova catena.
      * Normalmente sono forzati (hard) ad aggiornare, in quanto i nodi che non si adeguano alle nuove regole di consenso non possono più partecipare al meccanismo di consenso presente.
    - A differenza degli hard fork, le modifiche implementate dai *soft fork* rendono le catene compatibili con il futuro. Per essere compatibili con il futuro, i blocchi creati secondo le nuove regole devono essere validi anche secondo le vecchie regole (ma non viceversa). Per questo motivo, un soft fork non richiede l'aggiornamento dei nodi. Possono continuare a eseguire la vecchia versione del software e partecipare alla rete aggiornata come validatori di transazioni.
      * Le soft fork possono essere attivate dagli utenti (UASF, ovvero user activated soft fork) o dai miner (MASF, ovvero miner activated soft fork).
  + Alcune blockchain vengono create per testare funzionalità e non creare overhead sulla rete principale, permettendo di testare nuove caratteristiche prima che ne venga fatto il deploy sulla rete principale, chiamate *testnet*.

## Versioni

La tecnologia blockchain ha avuto tre iterazioni fondamentali nel corso degli anni. Per essere chiari, considereremo la prima iterazione come la versione concettualizzata e implementata dal gruppo Nakamoto. Ogni versione successiva della blockchain include ulteriori caratteristiche di codifica e strutturali che consentono agli utenti di eseguire una gamma più ampia di operazioni.

* Versione 1.0: Origine della blockchain moderna: Currency (Valuta)
  + Blockchain 1.0 è stata la prima applicazione della tecnologia implementata da Satoshi Nakamoto, creatore di Bitcoin. Questa versione è la forma più semplice di un libro mastro decentralizzato per la registrazione delle transazioni e l'archiviazione dei dati su diversi computer. In parole povere, le informazioni registrate nelle prime forme di blockchain erano limitate ai valori di un "oggetto" che cambiava proprietà nel tempo. Nella maggior parte dei casi, il "bene" a cui ci riferiamo era una valuta digitale.
  + Questa versione di Bitcoin è essenzialmente un sistema di trasferimento di denaro elettronico che è automatizzato e funziona senza la necessità di un intervento umano tra le transazioni come autorità di fiducia. Questo sistema consente agli utenti di scambiare valute senza l'intervento di una banca (del settore privato o governativo). Il sistema ha permesso agli utenti finali di trasferire fondi in modo anonimo utilizzando portafogli e dando accesso al registro completo delle transazioni a tutti coloro che sono collegati alla blockchain. Era effettivamente trasparente.
  + La tecnologia iniziale consentiva anche ai minatori (utenti che aiutavano a verificare le transazioni eseguendo compiti crittografici attraverso i loro computer) di ottenere ricompense attraverso i meccanismi di Proof of Work integrati nei protocolli.
* Versione 2.0: Smart Contract
  + Il successivo sviluppo della tecnologia blockchain ha ampliato le capacità dei protocolli blockchain. Quattro anni dopo l'ascesa del Bitcoin, Vitalik Buterin ha introdotto il concetto di Ethereum, una tecnologia basata sulla blockchain con alcuni notevoli miglioramenti rispetto alla generazione precedente.
  + Ethereum è stata la prima blockchain con lo smart contract integrato nel suo protocollo. In parole povere, gli smart contract sono un insieme di codici che vengono automatizzati quando si verificano determinate condizioni. Questi contratti consentono a due utenti o organizzazioni di effettuare più di semplici transazioni di criptovaluta. In questo modo, gli accordi tra due parti eseguono automaticamente task complessi e si ha l’utilizzo di applicazioni decentralizzate che hanno permesso la crescita di organizzazioni autonome (DAO)
* Versione 3.0: DApps (applicazioni decentralizzate)
  + Blockchain 3.0 è il livello successivo della tecnologia blockchain e ha una portata più ampia rispetto alle versioni precedenti. Ha applicazioni in settori diversi dalla finanza e dall'economia e si occupa di sostenibilità, scalabilità, economicità, decentralizzazione e sicurezza. Non si ha ancora una chiara definizione di quest’ultima, ma si intende il contenere un ampio insieme di industrie e applicazioni relative ed incentrare a queste tematiche.
  + Una DApp è simile a un'applicazione tradizionale in quanto può avere un frontend scritto in qualsiasi linguaggio che chiama il backend, e il codice del backend viene eseguito su una rete decentralizzata peer-to-peer. A differenza dei programmi centralizzati che vengono eseguiti su server centralizzati, queste applicazioni utilizzano l'archiviazione e la comunicazione decentralizzata e la maggior parte del loro codice backend viene eseguito su una rete decentralizzata.
  + Essa consente anche l'interoperabilità di diverse reti blockchain in tutto il mondo. Richiede soluzioni innovative per integrare diverse tecnologie, compresi i dispositivi IoT e i circuiti integrati specifici per le applicazioni (ASICS). Blockchain 3.0 utilizza algoritmi DAG e rende popolare il modello proof-of-stake (PoS), che elimina l'uso di dispositivi informatici altamente complessi e riduce il consumo energetico.
* Versione 4.0: Rendere la blockchain utilizzabile nell'industria
  + Con le basi gettate dalle versioni precedenti, per noi Blockchain 4.0 descrive soluzioni e approcci che rendono la tecnologia blockchain utilizzabile per le esigenze aziendali. In particolare alle esigenze dell'Industria 4.0.
  + Industria 4.0 significa in breve automazione, pianificazione delle risorse aziendali e integrazione di diversi sistemi di esecuzione. Tuttavia, questa rivoluzione industriale richiede un grado crescente di fiducia e protezione della privacy: è qui che entra in gioco la blockchain.
  + Quando si aggiunge la blockchain ai sistemi IT si ottiene un'integrazione aziendale che consente processi aziendali cross-system/cross-blockchain, ad esempio macchine che ordinano in modo sicuro e autonomo l'arrivo di parti di ricambio. La gestione della catena di approvvigionamento, i flussi di lavoro di approvazione, le transazioni finanziarie e i pagamenti basati sulle condizioni, la raccolta di dati IoT, la gestione della salute e degli asset sono solo alcuni esempi di aree che possono essere potenziate dalla tecnologia blockchain.
  + Blockchain 4.0 significa rendere Blockchain 3.0 utilizzabile in scenari aziendali reali. Soddisfare le richieste dell'Industria 4.0 facendo sì che le promesse della blockchain diventino realtà.

## Transazione



Riferimento: <https://www.investopedia.com/terms/b/blockchain.asp>

Una transazione in una rete blockchain comporta il trasferimento di beni digitali, come criptovalute o token, da un indirizzo o conto a un altro.

* In primo luogo, ci sono gli utenti, cioè persone che vogliono utilizzare la meccanica della blockchain per effettuare una transazione. Domanda lecita: in un sistema che non ha una struttura centrale, chi lo fa accadere?
* È qui che entrano in gioco i minatori. La blockchain è un sistema privo di fiducia, che consente agli individui di sapere che la loro transazione sarà onorata correttamente senza che vi sia un organismo centrale a supervisionarla. I minatori lo rendono possibile verificando i blocchi di transazione in arrivo, ovvero i lotti di transazioni richieste che attendono (nella mempool, la fase che intercorre tra la richiesta e l'aggiunta alla blockchain) di essere confermate. Se ciò avviene correttamente, si ottiene una ricompensa per i minatori, che è l'incentivo che mantiene in funzione il sistema.
* Paragone del loro utilizzo nel mondo reale:
  + Perché paghiamo le tariffe del gas? Ecco la risposta: compensano l'energia utilizzata nel processo di estrazione delle transazioni e contribuiscono alla ricompensa del blocco per i minatori.
* Infine, ci sono i nodi. Chiunque può essere un nodo. I nodi mantengono l'intero sistema sicuro e protetto convalidando i blocchi di transazioni inviati dai minatori prima che vengano aggiunti alla blockchain. Lo fanno verificando le informazioni in arrivo con la cronologia delle transazioni della blockchain, per assicurarsi che tutto corrisponda. I nodi della rete - sparsi in tutto il pianeta - raggiungono quindi collettivamente un consenso sulla validità delle nuove transazioni, prima di aggiungerle alla blockchain.

Ecco una spiegazione semplificata di come avviene una transazione in una blockchain:

1. L'utente avvia la transazione creando una firma digitale utilizzando la propria chiave privata. La firma dimostra che l'utente ha il diritto di inviare i beni.
2. La transazione viene trasmessa alla rete di nodi o computer che eseguono il software della blockchain. Ogni nodo riceve la transazione e la aggiunge a un pool di transazioni non confermate.
3. I nodi della rete convalidano la transazione per assicurarsi che il mittente abbia fondi sufficienti per completare la transazione e che questa sia conforme alle regole del protocollo blockchain.
4. Una volta che un numero sufficiente di nodi ha convalidato la transazione, questa viene aggiunta a un nuovo blocco di transazioni, insieme ad altre transazioni convalidate di recente.
5. Il blocco di transazioni viene aggiunto alla blockchain in un processo chiamato mining. L'estrazione comporta la risoluzione di complesse equazioni matematiche per creare un nuovo blocco, il che richiede una grande potenza di calcolo.
6. Una volta aggiunto il nuovo blocco alla blockchain, la transazione viene considerata confermata e i beni vengono trasferiti dall'indirizzo del mittente a quello del destinatario. La transazione è ora registrata in modo permanente sul libro mastro della blockchain, che può essere visualizzato e verificato da chiunque abbia accesso alla rete.
7. Infine, l'utente può controllare lo stato della sua transazione utilizzando il *blockchain explorer*, uno strumento che consente di visualizzare i dettagli della transazione e il suo avanzamento nella rete blockchain. Essendo decentralizzate, questo permette a tutti di vedere le transazioni che avvengono in diretta.
   1. Questo strumento presenta la blockchain in una maniera user-friendly, spesso tramite un’interfaccia web che permetta agli utenti di cercare e filtrare le informazioni secondo vari criteri. In particolare, si possono cercare
      1. Le singole transazioni (inserendo la hash, indirizzo, numero di blocco)
      2. I singoli indirizzi
      3. I dettagli sui blocchi
      4. Le statistiche sulla rete
      5. Grafici sulle performance

In generale, il processo di una transazione in una blockchain prevede la creazione di una firma digitale, la trasmissione della transazione alla rete, la convalida della transazione, l'aggiunta a un nuovo blocco, il mining del blocco, la conferma della transazione e la registrazione permanente sulla blockchain.

### Tipi di transazioni

Vi sono due tipi di transazioni: la transazioni P2PKH (Pay-to-PubKey-Hash) e le transazioni Coinbase.

* Le transazioni P2PKH (Pay-to-PubKey-Hash) sono il tipo più comune di transazioni utilizzate nella blockchain di Bitcoin e in molte altre criptovalute. In queste transazioni, il mittente invia una certa quantità di criptovaluta a un indirizzo Bitcoin. Questo indirizzo è una stringa alfanumerica che rappresenta la chiave pubblica del destinatario, che a sua volta è stata crittografata tramite la funzione hash SHA-256.
  + Per creare una transazione P2PKH, la persona che riceverà i fondi condivide il proprio portafoglio Bitcoin con il mittente. Si noti che un indirizzo è diverso da un hash a chiave pubblica. Un indirizzo codifica un hash a chiave pubblica in un formato facile da condividere e ha un rilevamento degli errori incorporato, oltre a un indicatore all'inizio che denota a quale rete appartiene. Il rilevamento degli errori avviene tramite una somma di controllo (talvolta chiamata somma di hash) che aiuta il software del portafoglio a rilevare un errore di battitura e ad evitare che i fondi vadano persi per sempre.
  + La conversione di una chiave pubblica in un indirizzo richiede che la chiave pubblica sia hashata due volte con due funzioni di hash diverse (SHA-256 per la prima e RIPEMD-160 per la seconda). Questo permette la creazione della transazione P2PKH
  + La conversione dell'hash della pubkey in un indirizzo richiede altri passaggi. Innanzitutto, i byte della versione (normalmente 0x00) vengono aggiunti all'inizio dell'hash della pubkey. Questa nuova stringa viene poi sottoposta a un nuovo hash con SHA-256. I primi quattro byte (32 bit) dell'hash risultante vengono aggiunti alla fine dell'hash RIPEMD-160 precedente. Questo funge da checksum. Quindi l'intera stringa viene convertita da una stringa di byte in una stringa con codifica base58. Il risultato è un indirizzo.
* Le transazioni Coinbase (dette anche transazioni di generazione) sono invece un tipo speciale di transazione che viene creata ogni volta che un nuovo blocco viene aggiunto alla blockchain. Questa transazione contiene una ricompensa in Bitcoin che viene assegnata al minatore che ha creato il blocco. In altre parole, è la transazione che crea nuovi Bitcoin e che permette di incentivare la partecipazione dei minatori alla rete. La transazione Coinbase non ha un mittente effettivo, in quanto viene creata direttamente dal protocollo della criptovaluta e non richiede una firma digitale.
  + Per evitare che la ricompensa resti sempre la stessa, il sistema è programmato per generare questo tipo di transazioni secondo una serie geometrica

## Criptovalute

Le criptovalute sono beni digitali creati e scambiati con tecniche crittografiche, costruite sul sistema blockchain. Formalmente, una criptovaluta è un sistema che soddisfa sei condizioni:

* Il sistema non richiede un'autorità centrale; il suo stato è mantenuto attraverso un consenso distribuito.
* Il sistema mantiene una panoramica delle unità di criptovaluta e della loro proprietà.
* Il sistema definisce se è possibile creare nuove unità di criptovaluta. Se è possibile creare nuove unità di criptovaluta, il sistema definisce le circostanze della loro origine e come determinare la proprietà di queste nuove unità.
* La proprietà delle unità di criptovaluta può essere dimostrata esclusivamente per via crittografica.
* Il sistema consente di effettuare transazioni in cui la proprietà delle unità crittografiche viene modificata. Una dichiarazione di transazione può essere emessa solo da un'entità che dimostri la proprietà attuale di tali unità.
* Se vengono inserite contemporaneamente due diverse istruzioni per la modifica della proprietà delle stesse unità crittografiche, il sistema ne esegue al massimo una.

Esistono molte criptovalute diverse; la tecnologia blockchain è open source, il che significa che qualsiasi sviluppatore di software può utilizzare il codice sorgente originale e creare qualcosa di nuovo con esso. Gli sviluppatori hanno fatto proprio questo. Si stima che al momento della stesura di questo file siano in circolazione più di 10.000 criptovalute diverse, e la cifra continua a crescere.

* Parte della ragione di questa impennata è la relativa facilità con cui è possibile creare nuove criptovalute. Il codice sorgente di una può essere utilizzato per costruirne un'altra. Ad esempio, la rete Ethereum può essere utilizzata per creare monete digitali personali. A volte si verificano dei "fork" nel codice del software che cambiano le regole di funzionamento di una criptovaluta, il che può portare alla creazione di una nuova criptovaluta. Bitcoin Cash (BCH -1,43%) è stato creato nel 2017 come risultato di un fork di Bitcoin che ha permesso di registrare più transazioni su un singolo blocco della blockchain.
* L'impennata dei prezzi delle criptovalute ha spinto molti sviluppatori a cercare di ottenere una fetta dell'azione. Inoltre, la tecnologia blockchain ha un'utilità che va oltre le valute digitali. Quindi, anche se alcune criptovalute potrebbero essere una bolla che alla fine scoppierà, la natura decentralizzata della tecnologia e l'ampio campo di applicazione nel mondo del software sono due ragioni per cui esistono così tante criptovalute.

### Tipi principali

Occorre fare subito alcune distinzioni tra i singoli tipi di valute. Esistono infatti:

* Le altcoin, tutte le criptovalute che non sono Bitcoin. Di queste distinguiamo:
* Le stablecoins, quindi criptovalute che seguono il prezzo del loro asset; non mantengono lo stesso valore e vengono usate per risparmi o per inviare denaro
* Le mining-based, basandosi sul mining per verificare le transazioni
* Le staking-based, per verificare transazioni ed aggiungere più monete a quelle disponibili

#### Bitcoin (BTC)

Il Bitcoin, la prima e più nota criptovaluta, è stato creato nel 2009 da una persona o un gruppo anonimo noto come Satoshi Nakamoto. Utilizza una blockchain per registrare tutte le transazioni e i nuovi bitcoin vengono creati attraverso un processo chiamato mining. Il mining consiste nel risolvere complessi problemi matematici per verificare le transazioni e aggiungerle alla blockchain.

* + Una delle caratteristiche interessanti di Bitcoin è la sua limitata fornitura di unità di valuta. Esistono solo 21 milioni di bitcoin, e finora ne sono stati creati circa 18,5 milioni. La quantità di bitcoin che viene creata in ogni blocco della blockchain viene ridotta a metà circa ogni 4 anni in un processo noto come "halving". Questo significa che la produzione di bitcoin si ridurrà gradualmente fino a quando non si raggiungerà la quantità massima di 21 milioni.
  + Nonostante le migliaia di concorrenti che sono sorti, il Bitcoin, la criptovaluta originale, rimane il giocatore dominante in termini di utilizzo e valore economico. Nessuna ha finora eguagliato il suo valore e la sua capacità di mercato.
  + Per contro, l'infrastruttura blockchain di Bitcoin inizia a rallentare una volta superata la velocità di sette transazioni al secondo.
  + Alcune precisazioni per quanto riguarda le transazioni bitcoin:
    - Quando vengono spesi soldi da una delle parti, esiste uno script di firma contenente la chiave pubblica di Alice e la firma digitale delle transazioni ricevute generate dalla chiave privata, cioè *ScriptSig*.
      * Esso è uno script di input che viene incluso in una transazione Bitcoin per sbloccare i fondi precedentemente bloccati in un'uscita di transazione precedente (un input di transazione). Lo ScriptSig contiene i dati di input necessari per sbloccare l'output della transazione precedente, come ad esempio una firma digitale generata con la chiave privata del mittente.
    - *ScriptPubKey*, invece, è uno script di output che viene incluso nella transazione che crea un nuovo output bloccando dei fondi. Lo ScriptPubKey contiene le istruzioni per bloccare i fondi, specificando le condizioni che il destinatario deve soddisfare per sbloccare quei fondi in futuro. Ad esempio, lo script potrebbe richiedere che il destinatario fornisca una firma digitale valida generata con la sua chiave pubblica, che deve corrispondere alla chiave pubblica inclusa nello script.
    - Quando una transazione viene elaborata, il contenuto dello ScriptSig e dello ScriptPubKey viene combinato per formare uno script completo. Se lo script completo restituisce un valore booleano vero, la transazione viene considerata valida e i fondi vengono sbloccati e trasmessi al destinatario. Se il valore booleano è falso, la transazione viene considerata non valida e i fondi rimangono bloccati nella transazione precedente.

#### Ethereum (ETH)

Ethereum è una piattaforma decentralizzata basata su blockchain, che permette agli utenti di creare e gestire contratti intelligenti e applicazioni decentralizzate (DApps). La sua criptovaluta nativa è l'Ethereum (ETH), che viene utilizzata per pagare le transazioni sulla rete Ethereum.

* + Rispetto ad altre valute, Ethereum si distingue principalmente per la sua funzionalità di contratti intelligenti, che consentono di eseguire automaticamente accordi e transazioni senza l'intervento di intermediari. Questo significa che le applicazioni costruite sulla piattaforma Ethereum possono essere programmate per eseguire operazioni complesse, come ad esempio la gestione di contratti finanziari, senza la necessità di un intermediario di fiducia.
  + Un'altra differenza significativa è la sua capacità di supportare token personalizzati e DApps, che possono essere sviluppati sulla piattaforma Ethereum. Ciò significa che gli sviluppatori possono creare nuove applicazioni decentralizzate che utilizzano la tecnologia di Ethereum come base, piuttosto che dover creare una propria blockchain da zero.
  + Inoltre, Ethereum è in grado di elaborare transazioni più velocemente rispetto a Bitcoin, poiché utilizza un algoritmo di consenso diverso chiamato Proof of Stake (PoS). Questo algoritmo richiede meno energia rispetto al Proof of Work (PoW) utilizzato da Bitcoin e altre valute, il che significa che la rete Ethereum può elaborare più transazioni in meno tempo.
  + Essa è una blockchain programmabile general-purpose e Turing completa. Il codice si esegue su smart contracts memorizzate nelle blockchain, in cui lo stato consiste nel balance (saldo) degli account e anche dello storage. La memoria permanente registra anche gli effetti delle transazioni eseguite, così come gli eventi emessi.
  + In essa esistono due tipi di profili: *EOA (Externally Owned Account)*, per cui ad ognuno è associata una chiave privata e generano transazioni esterne e i *contract account*, che non possono generare transazioni esterne e la loro logica è controllata dallo smart contract associato. Ciascuna transazione esterna è firmata digitalmente e ognuna è l’unico elemento in grado di innescare una transizione di stato o causare l’esecuzione di uno smart contract.

#### Tether (USDT)

Tether è una stablecoin, ovvero una valuta legata a una valuta fiat (*fiat currency,* anche detta fiat money, è una moneta a corso legale il cui valore è sostenuto dal governo che l'ha emessa. Si differenzia dalla moneta che è sostenuta da un bene fisico che ne stabilisce il valore, come l'oro, non dipendendo in questo modo da eventi economici), in questo caso il dollaro statunitense. L'idea alla base di Tether è quella di combinare i vantaggi di una criptovaluta (come l'assenza di intermediari finanziari) con la stabilità di una moneta emessa da un governo sovrano (rispetto alle fluttuazioni di prezzo selvagge tipiche di molte criptovalute).

* È una valuta trasparente, in quanto aggiorna quotidianamente sul proprio sito web una ripartizione delle riserve detenute.
* È una valuta stabile; questo promuove l'uso delle stablecoin come mezzo di scambio al pari della moneta convenzionale. In termini pratici, le stablecoin hanno reso più facile la speculazione sui mercati delle criptovalute. La loro rapida crescita di popolarità è anche il risultato dell'uso delle stablecoins come garanzia nei protocolli di prestito e di staking della finanza decentralizzata (DeFi).
* Essendo una stablecoin, ogni token Tether è ancorato a un'unità di valuta fiat o di metallo prezioso. Quindi, ad esempio, una singola unità di USD₮ dovrebbe mantenere un prezzo di mercato di un dollaro USA o molto vicino ad esso. Ciò avviene garantendo che ogni stablecoin sia sostenuta al 100% da attività di riserva detenute da Tether.
* Poiché le attività sono principalmente contanti o equivalenti a contanti emessi da emittenti finanziari tradizionali o garantiti da materie prime, le loro stablecoin sono esempi di stablecoin off-chain. Inoltre, tutti i token vengono controllati, essendo valute stabili e possono esserne creati di nuovi solo da Tether

#### Binance Coin (BNB)

Binance Coin è disponibile sulla piattaforma di scambio di criptovalute Binance, insieme ad altre monete digitali disponibili per il trading. Binance Coin può essere utilizzato come tipo di valuta, ma anche come token che possono essere utilizzati per pagare le commissioni sulla borsa di Binance e per alimentare il DEX (scambio decentralizzato) di Binance per la costruzione di app.

* La moneta di Binance inizialmente girava sulla blockchain di Ethereum con lo standard ERC 20, ma da allora è diventata la moneta nativa della catena di Binance. È stata lanciata durante un'offerta iniziale di monete (ICO) nel luglio 2017 e ha un tetto massimo di 200 milioni di token BNB. BNB era inizialmente basato sulla rete Ethereum, ma ora è la valuta nativa della blockchain di Binance, la Binance chain.
* Vengono spesse eseguite operazioni di *burning* delle monete; Sebbene nell'ICO siano stati inizialmente offerti 100 milioni di token Binance, l'offerta totale attuale è inferiore. Ogni trimestre, Binance utilizza un quinto dei suoi profitti per riacquistare e distruggere definitivamente, o "bruciare", le monete Binance detenute nel suo tesoro; Binance ha regolarmente eseguito questi eventi di burning.

Esso è stato creato come token di utilità per le commissioni di trading scontate nel 2017. Da allora, però, il suo utilizzo si è esteso a numerose applicazioni su un ampio numero di piattaforme. Viene utilizzato per pagare le commissioni di transazione su Binance.com, Binance DEX e Binance Chain.

* Può essere utilizzato anche per i pagamenti (su siti come Crypto.com, Monetha e HTC); per prenotare viaggi (su siti come TravelbyBit, Trip.io e Travala.com); per scopi di intrattenimento (regali virtuali, pacchetti di carte, lotterie); per pagare servizi online (BitTorrent, Canva, Storm); e persino per scopi finanziari (per ottenere un prestito su ETHLend o per fare investimenti su Moeda).

Al di là dell'exchange Binance, Binance Coin ha diversi utilizzi, tra cui la possibilità di:

* Effettuare pagamenti con carta di credito
* Pagare l'organizzazione di viaggi (su siti web selezionati)
* Acquistare regali virtuali
* Elaborare pagamenti
* Effettuare investimenti
* Effettuare prestiti e trasferimenti
* Donare in beneficenza

#### USD Coin (USDC)

USD Coin è un'altra stablecoin e, come Tether, è ancorata al dollaro statunitense. Come Tether, anche USD Coin è ospitata sulla blockchain di Ethereum. L'idea alla base di USD Coin è quella di creare un dollaro "completamente digitale", che abbia la stabilità della valuta fiat statunitense ma che non richieda un conto bancario o che il possessore viva in un determinato Paese. Piuttosto che un investimento, USD Coin è concepito come denaro di uso quotidiano che può essere speso presso i commercianti su Internet.

#### Cardano (ADA)

Charles Hoskinson, il cofondatore di Ethereum, ha iniziato lo sviluppo di Cardano nel 2015 e ha lanciato la piattaforma nel 2017.

Cardano si è posizionata come alternativa a Ethereum. Entrambe le piattaforme sono utilizzate per applicazioni simili, come i contratti intelligenti, e hanno l'obiettivo di costruire un sistema connesso e decentralizzato. Essa si considera una versione aggiornata di Ethereum e si è autodefinita una piattaforma di "terza generazione", rispetto alle credenziali di "seconda generazione" di Ethereum.

* La piattaforma Cardano si basa sul protocollo di consenso *Ouroboros*. Ouroboros, creato da Cardano nella sua fase di fondazione, è il primo protocollo proof of stake (PoS) progettato per ridurre il dispendio energetico richiesto dal mining proof of work (PoW). A tal fine, elimina le enormi risorse di calcolo utilizzate dall'algoritmo di proof of work.
* La sua criptovaluta prende il nome da Augusta Ada Lovelace, contessa e matematica inglese del XIX secolo riconosciuta come la prima programmatrice di computer.
* Nel sistema PoS di Cardano, la puntata determina la capacità di un nodo di aprire blocchi sulla blockchain. Lo stake di un nodo è pari alla quantità di Ada che detiene a lungo termine. Una partecipazione è un interesse detenuto da un partecipante al pool, garantito da un Ada dato in pegno. L'Ada dato in pegno non può essere utilizzato o speso dal titolare perché è tenuto come garanzia per un comportamento onesto di convalida. Gli utenti con Ada in pegno ricevono ricompense sotto forma di commissioni di transazione. Le ricompense sono distribuite in base alla quantità di Ada che un utente ha puntato.
  + Gli utenti si uniscono alle staking pools - gruppi di possessori di Ada che hanno impegnato le loro monete - e lavorano insieme per aggiornare il libro mastro, aprire nuovi blocchi e guadagnare ricompense.
* Cardano ha implementato il supporto per gli smart contract nel 2021 con l'aggiornamento Alonzo.

Questo aggiornamento della testnet è stata la prima iterazione per portare agli utenti la scalabilità e i casi d'uso promessi. Questo aggiornamento ha permesso agli utenti di creare contratti intelligenti e token non fungibili (NFT) e di gestire più asset. Le versioni future e i fork porteranno i contratti intelligenti e altre funzionalità nella mainnet.

#### Solana (SOL)

Solana è una piattaforma blockchain progettata per ospitare applicazioni decentralizzate e scalabili. Fondata nel 2017, è un progetto open-source attualmente gestito dalla Solana Foundation con sede a Ginevra, mentre la blockchain è stata costruita da Solana Labs con sede a San Francisco.

Solana è molto più veloce in termini di numero di transazioni che può elaborare e ha commissioni di transazione significativamente più basse rispetto alle blockchain rivali come Ethereum.

* Anatoly Yakovenko, cofondatore di Solana, ha pubblicato nel novembre 2017 un white paper che descrive il concetto di proof-of-history (PoH). La PoH è una prova per verificare l'ordine e il passaggio di tempo tra gli eventi, ed è utilizzata per codificare il passaggio di tempo senza fiducia in un libro mastro.
* Nel white paper, Yakovenko osserva che le blockchain allora pubblicamente disponibili non si basavano sul tempo, con ogni nodo della rete che si basava sul proprio orologio locale senza conoscere gli orologi degli altri partecipanti alla rete. La mancanza di una fonte affidabile di tempo (cioè di un orologio standardizzato) significava che quando si usava il timestamp di un messaggio per accettarlo o rifiutarlo, non c'era alcuna garanzia che ogni altro partecipante alla rete avrebbe fatto la stessa identica scelta.
* L'architettura di Solana mira a dimostrare una serie di algoritmi software che eliminano il software come collo di bottiglia delle prestazioni quando viene combinato con una blockchain. La combinazione consente al throughput delle transazioni di scalare proporzionalmente alla larghezza di banda della rete.
  + Essa soddisfa tutti e tre gli attributi desiderabili per una blockchain: è scalabile, sicura e decentralizzata. La sua architettura descrive un limite superiore teorico di 710.000 TPS su una rete gigabit standard e 28,4 milioni di TPS su una rete da 40 gigabit.
* La blockchain di Solana opera su un modello di consenso sia proof-of-history (PoH) che proof-of-stake (PoS). PoS consente ai validatori (coloro che convalidano le transazioni aggiunte al libro mastro della blockchain) di verificare le transazioni in base al numero di monete o token in loro possesso; PoH permette a queste transazioni di essere marcate a tempo e verificate molto rapidamente. Essa rimane molto veloce, avendo sempre la modalità di consenso PoS, supportando gli smart contract e processando decine di migliaia di transazioni al secondo

#### Polkadot (DOT)

Diversi progetti blockchain negli ultimi anni si sono concentrati meno su applicazioni specifiche e più su miglioramenti generali legati all'infrastruttura. Tra questi c'è Polkadot (DOT), probabilmente uno dei progetti di maggior successo che lavora per migliorare la tecnologia fondamentale che alimenta le applicazioni decentralizzate (dApp).

* Il protocollo Polkadot cerca di abbattere le barriere tra i vari ecosistemi blockchain, consentendo una comunicazione senza intermediari tra queste reti. Polkadot può essere considerato come una rete di reti, che permette ad architetture blockchain anche molto diverse di interagire tra loro. Lo fa attraverso le parachain, ovvero blockchain specializzate che hanno funzionalità e token propri.

La rete utilizza l'algoritmo di consenso nominato proof-of-stake (PoS) ed è stata ispirata dal protocollo Ouroboros.

Gli esperti del settore e gli sviluppatori elencano tre ostacoli principali che impediscono la crescita della tecnologia blockchain: velocità, scalabilità e sicurezza. La maggior parte delle blockchain di prima generazione, termine alquanto generico, ha apportato miglioramenti incrementali nel corso del tempo, ma rimane inibita da limitazioni tecniche come la scalabilità.

* Questo ha introdotto il concetto delle parachain, che sono essenzialmente blockchain PoS e possono funzionare in modo indipendente, potendo essere completamente personalizzate dal proprietario. Sono focalizzate su applicazioni con funzionalità e logiche di programmazione limitate a sé stesse. Queste catene fungono da livello di governance della rete e sono un meccanismo di gestione.
  + Ciò che lega queste parachains è la catena di relay (relay chain), responsabile della sicurezza, del consenso e dell'interoperabilità della rete condivisa. La catena di relay convalida i dati e ne garantisce la comprensibilità; in altre parole, è responsabile del raggiungimento del consenso e dell'esecuzione delle transazioni.
* Polkadot contiene anche dei ponti, che collegano le blockchain e permettono di trasferire i dati tra di esse. I ponti stabiliscono l'interoperabilità e possono essere utilizzati per connettersi a reti esterne come Bitcoin (BTC) ed Ethereum.
  + Un'altra parte della rete è costituita dai parathread, una versione su scala ridotta dei parachain che funziona secondo un modello a pagamento. È utile per le blockchain che non richiedono una connettività continua alla rete Polkadot.
* Per quanto riguarda il consenso, All'interno del modello di consenso nominato proof-of-stake di Polkadot esistono tre soggetti interessati. Sono i nominatori, i validatori e i collatori:
  + Nominatori: assicurano la catena di relay e selezionano i validatori affidabili.
  + Validatori: sono responsabili dell'imputazione del DOT, della convalida delle prove dei collatori e della partecipazione al consenso.
  + Collatori: richiedono un coinvolgimento minore rispetto ai validatori: sono responsabili di tenere un registro delle transazioni paracadutate valide e di inviarle ai validatori della catena di relay.
* Il token DOT ha un duplice utilizzo: per la partecipazione alla rete per rafforzarne la sicurezza, detta anche bonding, e nel meccanismo di governance.
  + Staking: Come altre blockchain proof-of-stake, il DOT staking incentiva i partecipanti alla rete ad agire onestamente, detenendo il DOT come garanzia per il buon comportamento. Le ricompense vengono assegnate a coloro che puntano i loro DOT.
  + Governance: I titolari di DOT hanno il diritto e la possibilità di partecipare alle votazioni attraverso i referendum, con uno schema di voto ponderato in base alla partecipazione.

## Wallet

Un portafoglio blockchain è un portafoglio di criptovalute che consente agli utenti di gestire diversi tipi di criptovalute, ad esempio Bitcoin o Ethereum. Un portafoglio blockchain aiuta a scambiare facilmente i fondi. Le transazioni sono sicure, poiché sono firmate crittograficamente. Il portafoglio è accessibile dai dispositivi web, compresi quelli mobili, e la privacy e l'identità dell'utente sono mantenute. l wallet, anche se si chiama portafoglio, in realtà è un portachiavi, perché non contiene realmente gli asset, ma conserva solo le chiavi per accedervi.

* L’indirizzo è la sola informazione per poter ricevere un pagamento e sarà composto da 27-34 caratteri alfanumerici. La chiave privata è invece composta da 51 caratteri.
* In primo luogo, occorre scaricare il portafoglio blockchain e creare un account. Dopodiché si riceve una coppia di chiavi, composta da una chiave pubblica e da una chiave privata. Queste chiavi vi aiuteranno a ricevere e trasferire fondi dal vostro conto.
  + La *chiave pubblica* viene fornita per inviare messaggi e ricevere pagamenti. Ogni volta che qualcuno desidera inviarvi dei bitcoin, o qualsiasi altra criptovaluta, dovrà includere la chiave pubblica e la sua chiave privata per completare la transazione. Essa è un indirizzo alfanumerico formato da 32 caratteri
  + La *chiave privata* è per un portafoglio di criptovalute simile al PIN del bancomat o al TAN dell'Online Banking per un conto bancario. Ogni portafoglio ha una o più chiavi private uniche. È nota solo al proprietario del portafoglio e viene utilizzata per dimostrare che possiede legittimamente il conto e i fondi contenuti e può inviare transazioni. Ogni transazione crittografica inviata è firmata con la chiave privata del portafoglio, che tuttavia non viene rivelata a terzi.
* La *moltiplicazione della curva ellittica (elliptic curve multiplication)* è un'operazione matematica fondamentale utilizzata per la creazione di chiavi pubbliche e private per l'utilizzo della blockchain.

Viene utilizzata per generare queste coppie di chiavi pubbliche e private. Il processo consiste nel partire da un punto scelto su una curva ellittica e nel moltiplicarlo per un grande numero casuale (la chiave privata) per ottenere un nuovo punto della curva (la chiave pubblica). Questo processo è matematicamente reversibile: la chiave privata può essere utilizzata per generare la chiave pubblica, ma la chiave pubblica non può essere utilizzata per determinare la chiave privata.

* La sicurezza della crittografia a curva ellittica risiede nel fatto che è estremamente difficile invertire il processo e calcolare la chiave privata dalla chiave pubblica. Questo rende praticamente impossibile per chiunque rubare fondi da un portafoglio senza avere accesso alla chiave privata.
* Se un utente dovesse perdere (e/o dimenticare) la chiave privata del suo portafoglio, non potrebbe più accedere, gestire o inviare i fondi contenuti nel portafoglio. In breve, i fondi andrebbero irrimediabilmente persi. Per evitare che ciò accada, nei portafogli di criptovalute è presente un meccanismo di backup chiamato frase segreta (talvolta indicata anche come frase mnemonica, seme di backup, frase di recupero).
* La *frase segreta* è un insieme di 12-24 parole che contiene tutte le informazioni necessarie per recuperare e accedere a tutti i fondi di un portafoglio crittografico. Può essere utilizzata per ricavare la chiave privata del portafoglio, poiché la frase segreta è una rappresentazione del numero casuale della chiave privata. I fornitori di portafogli indicheranno agli utenti di annotare la frase segreta generata su un pezzo di carta e di conservarla in modo sicuro, fuori dalla portata di terzi.
* Ciascun wallet ha un suo indirizzo (*wallet address*). L'indirizzo del portafoglio è matematicamente derivato dalla chiave pubblica del portafoglio attraverso una funzione unidirezionale chiamata "hashing". L'indirizzo del portafoglio è una rappresentazione più breve della parte finale della chiave pubblica e di solito ha una lunghezza di 160 bit. L'indirizzo del portafoglio e la chiave pubblica non sono la stessa cosa, poiché l'indirizzo del portafoglio è la parte finale della chiave pubblica.
* Gli indirizzi sono *pseudonimi* dato che non si sa esattamente a quale entità fisica o meno possano corrispondere dato il numero di caratteri. Non è completamente anonimo, ma non si può affermare che sia identificato nello specifico da una certa persona.

Ci sono due tipi principali di wallet:

* Gli *hot wallet* vengono utilizzati con l'aiuto di Internet e sono considerati di facile utilizzo. Il problema principale è la sicurezza. A causa della connettività a Internet, sono più inclini al rischio di hacking. Per questo motivo gli esperti non consigliano di tenere una grande somma in un portafoglio hot. Ve ne sono di vario tipo:
  + Mobile wallet - I portafogli mobili sono piuttosto comodi per gli utenti, poiché l'interfaccia è molto facile da usare. È possibile installare facilmente un portafoglio mobile sul proprio smartphone. Gli investitori tendono a utilizzare questo portafoglio per effettuare le transazioni quotidiane con l'aiuto del codice QR che ricevono. I portafogli mobili possono essere molto vantaggiosi per molti investitori. Possono verificarsi problemi come virus e infezioni da malware.
  + Desktop wallet - Questi portafogli possono essere installati sul desktop come software. Possono essere un'ottima fonte per conservare le criptovalute. Assicuratevi di avere un antivirus sul vostro desktop, in modo da ridurre il rischio di hacking e mantenere la sicurezza. Hanno un'interfaccia utente molto amichevole e forniscono una privacy di prim'ordine. Inoltre, sono considerati il terzo portafoglio più sicuro per l'archiviazione di criptovalute.
  + Web wallet - Questi portafogli sono accessibili solo attraverso un browser web. Possono essere utilizzati in modalità hosted o non-hosted. Gli esperti raccomandano l'uso di un metodo non ospitato perché consente di mantenere il controllo sulle criptovalute. Il fatto che si possa accedere solo attraverso un browser web lo rende molto esposto agli attacchi. È considerato il portafoglio meno sicuro, quindi vi consigliamo di non investire grandi somme attraverso un portafoglio web. È perfetto se si vogliono fare solo piccoli investimenti o transazioni veloci.
* I *cold wallet* sono considerati un'opzione altamente sicura per conservare le criptovalute. Uno degli aspetti migliori di questi tipi di portafogli è che non sono collegati a Internet. Si collegano a internet solo quando si effettuano le transazioni. Molti investitori in criptovalute preferiscono utilizzare i portafogli a freddo. Non solo perché sono sicuri, ma anche facili da usare. La cosa migliore è che sono dotati di assistenza clienti. Essi comprendono:
  + Hardware wallet - Un portafoglio hardware utilizza dispositivi hardware per memorizzare in modo sicuro le chiavi private. È considerato uno dei portafogli migliori e sicuri per conservare le chiavi lontano dalla portata degli hacker. In genere, il costo si aggira tra i 70 e i 150 dollari. Può sembrare un po' costoso, ma ne vale la pena perché offre un'elevata sicurezza per le chiavi. Per quanto riguarda l'aspetto, si tratta di un dispositivo USB composto da pulsanti laterali e da uno schermo OLED. L'interfaccia è molto migliore rispetto a quella del portafoglio cartaceo. Ma se siete principianti, non scegliete un portafoglio hardware. Potrebbe non essere piacevole per gli occhi.
  + Paper wallet - I portafogli di carta sono simili ai portafogli hardware. Ma conservano le chiavi private su pezzi di carta invece che su un dispositivo hardware come l'USB. Sono anche molto sicuri e per gli hacker è molto difficile attaccarli. I portafogli cartacei erano piuttosto famosi nel mercato delle criptovalute fino all'arrivo dei portafogli hardware. I portafogli hardware avevano molte caratteristiche migliori rispetto a quelli cartacei. È proprio questo che ha spinto molti investitori a smettere di usare i portafogli cartacei e a optare per un'opzione migliore.
* Esistono anche dei wallet deterministici, che rappresentano una possibile soluzione ai rischi di furto dato che generano un numero infinito di chiavi da un unico codice seme (seed); da esso, quando salvato, saranno generati chiavi ed indirizzi.

Alcuni esempi di wallet utilizzati:

1. Metamask: È un wallet molto popolare che supporta la rete Ethereum e funziona come una estensione del browser web, permettendo agli utenti di interagire con applicazioni decentralizzate (dApps) e di effettuare transazioni sulla blockchain di Ethereum direttamente dal proprio browser. Caratteristiche distintive includono una buona usabilità, una interfaccia intuitiva e la possibilità di importare o esportare le proprie chiavi private.
2. Ledger Nano S: È un hardware wallet che permette di conservare criptovalute offline in un dispositivo fisico sicuro. Questo dispositivo si connette al computer o allo smartphone tramite USB e permette di eseguire transazioni in modo sicuro. Caratteristiche distintive includono la possibilità di conservare più criptovalute, una interfaccia facile da usare e l'alta sicurezza offerta dalla conservazione offline.
3. Trezor: È un altro hardware wallet che permette di conservare criptovalute offline in un dispositivo fisico. A differenza di Ledger Nano S, Trezor ha una schermata OLED per visualizzare le transazioni in tempo reale e verificare l'autenticità delle transazioni. Caratteristiche distintive includono la compatibilità con più criptovalute, l'interfaccia semplice da usare e l'elevata sicurezza offerta dalla conservazione offline.
4. Exodus: È un wallet software che supporta diverse criptovalute tra cui Bitcoin, Ethereum, Litecoin e molti altri. Exodus ha una interfaccia molto accattivante e facile da usare, con grafici e funzioni di analisi per tenere traccia del valore delle proprie criptovalute. Caratteristiche distintive includono la possibilità di scambiare criptovalute all'interno del wallet e la presenza di una funzione di backup automatico.

### Crittografia nelle blockchain

Come sopra descritto, si tratta di avere una coppia chiavi pubblica e privata (quindi asimmetrica), fatta di una chiave distribuita ed una chiave personale/segreta, al fine di poter permette senza scambio di informazioni sul canale di comunicazione l’autenticità dell’informazione, rendendo valida l’informazione trasmessa (*integrità*), dando garanzie sull’identità delle parti in gioco (*autenticazione*) e proibendo di negare la transazione, anzi validandola subito (*non ripudiabilità*).

* Un algoritmo usato, ad esempio sulla rete Bitcoin, è l’algoritmo di firma digitale su Curve Ellittiche (ECDSA) è un meccanismo crittografico utilizzato per garantire l'autenticità e l'integrità delle transazioni nella blockchain. L'ECDSA utilizza una coppia di chiavi crittografiche, una pubblica e una privata, come nella firma asimmetrica tradizionale.
* L'ECDSA funziona utilizzando le proprietà matematiche delle curve ellittiche. In pratica, la chiave privata dell'utente viene utilizzata per generare una firma digitale univoca per la transazione. Questa firma viene quindi verificata utilizzando la chiave pubblica dell'utente.
* L'ECDSA viene utilizzato in diverse blockchain, tra cui Bitcoin, per garantire la sicurezza e l'integrità delle transazioni. In particolare, l'ECDSA viene utilizzato nella verifica delle firme digitali delle transazioni nella blockchain Bitcoin.
* Quando un utente invia una transazione Bitcoin, questa viene firmata digitalmente utilizzando la chiave privata dell'utente e l'algoritmo ECDSA. La firma digitale viene quindi trasmessa alla rete Bitcoin insieme alla transazione. Quando la transazione viene ricevuta dalla rete, la firma viene verificata utilizzando la chiave pubblica dell'utente. Se la verifica è positiva, la transazione viene considerata valida e aggiunta alla catena di blocchi.

Ricapitoliamo quanto detto sopra con un esempio pratico di utilizzo di crittografia asimmetrica nelle transazioni blockchain è quello che avviene quando un utente A invia una transazione a un utente B sulla blockchain.

* Innanzitutto, l'utente A utilizza la sua chiave privata per firmare digitalmente la transazione. La firma digitale viene poi trasmessa insieme alla transazione alla rete blockchain.
* Quando la transazione viene ricevuta dalla rete, la chiave pubblica dell'utente A viene utilizzata per verificare la firma digitale. Se la firma è valida, la transazione viene aggiunta alla catena di blocchi e viene trasmessa all'utente B.
* L'utente B può quindi utilizzare la chiave pubblica dell'utente A per verificare l'autenticità della transazione e la validità della firma digitale. In questo modo, l'utente B può essere certo che la transazione è stata effettivamente inviata dall'utente A e che non è stata alterata durante il processo di trasmissione.

Un esempio di curva che usa ECDSA è *Secp256k1* è il nome della curva ellittica utilizzata da Bitcoin per implementare la sua crittografia a chiave pubblica. Tutti i punti di questa curva sono chiavi pubbliche Bitcoin valide. Quando un utente desidera generare una chiave pubblica utilizzando la propria chiave privata, moltiplica la propria chiave privata, un numero elevato, per il Punto Generatore, un punto definito sulla curva secp256k1.

La curva Secp256k1 è definita su un campo finito di 256 bit ed è rappresentata dall'equazione:

dove p è un numero primo molto grande e le coordinate x e y sono interi nel campo finito definito da p.

Questa curva ellittica viene usata dato che non ha una struttura “del tutto” casuale, ma grazie alle costanti e , il modulo del campo, punto base della curva e ordine del punto base, è possibile accelerare i calcoli essendo la funzione dipendente da questa serie di parametri. Grazie alla scelta di numeri grandi, si garantisce sicurezza e un calcolo più veloce nelle sue parti.

* Molto usate anche le funzioni di hash, che non sono iniettive e hanno la proprietà di non permettere di trovare due controimmagini diverse per una stessa immagine (preimage resistant). Il loro principale vantaggio è essere calcolate in modo efficiente; data infatti una stringa di qualsiasi lunghezza, si vuole generare un *digest* di messaggio a lunghezza fissata, definito come . Come detto sopra, sono facilmente rimodificabili; ciò rende difficile la collisione, cioè avere due stringhe di hash uguali generate da due stringhe diverse in partenza (quindi, due stringhe

In particolare hanno i seguenti effetti:

* + Effetto valanga (avalanche effect)- Una piccola modifica dei dati può produrre un risultato significativamente diverso.
  + Unicità - Ogni input ha un output unico.
  + Determinismo - Qualsiasi input avrà sempre lo stesso output se passato attraverso la funzione hash.
  + Rapidità - L'output può essere generato in un tempo molto ridotto.
  + Non è possibile il reverse engineering, cioè non si può generare l'input avendo a disposizione l'output e la funzione hash.
  + Gli algoritmi più usati a livello hash nelle blockchain sono:
    - *SHA-256* (Secure Hash Algorithm 256), che utilizza una rete di funzioni di compressione, chiamate "funzioni di compressione di blocco", per elaborare i dati di input e produrre un hash di 256 bit come output. La funzione di compressione prende in input un blocco di dati di 512 bit e produce un hash di 256 bit. La funzione di compressione utilizza una combinazione di operazioni logiche, tra cui l'operazione XOR (or esclusivo), l'operazione AND (and logico), l'operazione NOT (not logico) e l'operazione ADD (somma) per produrre l'hash finale.
    - *SHA-3* (Secure Hash Algorithm 3), funzione di hashing crittografica alternativa a SHA-256, sviluppata dal National Institute of Standards and Technology (NIST). SHA-3 è basato su un algoritmo chiamato Keccak, che utilizza una struttura a "sponge function" per produrre l'hash.
      * La funzione di hash SHA-3 utilizza un'operazione di permutazione non lineare per manipolare i dati di input. Durante l'elaborazione, i dati vengono assorbiti in un "sponge", ovvero una struttura di dati che viene manipolata tramite l'operazione di permutazione. Una volta che i dati sono stati completamente assorbiti, l'output viene estratto dallo sponge.
    - L'algoritmo di hash RIPEMD-160 è una funzione di hashing crittografica che viene utilizzata in diverse criptovalute, tra cui Bitcoin. Il suo nome deriva dalla sua struttura "RACE Integrity Primitives Evaluation Message Digest" e dalla sua lunghezza di output di 160 bit.
      * L'algoritmo di hash RIPEMD-160 è basato su una rete di funzioni di compressione, che utilizza una combinazione di operazioni di permutazione, rotazione e operazioni logiche per produrre l'output finale. La funzione di compressione prende in input un blocco di dati di 512 bit e produce un hash di 160 bit.
      * La funzione di compressione consiste di due fasi principali: una fase di pre-processing e una fase di elaborazione. Durante la fase di pre-processing, il blocco di dati viene diviso in 16 parole da 32 bit ciascuna e queste parole vengono mescolate e manipolate tramite operazioni di permutazione e rotazione. Durante la fase di elaborazione, le parole vengono sottoposte a diverse operazioni logiche, tra cui l'operazione XOR (or esclusivo), l'operazione AND (and logico), l'operazione NOT (not logico) e l'operazione ADD (somma).
      * L'output dell'algoritmo di hash RIPEMD-160 è una stringa di 160 bit che rappresenta l'hash del messaggio originale. L'hash viene utilizzato per garantire l'integrità e la sicurezza dei dati nella blockchain, poiché qualsiasi modifica ai dati originali comporterebbe una modifica all'hash corrispondente.
  + Sono usate alcune strutture per verificare l’integrità dei dati; un esempio sono gli alberi di Merkle (Merkle trees*)*, la cui idea è di suddividere i dati in blocchi più piccoli e di calcolare l'hash di ciascun blocco. Gli hash dei blocchi vengono poi combinati a coppie e nuovamente "hashati" fino a quando non rimane un solo hash, noto come root hash o Merkle root (questa è la hash di tutte le hash del Merkle tree). L'albero di Merkle può essere rappresentato come un albero binario in cui le foglie rappresentano gli hash dei blocchi e i nodi intermedi rappresentano gli hash delle coppie di blocchi.
    - Viene definito come *Merkle proof* il processo di attraversamento di un albero Merkle da una foglia alla radice, con l'hashing di ogni livello con il precedente per produrre un hash unico per la struttura dell'albero
    - Le prove di Merkle vengono utilizzate per decidere i seguenti fattori:
      * Se i dati appartengono all'albero di Merkle
      * Per dimostrare in modo conciso la validità dei dati che fanno parte di un insieme di dati senza memorizzare l'intero insieme di dati.
      * Per garantire la validità di un certo insieme di dati inclusi in un insieme di dati più grande senza rivelare né l'insieme completo né il suo sottoinsieme.
    - Assieme a questa struttura, è importante citare che viene usato anche un particolare tipo di albero in Ethereum. Per introdurla, occorre parlare del *trie*. Esso è una struttura dati ad albero usata per memorizzare e cercare array associativi a coppie chiave valore.
      * Introduciamo il trie PATRICIA, che è una variante speciale del trie radix 2 (binario), in cui invece di memorizzare esplicitamente ogni bit di ogni chiave, i nodi memorizzano solo la posizione del primo bit che differenzia due sottoalberi. Esso utilizza nodi compressi per ridurre la quantità di memoria richiesta per la memorizzazione. In un trie PATRICIA, i nodi con un solo figlio vengono fusi con il loro nodo padre e gli archi tra di essi sono etichettati con i caratteri che rappresentano. Questa tecnica di compressione consente di ottenere una struttura di dati più efficiente dal punto di vista dello spazio, pur mantenendo i tempi di ricerca rapidi di un trie standard.
      * Ethereum combina le strutture per realizzare un Merkle Patricia Trie. In Ethereum, ogni nodo dell'albero è referenziato dal suo valore di hash, come un albero di Merkle. L'hash della radice funge da impronta digitale dell'intero insieme di dati da essa rappresentato. Se un valore-chiave viene aggiornato, anche la radice Merkle cambia. Le chiavi di Ethereum sono rappresentate in formato esadecimale, con 16 caratteri possibili (0-9 e A-F).
      * Il Merkle Patricia trie comprende 4 tipi di nodi:
        + null: Un nodo inesistente, rappresentato come una stringa vuota.
        + ramo: Un nodo che ha collegamenti a un massimo di 16 note figlio distinte, corrispondenti a 16 caratteri esadecimali. Ha anche un campo valore.
        + foglia: Un "nodo finale" che contiene la parte finale della chiave e un valore.
        + estensione: Un nodo "scorciatoia" che memorizza una parte della chiave basata su un prefisso comune e un collegamento al nodo successivo.
      * Ethereum utilizza tre strutture trie. Ne vengono usate diverse in quanto dobbiamo gestire dati permanenti come le transazioni in un blocco impegnato e dati temporanei come il saldo del conto che viene aggiornato. Le strutture dati sono richieste per memorizzarli separatamente. In Ethereum, ogni intestazione di blocco memorizza le radici di tre strutture trie: stateRoot, transactionRoot e receiptsRoot.
        + State trie: Lo state trie / world state trie rappresenta una mappatura tra gli indirizzi dei conti e gli stati dei conti. Lo stato del conto comprende il saldo, il nonce, il codeHash e la storageRoot. La storageRoot è la radice di un trie di memorizzazione del conto, che memorizza i dati contrattuali associati a un conto. La chiave è un indirizzo a 160 bit di un conto Ethereum e il valore è lo stato del conto. Il nodo radice dipende da tutti i dati interni e il suo valore di hash funge da identificatore unico per l'intero stato del sistema.
        + I conti Ethereum vengono aggiunti al trie di stato solo dopo che una transazione è stata registrata con successo su quel conto. Il trie dello stato mondiale memorizza alcuni dati temporanei che vengono aggiornati, modificando frequentemente l'hash della radice. StateRoot è l'hash del nodo radice dello state trie, dopo che tutte le transazioni sono state eseguite e impegnate.
        + Transaction trie: viene creato sull'elenco delle transazioni all'interno di un blocco. Il percorso di una specifica transazione nel trie delle transazioni è tracciato in base alla posizione della transazione all'interno del blocco.

Una volta estratta, la posizione di una transazione in un blocco non cambia. Quindi il trie delle transazioni non viene mai aggiornato. Questo è simile alla rappresentazione ad albero di Merkle delle transazioni in Bitcoin e la verifica delle transazioni può essere fatta in modo simile. Il transactionRoot è l'hash del nodo radice del trie della transazione.

* + - * + Receipt trie: la ricevuta registra il risultato della transazione eseguita con successo. Consiste di quattro elementi: codice di stato della transazione, gas cumulativo utilizzato, log della transazione, filtro Bloom (una struttura dati per trovare rapidamente i log).

La chiave è un indice delle transazioni nel blocco e il valore è la ricevuta della transazione. Come il trie delle transazioni, il trie delle ricevute non viene mai aggiornato. Il receiptsRoot è l'hash del nodo radice della struttura trie popolato con le ricevute di ogni transazione nella parte dell'elenco delle transazioni del blocco;

* + L'utilizzo dell'albero di Merkle consente di verificare l'integrità di una grande quantità di dati in modo efficiente. Infatti, invece di dover verificare l'integrità di tutti i dati, è sufficiente verificare l'integrità del Merkle root. Se il Merkle root viene alterato, anche gli hash dei blocchi vengono alterati e di conseguenza viene rilevata una violazione dell'integrità.
  + Gli alberi di Merkle sono ampiamente utilizzati nella blockchain, in cui i blocchi sono composti da un gran numero di transazioni. In particolare, il Merkle root viene incluso nell'header di ogni blocco e consente di verificare l'integrità di tutte le transazioni contenute nel blocco. In questo modo, gli utenti della blockchain possono essere certi che le transazioni sono state eseguite correttamente e che non sono state alterate.

## Mining dei blocchi

* Ci sono vari modi di guadagnare criptovalute:
  + Vendere beni o servizi ed essere ricompensati con la valuta di interesse
  + Accedere a servizi di cambi valute reali-cripto
  + Partecipare al protocollo di validazione/inserimento transazioni sulla rete, ottenendo una commissione per ogni transazione trattata; ogni 10 minuti vengono generate e generate nuove cripto (caso Bitcoin) e chi risolve per primo il processo di creazione di nuovi blocchi risolvendo un problema crittografico ne viene ricompensato; è quello che approfondiremo in questa sezione
* Il processo di mining permette di creare nuovi blocchi sulla catena. In una blockchain ogni blocco ha un nonce e un hash unici, ma fa anche riferimento all'hash del blocco precedente nella catena, quindi estrarre un blocco non è facile, soprattutto nelle catene di grandi dimensioni. L'obiettivo del processo di mining nel contesto della blockchain è aggiungere nuove transazioni al registro della blockchain e convalidarle. Questo obiettivo viene raggiunto attraverso un processo chiamato *consenso*, che prevede la risoluzione di complessi puzzle matematici utilizzando la potenza di calcolo.
* I minatori (miners) utilizzano un software speciale per risolvere il problema matematico incredibilmente complesso di trovare un nonce che generi un hash accettato. Poiché il nonce è di soli 32 bit e l'hash di 256, ci sono circa quattro miliardi di possibili combinazioni nonce-hash che devono essere estratte prima di trovare quella giusta. Quando ciò accade, si dice che i minatori hanno trovato il "nonce d'oro" e il loro blocco viene aggiunto alla catena. Un computer normale potrebbe non esserne pienamente in grado; per esempio, si tratterebbe di individuare il numero corretto tra 4 miliardi di possibilità. Il fatto di avere correttamente individuato il blocco dona criptovalute o ricompense similari.
* Per modificare un blocco precedente nella catena è necessario estrarre nuovamente non solo il blocco con la modifica, ma anche tutti i blocchi successivi. Ecco perché è estremamente difficile manipolare la tecnologia blockchain. Si pensi alla "sicurezza in matematica", poiché trovare i nonce d'oro richiede un'enorme quantità di tempo e di potenza di calcolo.
* Quando un blocco viene estratto con successo, la modifica viene accettata da tutti i nodi della rete e il miner viene ricompensato finanziariamente. Il primo minatore che risolve il puzzle e aggiunge un nuovo blocco alla blockchain viene ricompensato con un blocco di criptovaluta di nuovo conio. Questo incentiva i minatori a contribuire con la loro potenza di calcolo alla rete e aiuta a distribuire equamente la criptovaluta tra i partecipanti alla rete.

### Perché fare mining

Il processo di mining può diventare molto complesso e un normale desktop o PC non è sufficiente. Per questo motivo, è necessario un set unico di hardware e software che funzioni bene per l'utente. È utile avere un set personalizzato specifico per l'estrazione di determinati blocchi.

Questo succede in particolare per tre motivazioni principali:

1. *Meccanismo di consenso proof-of-work*: Molte reti blockchain, come Bitcoin, utilizzano un meccanismo di consenso proof-of-work per convalidare le transazioni e aggiungere nuovi blocchi alla blockchain. Ciò richiede la risoluzione di un complesso puzzle matematico, noto come funzione hash, utilizzando la potenza di calcolo. Il primo miner che risolve il puzzle viene ricompensato con un blocco di criptovaluta di nuovo conio. La difficoltà del rompicapo aumenta nel tempo con l'ingresso di altri minatori nella rete, che richiedono maggiore potenza di calcolo per essere risolti.
2. *Competizione tra minatori*: L'estrazione di blockchain è un processo competitivo, con i minatori che competono tra loro per risolvere il puzzle ed essere i primi ad aggiungere un nuovo blocco alla catena. Questa competizione crea una corsa alla potenza di calcolo, in quanto i minatori cercano di superarsi l'un l'altro con hardware più potente per aumentare le loro possibilità di vincere la ricompensa.
3. *Requisiti di elaborazione e archiviazione*: Oltre alla potenza di calcolo necessaria per risolvere il puzzle, il mining per la blockchain richiede anche una quantità significativa di capacità di elaborazione e di archiviazione per convalidare e registrare le transazioni. Ciò richiede hardware potente, come schede grafiche di fascia alta e piattaforme di mining specializzate, per gestire il carico di lavoro in modo efficiente.

Il processo di estrazione può essere suddiviso in tre categorie:

1. Estrazione individuale

Quando l'attività di mining viene svolta da un individuo, è necessaria la registrazione dell'utente come minatore. Non appena avviene una transazione, a tutti i singoli utenti della rete blockchain viene sottoposto un problema matematico da risolvere. Il primo che lo risolve viene premiato.

Una volta trovata la soluzione, tutti gli altri minatori della rete blockchain convalideranno il valore decifrato e lo aggiungeranno alla blockchain. In questo modo, la transazione viene verificata.

1. Mining in pool

Nel pool mining, un gruppo di utenti lavora insieme per approvare la transazione. A volte, la complessità dei dati crittografati nei blocchi rende difficile per un utente decifrare i dati codificati da solo. Pertanto, un gruppo di minatori lavora in squadra per risolvere il problema. Dopo la convalida del risultato, la ricompensa viene divisa tra tutti gli utenti.

1. Mining in cloud

Il cloud mining elimina la necessità di hardware e software. È un metodo senza problemi per estrarre i blocchi. Con il cloud mining, la gestione di tutti i macchinari, la tempistica degli ordini o la vendita dei profitti non sono più una preoccupazione costante.

### Come si realizza il mining

I minatori utilizzano un software specializzato per partecipare al processo di estrazione e convalidare le transazioni sulla blockchain. Il software è progettato per funzionare con la rete blockchain specifica su cui viene effettuato il mining e comprende in genere i seguenti componenti:

* *Client di mining*: Il client di mining è l'applicazione software principale utilizzata dai minatori per partecipare al processo di mining. Collega l'hardware del miner alla rete blockchain e comunica con gli altri nodi della rete per convalidare le transazioni e aggiungere nuovi blocchi alla catena.
* *Portafoglio*: Un portafoglio è utilizzato per memorizzare la criptovaluta guadagnata dal miner come ricompensa per la convalida delle transazioni e l'aggiunta di nuovi blocchi alla blockchain. Il portafoglio è solitamente incluso nel software di mining e può essere utilizzato anche per inviare e ricevere criptovalute.
* *Software di pool di mining*: Alcuni minatori scelgono di unirsi a pool di mining, che sono gruppi di minatori che uniscono la loro potenza di calcolo per aumentare le possibilità di guadagnare una ricompensa. Il software del pool di mining viene utilizzato per gestire il pool e distribuire le ricompense tra i minatori partecipanti.
* *Driver hardware*: I minatori utilizzano hardware specializzato, come ASIC (Application-Specific Integrated Circuits), GPU (Graphics Processing Units) o CPU (Central Processing Units), per risolvere i rompicapo matematici richiesti dal mining. Il software di mining deve essere compatibile con l'hardware specifico utilizzato dal miner e richiede l'installazione dei driver appropriati sul sistema.

Approfondiamo ad esempio gli ASIC, sviluppati a partire dal 2012. I miner ASIC sono progettati specificamente per fornire le migliori prestazioni possibili alle criptovalute cui sono destinati. Sono progettati esclusivamente per l'estrazione di criptovalute; a parte questo, la loro potenza è di scarsa utilità.

* Il processo di produzione degli ASIC è altamente specializzato e prevede una serie di fasi complesse, tra cui la progettazione, la fabbricazione e il collaudo. Il processo di progettazione prevede la creazione di un layout di circuito personalizzato e la sua ottimizzazione per ottenere le massime prestazioni ed efficienza. Una volta finalizzato il progetto, il circuito viene fabbricato utilizzando tecniche di produzione avanzate come la litografia, l'incisione e la deposizione.
* Dopo la produzione, gli ASIC vengono testati per garantire che soddisfino le specifiche richieste per il mining di criptovalute. Questo processo di verifica prevede che gli ASIC vengano sottoposti a una serie di test rigorosi per garantire che possano eseguire i calcoli richiesti in modo accurato ed efficiente. Le parti tipiche che compongono una postazione da ASIC mining sono il software per il mining, il chip ASIC, una fonte di alimentazione e le ventole per raffreddare il sistema.
* Nel complesso, gli ASIC sono un componente critico dell'ecosistema di mining di criptovalute e sono essenziali per i minatori che vogliono competere in un ambiente di mining altamente competitivo. Il loro design specializzato e l'elevata potenza di elaborazione li rendono uno strumento prezioso per il mining di criptovalute in modo efficiente e redditizio. Di fatto, sono molto costosi e hanno un’elevata potenza di calcolo che si traduce in un alto dispendio e consumo di energia.

Qualche esempio di dispositivo:

* <https://www.hola-cripto.com/corso-criptovalute/cosa-e-asic-mining-e-come-funziona/>
* <https://academy.bit2me.com/it/que-son-mineros-asic/>

### Considerazioni sul mining

Il mining nel contesto della blockchain e delle criptovalute si riferisce al processo di aggiunta di nuove transazioni alla blockchain attraverso l'uso della potenza di calcolo per risolvere complessi problemi matematici. Questo processo richiede una quantità significativa di energia, che può portare a preoccupazioni circa il suo impatto ambientale.

Alcuni sostengono che il mining è volutamente dispendioso perché la difficoltà dei problemi matematici da risolvere è intenzionalmente elevata per limitare la velocità di aggiunta di nuovi blocchi alla blockchain. Questo crea una competizione tra i minatori per risolvere questi problemi il più velocemente possibile, il che fa aumentare il consumo di energia in quanto i minatori utilizzano sempre più potenza di calcolo per cercare di superarsi a vicenda.

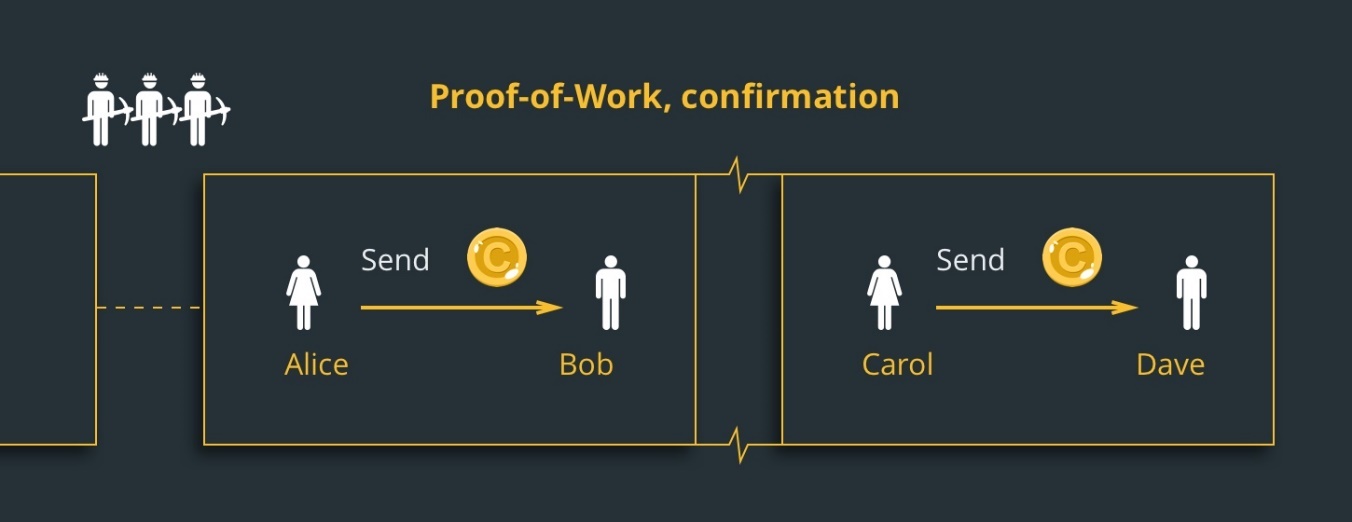
Tuttavia, è importante notare che il consumo di energia associato all'attività di mining non è del tutto dispendioso, in quanto ha un'importante funzione di protezione della rete blockchain e di prevenzione delle frodi. Inoltre, alcuni sostengono che, man mano che le fonti di energia rinnovabile diventano più diffuse e convenienti, l'impatto ambientale del mining diminuirà. Inoltre, è certamente più economico dell’impiego di personale umano e previene i malintenzionati.

Nel complesso, sebbene vi siano valide preoccupazioni sull'impatto ambientale dell'attività di mining nel contesto della blockchain e delle criptovalute, non è esatto affermare che si tratti di uno spreco intenzionale. La complessità dei problemi matematici da risolvere è una componente necessaria della sicurezza della blockchain e si stanno compiendo sforzi per rendere il mining più efficiente dal punto di vista energetico e sostenibile per l'ambiente.

## Algoritmi di consenso

### Proof of Work

Il Proof of Work (PoW) (noto anche come mining in effetti, ma ha un suo significato separato e preciso) è un algoritmo di consenso utilizzato dalle blockchain per confermare la validità delle transazioni e creare nuovi blocchi. Il PoW è stato introdotto per la prima volta con la creazione di Bitcoin, ma è stato successivamente adottato da numerose altre criptovalute e blockchain.



Riferimento: <https://cointelegraph.com/explained/proof-of-work-explained>

* In poche parole, il Proof of Work funziona in questo modo: i partecipanti alla rete, detti miner, risolvono un complesso problema matematico per creare un nuovo blocco di transazioni sulla blockchain. Il problema matematico consiste nella ricerca di un hash (una stringa di numeri e lettere) che soddisfa una determinata condizione, detta target. La condizione target richiede che l'hash del nuovo blocco sia inferiore a un determinato valore. Questo valore viene stabilito in base alla difficoltà della blockchain, che viene regolata automaticamente per mantenere il tempo medio di creazione di un nuovo blocco costante. Di fatto:
  + Ogni qual volta si invii una transazione, la rete richiede 10 minuti circa per confermarla e può gestire circa sette transazioni al secondo. Entro questi dieci minuti si forma un nuovo blocco
  + Ogni blocco detiene diverse transazioni, che devono essere validate in modo indipendente
  + I miners, con più alto rendimento computazionale, prendono una transazione alla volta e risolvono un algoritmo crittografico, altrimenti noto come proof of work
  + La transazione, a questo punto viene validata e pubblicata sulla blockchain pubblica affinché tutti possano vederla. I miners più veloci, i primi che risolvono l’algoritmo, vengono ricompensati in criptovalute, quella di utilizzo della blockchain su cui operano. Di fatto, quindi, non è una cosa equa, in quanto si consuma molta energia ed è pesantemente sbilanciata a favore di chi abbia le attrezzature migliori
  + Di fatto, il meccanismo di validazione è molto lento. Le transazioni sono memorizzate in una mempool in attesa di convalida, con tempi medi di conferma tra il 1° gennaio e il 9 febbraio 2023 che vanno da sette a 91 minuti (la conferma è quando la transazione viene confermata).
    - Una *mempool* (o txpool) è l’area in cui si trovano le transazioni che non sono state aggiunte ad un blocco e sono tuttora non state confermate. Questa area di attesa o zona cuscinetto è necessaria perché le transazioni non vengono aggiunte immediatamente alla Blockchain.
    - I nodi eseguono anche una serie di controlli di validità su queste transazioni. Questi controlli includono la verifica che i fondi siano ancora disponibili, che l'output non superi l'input, che la firma sia valida, ecc. La transazione viene rifiutata se non supera uno di questi test.
    - Si noti che il Mempool non può essere considerato come un riferimento principale condiviso universalmente da tutti i nodi. Ogni nodo configura le proprie regole per il suo mempool. Inoltre, un nodo può essere il primo a ricevere una transazione, ma potrebbe non averla propagata al resto della rete.
  + Tuttavia, l’algoritmo di consenso PoW Proof of Work consente ai miners di convalidare un nuovo blocco e aggiungerlo alla blockchain solo se gli altri nodi del network concordano con la soluzione fornita dal miner, ripetendo l’operazione risolta.

La Proof-of-Work è utilizzata in molte criptovalute.

* L'applicazione più famosa di PoW è Bitcoin. È stato Bitcoin a gettare le basi per questo tipo di consenso. Il puzzle è Hashcash. Questo algoritmo consente di modificare la complessità di un puzzle in base alla potenza totale della rete. Il tempo medio di formazione dei blocchi è di 10 minuti. Le criptovalute basate su Bitcoin, come Litecoin, hanno un sistema simile.
* Un altro grande progetto con PoW è Ethereum. Dato che quasi tre progetti su quattro sono implementati sulla piattaforma Ethereum, si può affermare che la maggior parte delle applicazioni Blockchain utilizza il modello di consenso PoW.

I vantaggi principali sono la difesa dagli attacchi DoS e il basso impatto della posta in gioco sulle possibilità di estrazione.

* Difesa dagli attacchi DoS. Il PoW impone alcuni limiti alle azioni nella rete. Per essere eseguite necessitano di un grande impegno. Un attacco efficiente richiede molta potenza di calcolo e molto tempo per eseguire i calcoli. Pertanto, l'attacco è possibile ma piuttosto inutile, poiché i costi sono troppo elevati.
* Possibilità di estrazione. Non importa quanto denaro avete nel vostro portafoglio. Ciò che conta è avere una grande potenza di calcolo per risolvere i puzzle e formare nuovi blocchi. In questo modo, i detentori di ingenti somme di denaro non sono incaricati di prendere decisioni per l'intera rete.

#### Puzzle matematici della PoW

Ce ne sono molti, ad esempio:

* *funzione hash*, ovvero come trovare l'input conoscendo l'output.
  + Queste funzioni non sono reversibili, dato che sarebbe impossibile ottenere il dato originale di input dall’output della funzione di hash. Le funzioni sono infatti designate per essere a via unica. La computazione è molto più semplice in avanti che all’indietro.
* *fattorizzazione di interi* (prime/integer factorization), in altre parole, come presentare un numero come moltiplicazione di altri due numeri.
  + I numeri primi sono tutti i numeri divisibili solo per sé stessi e per 1

Una pratica comune è quella di utilizzare semi-primi molto grandi (cioè il risultato della moltiplicazione di due numeri primi) come numero di sicurezza della crittografia. Per decifrarlo, si dovrebbe trovare la fattorizzazione prima del numero semiprimo grande, cioè due o più \*\*numeri primi\*\* che moltiplicati insieme danno come risultato il numero originale.

Normalmente è facile generare un numero primo grande, ma fattorizzarlo richiede un processo di trial-and-error non indifferente

* + Algoritmi crittografici come RSA, che si basa sulla moltiplicazione di due numeri primi (P e Q) per ottenere il valore del modulo (N). Ci possono essere attacchi al valore del modulo; normalmente, si cercano una serie di numeri primi che permettano di ottenere , ottenuta come . Ci sono vari modi per realizzare questo attacco, ad esempio tramite:
    - La differenza di quadrati
    - Residui quadratici mod n
    - Fattorizzazione di curve ellittiche
    - Il metodo di Pollard
* protocollo puzzle di visita guidata (guided tour puzzle protocol). Se il server sospetta un attacco DoS, richiede il calcolo delle funzioni hash per alcuni nodi in un ordine definito. In questo caso, si tratta di un problema di "come trovare una catena di valori di funzioni hash".
  + Ecco come funziona il protocollo dei puzzle delle visite guidate:
    - Il mittente genera una serie di enigmi, ciascuno con una soluzione unica. I puzzle sono progettati in modo da essere computazionalmente difficili da risolvere, ma facili da verificare.
    - Il mittente invia l'insieme dei puzzle al destinatario.
    - Il destinatario seleziona uno dei puzzle e lo risolve. La soluzione del rompicapo serve come biglietto o chiave per accedere al canale di comunicazione.
    - Il ricevitore invia la soluzione al mittente.
    - Il mittente verifica la soluzione e concede l'accesso al canale di comunicazione.
    - Il mittente e il destinatario possono ora comunicare in modo sicuro utilizzando tecniche di crittografia e decrittografia.
    - Il protocollo del puzzle di visita guidata è progettato per impedire agli aggressori di intercettare e leggere la comunicazione tra il mittente e il destinatario. Anche se un aggressore intercettasse l'insieme dei puzzle, sarebbe computazionalmente impossibile risolvere tutti i puzzle per trovare la soluzione corretta. Pertanto, il protocollo garantisce che solo il destinatario previsto possa accedere al canale di comunicazione.

#### Tipi di PoW

##### Proof of Meaningful Work (PoMW)

Il Proof of Meaningful Work (PoMW) è un algoritmo di consenso che cerca di combinare i vantaggi di Proof of Work (PoW) e Proof of Stake (PoS), cercando di mitigare i loro svantaggi. In PoMW, i partecipanti alla rete devono svolgere un lavoro utile per risolvere un problema specifico (progetti scientifici, es. ricerca medica per cure, ricerca chimica, ricerca astrofisica, etc.) in modo da dimostrare di aver svolto "un lavoro significativo" che richiede una quantità significativa di potenza di elaborazione.

* Ciò significa che i partecipanti alla rete che svolgono il lavoro ricevono una ricompensa per la loro fatica, in modo simile a PoW, ma evitando di utilizzare una quantità enorme di energia come avviene in PoW.
* Consuma meno rispetto a PoW ed è più equo rispetto a PoS e si ha un riconoscimento del lavoro svolto dai partecipanti
* Tuttavia, vi è la possibilità che partecipanti alla rete possano cercare di "imbrogliare" il sistema svolgendo il lavoro in modo inefficiente o in modo tale da ottenere il maggior numero possibile di ricompense.

##### Hybrid Proof of Work (HPoW)

L’Hybrid Proof of Work (HPoW) è un'altra variante dell'algoritmo PoW che cerca di ridurre i suoi svantaggi. In HPoW, gli utenti partecipano alla creazione di nuovi blocchi condividendo potenza di elaborazione, come avviene in PoW. Tuttavia, a differenza di PoW tradizionale, HPoW richiede anche la partecipazione dei possessori di token, come avviene in PoS, per validare i blocchi. Questo significa che l'algoritmo è meno suscettibile alle attività di mining centralizzate che possono verificarsi in PoW tradizionale, senza sacrificare la sicurezza del network come avviene in PoS.

* HPoW elimina l'incentivo al profitto per i minatori perché la ricompensa per l'estrazione è così bassa.
* Un singolo minatore non può vincere più di una volta ogni 30 minuti.
* Il saldo dell'indirizzo di ricompensa del minatore deve essere maggiore o uguale a una quantità minima di Lynx richiesta e fluttuante per vincere un blocco.
* Utilizzando una selezione casuale, i minatori più veloci non hanno sempre la garanzia di vincere la ricompensa del blocco.
* Risulta più sicuro rispetto a PoS e consuma meno rispetto a PoW, ma può essere vulnerabile alla concentrazione di potere da parte dei possessori di token.
* Di fatto, viene usato dalla criptovaluta Lynx

##### Delayed Proof of Work (dPoW)

La Delayed Proof of Work (dPoW) è un metodo di consenso ibrido che consente a una blockchain di sfruttare la sicurezza fornita dalla potenza di hashing di una blockchain secondaria. Ciò avviene attraverso un gruppo di nodi notarili (notary nodes) che aggiungono dati dalla prima blockchain alla seconda, il che richiederebbe la compromissione di entrambe le blockchain per minare la sicurezza della prima.

* Viene usato nelle blockchain Komodo. Il processo di inserimento dei backup delle transazioni di Komodo in un PoW sicuro è la "notarizzazione". La notarizzazione viene eseguita dai nodi notarili eletti. All'incirca ogni dieci minuti, i nodi notarili eseguono uno speciale hash dei blocchi estratto sulla blockchain di Komodo e prendono nota dell'"altezza" complessiva della blockchain di Komodo.
* I nodi notarili elaborano questo blocco specifico in modo che le loro firme siano incluse crittograficamente nel contenuto dei dati autenticati. In questo modo, a basso costo è possibile minare e ritrovare le giuste informazioni di riferimento.

### Proof of Stake

La Proof of Stake (PoS) è un algoritmo di consenso utilizzato in alcune criptovalute per raggiungere un accordo sulla validità delle transazioni e delle operazioni nella blockchain. La PoS funziona in modo diverso dalla Proof of Work (PoW) utilizzata in criptovalute come Bitcoin.

* Invece di affidarsi alla potenza di calcolo per risolvere complessi problemi matematici, come avviene nella PoW, la PoS si basa sulla detenzione di una certa quantità di criptovaluta come garanzia per la validazione delle transazioni. In pratica, i nodi della rete bloccano una certa quantità di criptovaluta come "punteggio" per dimostrare che hanno un interesse nella corretta validazione delle transazioni. Questo punteggio viene utilizzato come base per la selezione del nodo che convalida la transazione successiva.
* A differenza del PoW, non ci sono miner coinvolti nel processo. Al loro posto, i partecipanti alla rete che vogliono essere coinvolti nella verifica della validità delle transazioni e nella creazione di blocchi nella rete devono detenere una certa quota nella rete, per esempio mettendo una certa quantità di moneta della rete in un portafoglio collegato alla sua blockchain. Questo processo è noto come "placing a stake" o "staking”, che può essere tradotto come il fatto di mettere i propri interessi in gioco. Un creatore di blocchi in un sistema PoS può solo creare blocchi proporzionati alla propria partecipazione alla rete.
* Pertanto, le reti PoS sono basate su algoritmi deterministici, il che significa che i validatori dei blocchi sono eletti a seconda della natura della posta in gioco. Per esempio, selezionare il saldo del conto come unico criterio su cui viene definito il prossimo blocco valido in una blockchain potrebbe potenzialmente portare a una centralizzazione indesiderata. Ciò vorrebbe dire che i membri ricchi di una rete godrebbero di grandi vantaggi.

A livello di vantaggi:

* La PoS ha il vantaggio di essere meno intensiva in termini di energia rispetto alla PoW e richiede meno potenza di calcolo per mantenere la sicurezza della rete. Inoltre, la PoS è meno vulnerabile ad attacchi del 51 percento rispetto alla PoW, poiché gli attaccanti dovrebbero possedere la maggioranza delle criptovalute della rete, e non la maggioranza della potenza di calcolo.
* Migliora il decentramento e democratizza l’accesso alla rete (permettendo a tutti di accedere), è più scalabile e anche sicura, dato che la quantità di criptovaluta viene usata come garanzia per partecipare al processo di consenso e riducendo il rischio di comportamenti fraudolenti.

A livello di svantaggi:

* I produttori di blocchi di alcune monete possono esercitare una quantità incredibile di potere se il numero di produttori di blocchi in una rete è basso, e se quindi sono loro a poter convalidare tutte le transazioni. Tuttavia, il potere di un produttore può essere automaticamente revocato ogni volta che fa qualcosa contro gli interessi della rete. Se, per esempio, un produttore della moneta EOS non riesce a lavorare su nessun blocco per 24 ore, un produttore di riserva prende rapidamente il suo posto.
* La seconda grande debolezza è che un certo numero di sistemi PoS favorisce gli utenti ricchi: quante più monete si hanno in gioco nella rete, tanto più si estende il proprio diritto di voto. Reti come Cardano hanno già affrontato questo problema implementando la selezione randomizzata dei produttori di blocchi. In questo caso, gli utenti più ricchi hanno ancora una migliore possibilità di essere produttori di blocchi, ma è invece diminuita l'influenza esterna delle "cripto balene", ovvero dei partecipanti che detengono molte più monete di una particolare rete rispetto all'utente medio.
* Infine, una rete Proof of Stake ha un problema noto come "nessuna posta in gioco". In una rete PoW c’è l'eventualità che due miner producano un blocco quasi simultaneamente a causa di un ritardo temporale. Questo si traduce in una confusione temporanea nella rete e i nodi devono raggiungere il consenso su quale blocco sia valido. Di conseguenza, i miner devono scegliere su quale versione della blockchain spendere le loro risorse, bypassando le altre opportunità.

Molti hanno provato a risolvere questi problemi (di seguito alcune criptovalute)

* Peercoin ha messo in atto dei checkpoint firmati dalla chiave privata del programmatore e poi veicolati centralmente dal sistema. Ciascun checkpoint è un punto fermo e non può essere riorganizzato né esso stesso né tantomeno la parte “approvata” da esso. Lo sviluppatore, quindi, è l’autorità centrale che pesa estremamente sulla blockchain.
* Nxt consente di rilavorare un numero definito di blocchi: 720. Possibile conseguenza: il client potrebbe quindi “seguire” una ramificazione di 721 blocchi non accertandosi che sia la blockchain più lunga, prevenendo il consenso.
* Ethereum – Slasher: gli utenti possono “punire” un “imbroglione” che forgia sui forks (cambiamenti al protocollo non retrocompatibili) della blockchain. L’utente avrà dovuto firmare due volte per creare una biforcazione e sarà punito se avrà creato una biforcazione senza avere nessuna posta in gioco.

#### Delegated Proof of Stake

La Delegated Proof of Stake (DPoS) è una variante dell'algoritmo di consenso Proof of Stake (PoS) utilizzato in alcune criptovalute, come ad esempio EOS.

* A differenza della PoS tradizionale, dove tutti i nodi della rete possono partecipare al processo di validazione delle transazioni, nella DPoS vengono scelti un numero limitato di nodi (solitamente tra 20 e 100) come "delegati" che hanno il compito di convalidare le transazioni e aggiornare la blockchain. Questi delegati sono selezionati in base al numero di voti che ricevono dai detentori di criptovalute della rete.
* I voti sono proporzionali alla quantità di criptovaluta detenuta dai votanti e i delegati vengono scelti in ordine decrescente di voti ricevuti. I delegati eletti sono quindi responsabili della validazione delle transazioni e vengono ricompensati con le commissioni di transazione per il lavoro svolto.
* La DPoS ha alcuni vantaggi rispetto alla PoS tradizionale e alla Proof of Work (PoW). In primo luogo, la DPoS è più veloce e scalabile rispetto alla PoW, in quanto la selezione dei delegati è più efficiente rispetto al processo di mining. Inoltre, la DPoS riduce la centralizzazione, poiché solo un piccolo numero di nodi è coinvolto nel processo di convalida delle transazioni.
* Tuttavia, la DPoS ha anche alcune criticità. Innanzitutto, la selezione dei delegati è basata sulla quantità di criptovaluta detenuta, il che potrebbe creare disuguaglianze tra i detentori di criptovalute e potenzialmente portare alla creazione di oligarchie. Inoltre, i delegati potrebbero subire pressioni esterne per convalidare transazioni non valide o dannose.

#### Leased Proof of Stake

La Leased Proof of Stake è un meccanismo di consenso utilizzato in particolare dalla blockchain Waves, in base al quale gli utenti affittano token crittografici a un nodo che intende agire come produttore di blocchi della rete. Più token un nodo ha in palio, più è probabile che venga scelto per generare il blocco successivo e ricevere la ricompensa corrispondente, e i proprietari dei token hanno il diritto di cancellare il loro contratto di locazione in qualsiasi momento.

* Di conseguenza, i piccoli detentori di token che altrimenti non avrebbero diritto a partecipare al processo di creazione dei blocchi in un sistema PoS tradizionale possono unire le loro risorse e aumentare le possibilità di ricevere una quota delle commissioni di transazione della rete. Gli utenti possono fare acquisti per trovare il nodo che meglio si adatta alla loro strategia di investimento, poiché alcuni nodi possono distribuire ricompense maggiori.
* I protocolli LPoS si applicano meglio alle reti che hanno requisiti tecnici elevati per gestire un nodo completo in grado di convalidare le transazioni sulla catena. Questo meccanismo di consenso premia i nodi più performanti, incentivando gli utenti più piccoli a sostenere i validatori più efficienti in modo trasparente e sostenibile. In termini di effetto netto, questo meccanismo di consenso è abbastanza simile a DPoS. Tuttavia, mentre i validatori di DPoS sono selezionati dai voti ponderati degli altri partecipanti alla rete, i possessori di token all'interno di una rete LPoS possono prendere in prestito e prestare token direttamente per partecipare al processo di produzione dei blocchi.

#### Pure Proof of Stake (PPoS)

La Pure Proof of Stake (PPoS) è una forma altamente democratizzata di PoS utilizzata da Algorand, un progetto di blockchain pubblica incentrato sullo sviluppo di applicazioni decentralizzate (dApp) di facile utilizzo. A differenza di molte altre forme di PoS, i meccanismi di consenso PPoS non dispongono di un meccanismo di sanzione incorporato per prevenire l'attività di nodi maligni o potenziali difetti di sicurezza come la duplicazione delle convalide dei blocchi. Al contrario, PPoS offre requisiti minimi di puntata per la partecipazione e la sicurezza della rete, aprendo le porte a tutti gli utenti interessati. Questo crea un sistema in cui sarebbe finanziariamente autodistruttivo per gli attori disonesti interrompere o dirottare la rete.

* Sulla rete Algorand è necessaria una sola moneta ALGO per partecipare al processo di staking della rete. Ogni utente della rete Algorand può essere selezionato in modo casuale e segreto per proporre nuovi blocchi e votare le proposte, e la probabilità che un determinato utente venga scelto - insieme al peso delle sue proposte e dei suoi voti - è proporzionale alla sua partecipazione. Un sistema PPoS funzionerà normalmente finché due terzi dei nodi della rete agiranno onestamente.

#### Liquid Proof of Stake (LPoS)

Liquid Proof of Stake (LPoS) consente ai titolari di token di prestare i propri diritti di convalida ad altri utenti senza rinunciare alla proprietà del token. Sebbene possa sembrare simile a DPoS, i titolari di token in una rete LPoS scelgono autonomamente se delegare i propri diritti di convalida tokenizzati ad altri utenti o mettere in gioco i propri token. Inoltre, il numero di nodi validatori attivi in LPoS è dinamico, a differenza del numero fisso di validatori di DPoS.

In LPoS, un validatore è chiamato "baker" o "endorser". A differenza del DPoS, ogni utente può diventare un validatore se ha abbastanza monete. Se non ne ha, può scegliere di delegare. L'idea è di diluire ancora di più l'attività e di aumentare l'inclusione. L'attenzione è rivolta più alla liquidità della governance che alla scalabilità della rete. I due ruoli dei delegati sono semplici:

* Creatori: creano blocchi
* Endorser: concordano con i blocchi

Di conseguenza, gli utenti di una rete LPoS hanno un alto grado di flessibilità in termini di partecipazione alla rete. Ad esempio, i grandi possessori di token possono diventare validatori di blocchi puntando i propri fondi senza bisogno di un'approvazione esterna. I piccoli detentori che non hanno le risorse per convalidare da soli i blocchi possono sostenere i detentori più grandi o riunirsi per formare coalizioni efficaci. Allo stesso tempo, poiché i diritti di convalida LPoS sono così fluidi e possono essere facilmente riorganizzati, questa configurazione aiuta a mitigare il rischio che una coalizione di maggioranza prenda il controllo dell'intera rete.

#### Bonded Proof of Stake (BPoS)

Il Bonded Proof of Stake (BPoS) è molto simile all'LPoS: la delega è facoltativa, non vincolante e i detentori di token godono del diritto di voto nelle modifiche del protocollo. Tuttavia, c'è un motivo per cui si chiama BPoS: in caso di errore di sicurezza o di liveness, una parte della partecipazione dei validatori e dei delegatori sarà tagliata. In LPoS, solo il validatore è a rischio di taglio, mentre l'unico rischio dei delegatori è quello di perdere alcune ricompense/interessi nel caso in cui il suo validatore sia disonesto o non efficiente.

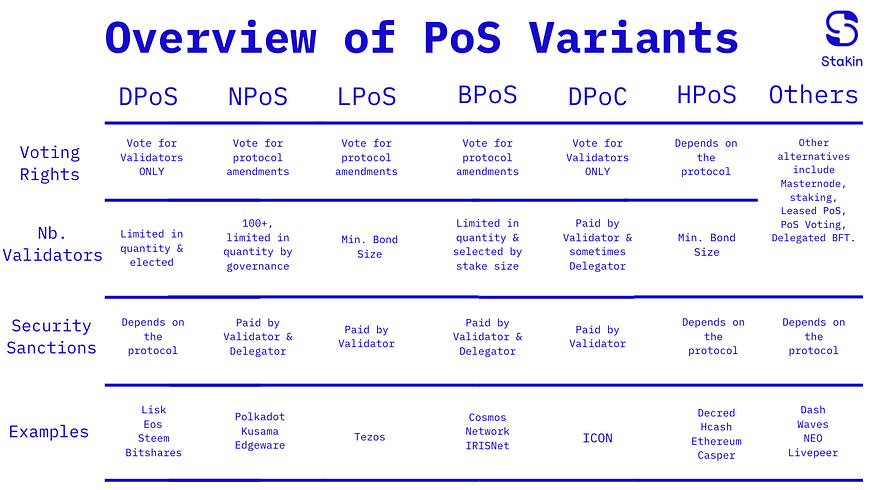
* Questo meccanismo BPoS ha il vantaggio di fornire una soluzione chiara al problema dei rapporti di puntata (simili ai requisiti di capitale) che alcuni validatori dei protocolli LPoS devono mantenere se non vogliono diventare troppo delegati e deludere alcuni dei loro delegatori. Se da un lato risolve questo problema, dall'altro significa che i deleganti devono condurre un approfondimento di informazioni e dati supplementare prima di delegare e rimanere attivi nel verificare le prestazioni del loro validatore.

#### Nominated Proof of Stake (NPoS)

L'ecosistema Polkadot ha introdotto la Nominated Proof-of-Stake (NPoS). In questo sistema, i validatori vengono selezionati automaticamente un paio di volte al giorno. Questi validatori devono eseguire operazioni costose, come garantire un'elevata reattività delle comunicazioni, costruire una reputazione di affidabilità a lungo termine e mettere in gioco il proprio token nativo. Questo viene fatto come garanzia di un buon comportamento e la loro quota viene tagliata ogni volta che si discostano dal loro protocollo.

* In NPoS, i delegatori sono chiamati nominatori. Un nominatore designa i validatori tra un elenco di candidati e blocca una quantità di gettoni di palo con cui sostenerli. In NPoS, il numero di validatori è limitato dalla governance, e questi sono selezionati in base alla loro partecipazione totale nella rete. A differenza dei validatori, un numero illimitato di soggetti può essere nominato. I nominatori sono incentivati a essere sempre alla ricerca di nuovi candidati solidi. Grazie a questi due ruoli, NPoS consente a tutti i titolari di token di partecipare costantemente alla rete. Pertanto, mantiene un elevato livello di sicurezza, pur mantenendo il numero di validatori limitato.

Una piccola overview di tutte le varianti PoS:



Riferimento: <https://medium.com/stakin/proof-of-stake-guide-dpos-vs-lpos-vs-bpos-vs-hybrid-1393a33e849c>

### Altre tipologie di consenso

#### Proof of Importance (PoI)

Nella Proof of Importance la partecipazione alla convalida della rete è basata sulla quantità di criptovaluta detenuta e sulla sua attività sulla rete. In pratica, i nodi con un saldo maggiore e un'attività più attiva sulla rete hanno maggiori probabilità di essere selezionati per la convalida delle transazioni.

* In altre parole, PoI valuta la "importanza" di un nodo sulla base del suo coinvolgimento nella rete, incentivando la partecipazione attiva degli utenti. PoI ha il vantaggio di essere meno vulnerabile agli attacchi del 51 percento rispetto alla PoW e alla PoS, poiché gli attaccanti dovrebbero possedere la maggioranza delle criptovalute della rete e un'attività molto elevata sulla rete per poter manipolare il processo di convalida.
* I criteri esatti di attribuzione dei punteggi utilizzati nei PoI variano, anche se molti di questi meccanismi di consenso prendono in prestito caratteristiche dagli algoritmi utilizzati nel clustering della rete e nel ranking delle pagine. Tra i fattori comuni vi sono il numero di trasferimenti a cui un nodo ha partecipato in un determinato periodo di tempo e il grado di interconnessione tra i diversi nodi attraverso i cluster di attività.
* Il PoI aiuta a mitigare il rischio di un eccesso di ricchezza concentrata, in quanto i maggiori detentori di token non esercitano un potere assoluto sulla rete. Poiché il punteggio di importanza di ciascun nodo è dinamico e basato sull'attività della rete, questo meccanismo di consenso scoraggia le biforcazioni della blockchain: Gli utenti dovrebbero spendere risorse per rimanere attivi su entrambe le reti biforcate al fine di mantenere il proprio punteggio.

#### Proof of Burn (PoB)

Il proof-of-burn (PoB) è un meccanismo di consenso della blockchain con un consumo energetico minimo, rispetto al proof-of-work (PoW). Le piattaforme decentralizzate che utilizzano il metodo PoB assicurano che i minatori raggiungano il consenso bruciando le monete. La masterizzazione è il processo di eliminazione permanente delle criptovalute dalla circolazione. Sebbene questa pratica riduca l'inflazione, le blockchain basate su PoB la utilizzano per convalidare le transazioni.

* In parole povere, i minatori PoB iniziano a bruciare monete virtuali come modo per dimostrare il loro coinvolgimento nella rete ed essere autorizzati a minare. Il numero di monete bruciate da un minatore dimostra la sua potenza mineraria virtuale. Pertanto, maggiore è il numero di monete, maggiore è la potenza e viceversa. Si noti che, proprio come nei sistemi PoW, una maggiore potenza di mining migliora la velocità di trovare nuovi blocchi. Di conseguenza, il minatore ottiene maggiori ricompense.
* Per bruciare le monete, i minatori le inviano a un indirizzo non spendibile e verificabile. Questo processo non consuma molte risorse (a parte le monete bruciate) e garantisce che la rete rimanga attiva e agile. A seconda dell'implementazione, i minatori possono bruciare la valuta nativa o la valuta di una catena alternativa, come Bitcoin. In cambio, ricevono una ricompensa in token della valuta nativa della blockchain. Nelle reti proof-of-burn (PoB), il processo di masterizzazione delle monete prevede l'invio delle stesse a un "indirizzo di eater". Questo indirizzo è pubblicamente verificabile ma inaccessibile. Si noti che gli indirizzi degli eater sono randomizzati e non hanno chiavi private.

#### Proof of Authority (PoA)

La Proof-Of-Authority (PoA) è un metodo di consenso che conferisce a un numero ristretto e designato di attori della blockchain il potere di convalidare le transazioni o le interazioni con la rete e di aggiornare il suo registro più o meno distribuito.

* Funziona come segue: in base allo schema scelto, una o più macchine validatrici sono responsabili della generazione di ogni nuovo blocco di transazioni che verrà inserito nella Blockchain. Il nuovo blocco può essere accettato direttamente senza verifica, o con voto unanime dei generatori di blocchi, o semplicemente a maggioranza, a seconda della configurazione scelta per la Blockchain. Questi nodi hanno il compito di validare le transazioni, creare nuovi blocchi e aggiornare il registro. Nel PoA, i nodi autorizzati sono motivati a fornire un servizio affidabile e sicuro, poiché il loro status di autorità dipende dalle loro prestazioni.
* Di fatto, mentre risulta molto veloce e scalabile dato che non richiede risoluzione di problemi complessi o sacrificio di cripto, risulta un sistema di consenso centralizzati, in quanto i nodi autorizzati sono selezionati e designati da un’autorità centrale, limitando la decentralizzazione. Inoltre, anche la sicurezza può essere ridotta, in quanto vulnerabile agli attacchi di coordinamento, dato che un attaccante potrebbe cercare di infiltrarsi nella rete diventando uno dei nodi autorizzati.

#### Tolleranza ai guasti bizantina (BFT)

Il problema che risolve è il Byzantine Generals Problem, problema fondamentale nei sistemi di calcolo distribuito, dove un gruppo di nodi o computer deve lavorare insieme per raggiungere un obiettivo comune. In questo caso, questo problema riguarda il raggiungimento di consenso comune da parte di tutti i nodi della rete.



Riferimento: <https://river.com/learn/what-is-the-byzantine-generals-problem/>

* Esso prende il nome da un ipotetico scenario in cui un gruppo di generali bizantini deve coordinare i propri piani di attacco per conquistare una città. Il problema è che alcuni dei generali possono essere dei traditori che cercano di interrompere la comunicazione tra gli altri generali, rendendo impossibile raggiungere un consenso su quando e come attaccare la città.
  + Il problema sorge quando alcuni dei nodi della rete sono difettosi o compromessi e possono inviare messaggi contraddittori o cercare di sabotare il funzionamento della rete. Questo può portare a un'interruzione della capacità della rete di raggiungere un consenso o di prendere decisioni, il che può essere disastroso in sistemi mission-critical come le transazioni finanziarie, la gestione della catena di approvvigionamento o le operazioni militari.
  + La tecnologia blockchain risolve il problema dell'accordo generale bizantino utilizzando un protocollo di consenso distribuito che consente alla rete di raggiungere un consenso anche se alcuni nodi sono difettosi o maligni.

La fault tolerance, per garantire il successo della squadra dei generali, richiede un algoritmo che rispetti le seguenti condizioni:

* Tutti i generali delle truppe devono essere d'accordo sulla prossima azione del piano.
* I generali devono essere affidabili e fedeli al sistema.
* I generali non devono essere influenzati e diventare traditori della rete.
* Devono seguire l'algoritmo del sistema.
* Il gruppo di generali deve raggiungere un consenso o una decisione, indipendentemente dalle azioni dei traditori.
* Il sistema o la rete non devono portare a un attacco del 51% in nessun punto dell'azione.

La tolleranza ai guasti bizantina (BFT) è un approccio di consenso che impedisce al sistema di entrare nel problema dei generali bizantini. Significa anche che il sistema dovrebbe rimanere intatto anche se uno dei nodi (o un generale) si guasta. Inoltre, BFT mira a ridurre l'effetto di nodi (o generali) bizantini dannosi sulla rete.

##### Practical Byzantine Fault Tolerance (pBFT)

Nel 1999, Barbara Liskov e Miguel Castro hanno introdotto un algoritmo di consenso Practical Byzantine Fault Tolerance (pBFT), nel tentativo di superare i problemi di bizantinismo. Il loro obiettivo è garantire una replica pratica della macchina a stati bizantini per tollerare nodi maligni o bizantini.

Il pBFT segue un approccio asincrono. Gli aspetti essenziali dell'algoritmo di consenso pBFT sono i seguenti:

* Tutti i nodi sono riuniti in una sequenza.
* Un nodo della rete funge da nodo leader e gli altri sono nodi di riserva.
* Il nodo primario o leader serve la richiesta del cliente. Funziona come moderatore tra il client e i nodi di backup.
* Tutti i nodi sono in grado di comunicare con altri nodi per verificare i nodi onesti e lo fanno consistentemente e senza soluzione di continuità.
* I nodi onesti devono essere in grado di raggiungere un consenso per il prossimo cambiamento globale della rete basato sulla regola della maggioranza.
* Identifica la fonte del messaggio per assicurarsi che sia stato inviato dal mittente corretto.
* Assicura che il messaggio non sia stato modificato o corrotto nel mezzo.

Il PBT dipende fortemente dalla condizione che il numero massimo di nodi maligni o bizantini deve superare un terzo di tutti i nodi della rete. Pertanto, la sicurezza della rete dipende direttamente dal numero totale di nodi onesti. In breve, un sistema pBFT può gestire nodi difettosi o bizantini quando ci sono nodi totali sulla rete.

Di seguito viene descritto il processo dell'algoritmo di consenso pBFT:

* Un client invia una richiesta al nodo leader.
* Il nodo leader invia la richiesta a tutti i nodi di backup.
* Tutti i nodi lavorano sulla richiesta e inviano una risposta al client.
* Il client attende risposte da tutti i nodi con lo stesso risultato.   
  Qui,

Il meccanismo pBFT consiste in 3 fasi:

* Fase di pre-preparazione: Il nodo leader invia un messaggio pre-preparato a ciascun nodo di backup.
* Preparazione: Dopo aver ricevuto il messaggio pre-preparato dal leader, i nodi di backup inviano il messaggio preparato come risposta a tutti gli altri nodi, incluso il leader. Un nodo è considerato preparato solo se ha ricevuto il messaggio pre-preparato dal leader e ha visto un numero di messaggi preparati da altri nodi.
* Impegno: Se i nodi sono preparati, inviano un messaggio di commit. Se un nodo riceve messaggi di commit, esegue la richiesta del client.

Di seguito un’immagine che ne spiega il funzionamento:  


Riferimento: <https://www.naukri.com/learning/articles/byzantine-fault-tolerance-in-blockchain/>

L'intero processo di verifica in un sistema distribuito utilizza il concetto di firma digitale (usando la crittografia a chiave pubblica come spiegato prima). La validità del messaggio e del mittente è garantita da numeri di sequenza e metadati.

Il modello BFT pratico per il consenso della blockchain è stato adattato alle applicazioni pratiche. L'articolo accademico originale alla base del BFT pratico ha anche delineato le particolari carenze, mostrando al contempo i miglioramenti chiave per l'implementazione dell'algoritmo nei sistemi reali. La Practical Byzantine Fault Tolerance può contribuire a garantire i seguenti vantaggi.

1. Flessibilità e velocità delle transazioni

Prima di tutto, può fornire la garanzia della finalizzazione delle transazioni senza alcun requisito di conferma come nella Proof-of-Work. È possibile riscontrare una notevole differenza tra il modello PoW utilizzato da Bitcoin e il BFT pratico. Quando i nodi in un modello BFT pratico sono d'accordo su un blocco interessato, il blocco è considerato definitivo. La definitività si basa sul fatto che tutti i nodi onesti sono d'accordo sullo stato del sistema in un determinato momento. La comunicazione tra i nodi onesti aiuta a garantire un accordo credibile sullo stato del sistema.

1. Basso consumo energetico

Il prossimo importante vantaggio di questo algoritmo di tolleranza ai guasti bizantino rispetto al consenso PoW è il ridotto consumo di energia. Il modello Proof-of-Work utilizzato in Bitcoin implica la necessità di un round PoW per ogni blocco. Gradualmente, i minatori della rete Bitcoin aumentano il consumo di energia elettrica, che può superare il consumo annuale di elettricità di piccole nazioni.

* Il BFT pratico non comporta sforzi computazionali intensivi, il che porta a una profonda riduzione del consumo di energia elettrica. Con il BFT pratico, i minatori non devono risolvere algoritmi di hashing PoW per ogni blocco, richiedendo risorse computazionali intensive.
* Tuttavia, il BFT pratico presenta anche alcuni inconvenienti. Ad esempio, il modello BFT pratico è applicabile solo nella sua forma classica. Pertanto, è limitato a gruppi di consenso di piccole dimensioni per evitare i volumi ingombranti di comunicazione necessari tra i nodi (poco scalabile). Inoltre, l'uso delle firme digitali e dei codici di autenticazione del metodo per l'autenticazione dei messaggi può presentare problemi di inefficienza. Di fatto, è suscettibile agli attacchi Sybil di impersonificazione di nodi

##### Delegated Byzantine Fault Tolerance (dBFT)

Delegated Byzantine Fault Tolerance o dBFT è un meccanismo di consenso reso popolare da una criptovaluta chiamata NEO. dBFT funziona essenzialmente in modo simile al sistema di governo di un Paese, con i suoi cittadini, i suoi delegati e i suoi relatori per garantire che il Paese (la rete) sia funzionale. Il metodo è più vicino al PoS che al PoW, in quanto utilizza un sistema di votazione per scegliere i delegati e gli oratori.

* Il dBFT è una variante del classico algoritmo di consenso Byzantine Fault Tolerance (BFT), ma con la caratteristica aggiuntiva che prevede la presenza di nodi rappresentativi (o "delegati") che agiscono come intermediari tra la rete e i partecipanti. Questo si nota essere simile a DPoS (Delegate Proof of Stake)
* A prima vista, il meccanismo di consenso dBFT è simile alla Proof-of-Stake delegata. Utilizzando un processo di votazione, i possessori di token NEO hanno il diritto di votare per i delegati. Questo indipendentemente dalla quantità di valuta in loro possesso.
  + Chiunque può diventare un delegato, purché soddisfi i requisiti richiesti. Ciò significa una connessione internet affidabile, l'attrezzatura giusta, un'identità convalidata e 1.000 GAS (token usato all’interno della rete NEO). Il GAS è la ricompensa che gli utenti ricevono per la loro attività sulla rete. Tra i delegati viene scelto a caso un oratore.
  + L'oratore costruisce un nuovo blocco dalle transazioni che attende di essere convalidato. Quindi, l'oratore invia la proposta ai delegati eletti. Questi ultimi devono tenere traccia di tutte le transazioni e registrarle sulla rete.
  + I delegati sono liberi di condividere e confrontare la proposta ricevuta per verificare l'accuratezza dei dati e l'onestà dell'oratore. Il blocco viene aggiunto alla blockchain se più di due terzi dei delegati raggiungono un consenso e lo convalidano. La votazione nella rete NEO è un processo che avviene in tempo reale.
  + Ciò significa che il titolare di un token NEO può sostenere uno specifico "bookkeeper" attraverso una votazione. Il gruppo selezionato di bookkeeper utilizza quindi il meccanismo di tolleranza ai guasti bizantina per raggiungere un consenso e generare altri blocchi.
  + Uno dei punti di forza dell'utilizzo del meccanismo dBFT consiste nell'assoluta definitività. Dopo la conferma finale, un blocco non può essere biforcato, quindi la transazione non può essere revocata o annullata. Si tratta però di un'arma a doppio taglio.
    - La definitività è in qualche modo garantita dal fatto che NEO non è una rete completamente decentralizzata. Nonostante gli sforzi di NEO per andare in questa direzione, attualmente ci sono solo sette nodi e alcuni delegati che operano sulla blockchain. La maggior parte di questi è collegata al consiglio di NEO.
  + Poiché tutti i delegati possono verificare la proposta di blocco, è facile capire se i dati inviati dall'oratore sono validi o meno. Quindi, se l'oratore è disonesto e invia proposte non valide a due terzi dei delegati, i blocchi non corrisponderanno e i proprietari dei nodi non li convalideranno. Il consenso viene raggiunto con i due terzi dei voti e viene selezionato un nuovo oratore.
  + Se uno dei nodi è corrotto, gli altri delegati possono determinare la validità della proposta confrontando le proprie versioni della proposta. Il consenso può ancora essere raggiunto, poiché sono necessari solo i due terzi dei delegati per convalidare il blocco e sostituire il delegato disonesto.
  + Utilizzando l'algoritmo Delegated Byzantine Fault Tolerance, il consenso può essere raggiunto anche quando sia il relatore che un delegato sono disonesti. Quando si confrontano i blocchi, i delegati possono vedere se l'oratore o un delegato è corrotto e possono concordare di invalidare il blocco, il che porta automaticamente alla selezione di un nuovo oratore.
  + In una di queste tre situazioni, i delegati disonesti devono controllare due terzi della rete per corrompere i dati scritti sulla blockchain. Questo obiettivo è difficile da raggiungere poiché tutti i possessori di token NEO possono votare, i delegati non sono anonimi e diventare proprietario di un nodo costa 1.000 GAS.
* A livello pratico, generare un nuovo blocco sulla catena impiega tra i 15 e i 20 secondi e si raggiunge un throughput (quantità) di transazioni vicino a 1000 TPS (Transaction Per Second). Non si spende energia e le transazioni sono definitivamente concluse dopo la conferma. Come detto, nel caso della blockchain NEO non si possono avere fork.
* Per contro, poiché i delegati devono operare con identità reali per essere eletti, non c'è anonimato sulla blockchain. Il meccanismo richiede blockchain regolamentate, che includono un certo livello di centralizzazione (esattamente ciò che blockchain come Bitcoin ed Ethereum stanno cercando di ottenere).

#### Proof-of-Elapsed-Time (PoET)

La prova del tempo trascorso (PoET) è una tecnica di consenso della rete blockchain che utilizza un sistema di lotteria equo per mantenere l'efficienza del processo evitando un consumo eccessivo di risorse ed energia. Questo algoritmo è stato sviluppato nel 2016 da parte di ricercatori di Intel utilizzando una tecnologia proprietaria, meglio spiegata nei seguenti punti.

* Su una rete blockchain, l'algoritmo decide chi può estrarre monete e chi vince un blocco utilizzando una quantità di tempo generata casualmente. L'algoritmo PoET aumenta la trasparenza garantendo che i risultati della lotteria possano essere verificati da parti esterne eseguendo un codice affidabile in un ambiente sicuro. Tutti i partecipanti hanno la stessa chance di vincere e di aggiungere un nuovo blocco, grazie alla supervisione di un controller. Questo garantisce la fine del periodo di attesa e la creazione di un nuovo blocco, verificando ogni blocco per svegliarlo o meno.

Ci sono due passi principali da considerare:

* Processo di selezione
  + In primo luogo, ogni nodo partecipante deve condividere il proprio certificato tramite Intel Software Guard Extension (SGX), che ne garantisce la validità per generare un nuovo blocco nella rete. Dopodiché, possono ottenere un oggetto timer.
  + I numeri assegnati a ciascun nodo come oggetto timer (tempo di attesa per il conto alla rovescia) dall'istruzione di generazione di numeri casuali di Intel, RAND. Genera numeri casuali difficili da individuare.
  + A questo punto, l'oggetto timer assegnato a ciascun nodo partecipante si attiva.
* Processo di generazione
  + Dopo che l'oggetto temporale termina e il nodo si sveglia, è idoneo a creare un nuovo blocco per la rete.
  + Il nodo attivo genera l'hash (utilizzando una funzione di hash come SHA-256) del suo blocco di transazioni e lo sottopone all'accettazione. Non è necessario che il nodo esegua alcun lavoro di calcolo.
  + Successivamente, l'aggiornamento viene inviato alla rete.

Ci sono alcune implicazioni importanti dell'algoritmo PoET:

* PoET richiede l'utilizzo della tecnologia SGX di Intel, il che significa che solo i computer che supportano SGX possono partecipare alla rete.
* Poiché PoET richiede che i nodi attendano un periodo di tempo casuale prima di poter creare un nuovo blocco, la velocità di creazione dei blocchi può essere influenzata dal numero di nodi nella rete.
* PoET è un algoritmo di consenso energicamente efficiente, poiché non richiede il processo di mining intensivo in termini di energia che è necessario per altri algoritmi di consenso come Proof of Work. In questo modo è veloce e molto scalabile
* Dipende tutta via da tecnologie proprietarie Intel ed è una rete che chiude e condiziona i permessi della rete che la incorpora.

#### Proof of Capacity (PoC)

Proof of Capacity (PoC) è un algoritmo di consenso utilizzato in alcune criptovalute, come Burstcoin. A differenza di altri algoritmi di consenso come Proof of Work (PoW) o Proof of Stake (PoS), PoC si basa sull'uso di spazio su disco invece che sulla potenza computazionale o sulla quantità di monete possedute.

* In breve, Invece di mostrare il tempo di calcolo pesante per estrarre un blocco come in PoW, PoC mostra le operazioni di archiviazione. Richiede ai minatori di fornire la prova della capacità di archiviazione che hanno utilizzato per estrarre un nuovo blocco di transazioni. Il PoC mira a risparmiare la potenza di calcolo per risolvere gli hash crittografici e trovare il nonce appropriato.
* Invece di alterare ripetutamente i numeri nell'intestazione del blocco e di ripetere l'hashing per il valore della soluzione come in un sistema PoW, il PoC funziona memorizzando un elenco di possibili soluzioni sul disco rigido del dispositivo di mining ancora prima che inizi l'attività di mining.

Più grande è il disco rigido, più valori di soluzione possibili possono essere memorizzati sul disco rigido, più possibilità ha un minatore di trovare il valore hash richiesto dal suo elenco, con conseguente aumento delle possibilità di vincere la ricompensa di estrazione.

L’algoritmo funziona in questo modo:

* In primo luogo, il disco rigido viene tracciato: l'elenco di tutti i possibili valori nonce viene creato attraverso l'hashing ripetuto dei dati, compreso l'account di un minatore. Ciascuno di questi nonce contiene 8192 hash, numerati da 0 a 8191. Tutti gli hash sono accoppiati in "scoop", ovvero gli hash adiacenti vengono combinati per formare una coppia di due. Ad esempio, gli hash 0 e 1 costituiscono lo scoop 0, gli hash 2 e 3 costituiscono l'hash 1 e così via.
* La seconda fase prevede l'esercizio di mining vero e proprio, durante il quale un miner calcola un numero di scoop. Ad esempio, se un minatore inizia l'attività di mining e genera uno scoop numero 38, il minatore va allo scoop numero 38 del nonce 1 e utilizza i dati di quello scoop per calcolare un valore di scadenza.
* Il processo si ripete per calcolare la scadenza di ogni nonce presente sul disco rigido del minatore. Dopo aver calcolato tutte le scadenze, il miner seleziona quella con la scadenza minima.

A livello di vantaggi:

* PoC può utilizzare qualsiasi disco rigido normale, compresi quelli con sistemi basati su Android
* Si dice che sia fino a 30 volte più efficiente dal punto di vista energetico rispetto al mining basato su ASIC della criptovaluta bitcoin
* Non è necessario un hardware dedicato o un aggiornamento costante dei dischi rigidi
* I dati di mining possono essere facilmente cancellati e l'unità può essere riutilizzata per qualsiasi altro scopo di archiviazione dati. Inoltre, sono più centralizzati

A livello di svantaggi:

* Non molti sviluppatori hanno adottato questo sistema
* È possibile che il malware influisca sulle attività di mining
* L'adozione diffusa del PoC potrebbe dare il via a una "corsa agli armamenti" per la produzione di dischi rigidi di maggiore capacità

## Tipi di blockchain

Partiamo da una distinzione esterna; le blockchain *permissionless (senza permessi), permissioned (con permessi-autorizzate)* o entrambe. In particolare definiamo che:

* Le permissioned richiedono il permesso di un’autorità centrale per partecipare. Esse limitano l'accesso alla rete a determinati nodi e possono anche limitare i diritti di tali nodi su tale rete. Le identità degli utenti di una blockchain autorizzata sono note agli altri utenti della blockchain autorizzata.
  + Le blockchain autorizzate tendono a essere più efficienti. Poiché l'accesso alla rete è limitato, ci sono meno nodi sulla blockchain, con conseguente riduzione del tempo di elaborazione per ogni transazione.
* Le permissionless non richiedono alcun permesso per partecipare e permettono agli utenti di essere pseudoanonimi (usando uno pseudonimo non si rivela alcun dettaglio relativo all’identità personale) e non restringono i permessi dei nodi.
  + Le blockchain senza permessi tendono a essere più sicure delle blockchain con permessi, perché ci sono molti nodi che convalidano le transazioni e sarebbe difficile per i cattivi attori colludere sulla rete. Tuttavia, le blockchain permissionless tendono anche ad avere lunghi tempi di elaborazione delle transazioni a causa dell'elevato numero di nodi e delle grandi dimensioni delle transazioni.

Chiaramente, la centralizzazione delle blockchain autorizzate a un'autorità centrale (che sia un governo, un'azienda, un gruppo commerciale o un'altra entità o gruppo che concede l'autorizzazione ai nodi e crea le restrizioni della blockchain) rende il sistema meno sicuro e più incline alle tradizionali vulnerabilità di hacking. Meno nodi ci sono su una blockchain, più è facile per i cattivi attori colludere, quindi gli amministratori di blockchain private devono assicurarsi che i nodi che aggiungono e verificano i blocchi siano altamente affidabili.



Riferimento: <https://www.foley.com/en/insights/publications/2021/08/types-of-blockchain-public-private-between>

I tipi di blockchain principali sono 4:

* Blockchain Pubbliche (Public Blockchain), sistema di libro mastro distribuito non restrittivo e privo di permessi. Chiunque abbia accesso a Internet può registrarsi su una piattaforma blockchain per diventare un nodo autorizzato e far parte della rete blockchain.
  + Un nodo o un utente che fa parte della blockchain pubblica è autorizzato ad accedere ai record attuali e passati, a verificare le transazioni o a fare il proof-of-work per un blocco in arrivo e a fare mining.
  + L'uso più semplice delle blockchain pubbliche è il mining e lo scambio di criptovalute. Pertanto, le blockchain pubbliche più comuni sono quelle di Bitcoin e Litecoin.
  + Su queste blockchain pubbliche, i nodi "minano" criptovalute creando blocchi per le transazioni richieste sulla rete risolvendo equazioni crittografiche. In cambio di questo lavoro, i nodi miner guadagnano una piccola quantità di criptovaluta. I minatori agiscono essenzialmente come cassieri di banca della nuova era che formulano una transazione e ricevono (o "minano") un compenso per i loro sforzi.
  + Le blockchain pubbliche sono per lo più sicure se gli utenti seguono rigorosamente le regole e i metodi di sicurezza. Tuttavia, è rischiosa solo quando i partecipanti non seguono scrupolosamente i protocolli di sicurezza.
* A livello di vantaggi:
  + Sono *sicure*, in quanto ci possono essere quanti più partecipanti o nodi; più grande la rete, maggiore la distribuzione di record ed è più difficile per gli hacker colpire l’intera rete. Inoltre, ciascun nodo farà verificata di transazioni e PoW, rendendo ogni transazione e blocco legittimi.
  + Sono *affidabili*, poiché due nodi o partecipanti non devono preoccuparsi dell'autenticità dell'altro. In altre parole, non hanno bisogno di conoscere personalmente o di fidarsi degli altri nodi, poiché il processo di proof-of-work assicura che non vi siano frodi nelle transazioni. Si può quindi fidare ciecamente delle blockchain pubbliche senza sentire il bisogno di fidarsi dei singoli nodi.
  + Sono *aperte* e *trasparenti* a tutti i nodi partecipanti. Una copia dei record della blockchain o del libro mastro digitale è disponibile presso ogni nodo autorizzato. Questo rende l'intero sistema blockchain completamente aperto e trasparente. Nessuno può mostrare una transazione falsa o nascondere una transazione esistente, poiché ogni nodo dispone di una copia aggiornata del database in qualsiasi momento.
* A livello di svantaggi:
  + Hanno meno TPS (transazioni al secondo), dato che è un enorme rete con molti nodi e ciascun nodo che verifica e faccia PoW consuma parecchio
  + Sono meno scalabili, essendo lente nel processing e nel completamento delle transazioni
  + Consumano molta energia, dato che appunto girano su PoW
* Blockchain Private (Private/Managed Blockchain), un tipo di blockchain restrittiva o con permessi, che opera solo in una rete chiusa.
  + In una blockchain privata, l'autorità centrale determina chi può essere un nodo. Inoltre, l'autorità centrale non concede necessariamente a ogni nodo gli stessi diritti di esecuzione delle funzioni. Le blockchain private sono solo parzialmente decentralizzate perché l'accesso pubblico a queste blockchain è limitato.
  + Alcuni esempi di blockchain private sono la rete di scambio di valuta virtuale business-to-business Ripple e Hyperledger, un progetto ombrello di applicazioni blockchain open-source.
  + Le reti blockchain private vengono utilizzate per le votazioni, la gestione della catena di approvvigionamento, l'identità digitale, la proprietà dei beni, ecc.
* A livello di vantaggi:
  + Sono veloci, misurate sempre in TPS, in quanto vi è un numero limitato di nodi e aggiungere nuove transazioni ad un blocco è veloce. Questo velocizza il consenso o il processo di verifica su tutti i nodi, raggiungendo un rate di centinaia/migliaia di TPS al secondo
  + Sono scalabili, in quanto essendo più controllate e ne si sceglie la dimensione sulla base degli usi dell’organizzazione; aggiungere nodi, essendo una dimensione controllata, sarà più semplice.
* A livello di svantaggi:
  + Richiede la costruzione di sicurezza (trust-building), in quanto non c’è un libro mastro-open ledger e dunque non è possibile sapere per certo tutti i colleghi
  + Risulta meno sicura, in quanto avendo meno nodi, il rischio di attacco è più alto. Se un nodo dovesse prendere il controllo del sistema di gestione centrale, prenderebbe facilmente il controllo di tutta la rete e dei nodi, abusando dell’informazione presente
  + Sono centralizzate, richiedendo un controllo specifico di identità e accessi (Identity and Access Management [IAM]) per funzionare correttamente, in quanto il sistema ha tutti i permessi di controllo ed amministrativi, gestendo tutti i nodi e quali permessi dare loro, venendo meno ad una delle idee base di blockchain.
* Consortium Blockchain, che è una tipologia semi-decentralizzata e autorizzata in cui più di un'organizzazione gestisce una rete blockchain. Tuttavia, la creazione di consorzi può essere un processo difficile, in quanto richiede la cooperazione tra diverse organizzazioni, che presenta sfide logistiche e potenziali rischi antitrust.
  + Inoltre, alcuni membri delle catene di fornitura potrebbero non avere la tecnologia necessaria né l'infrastruttura per implementare gli strumenti blockchain, mentre quelli che ce l'hanno potrebbero decidere che i costi iniziali sono un prezzo troppo alto da pagare per digitalizzare i propri dati e connettersi agli altri membri della catena di fornitura.
  + La società di software aziendale R3 ha sviluppato una serie di soluzioni consortili blockchain per il settore dei servizi finanziari e non solo.
* Blockchain Ibrida (Hybrid Blockchain), un tipo di blockchain controllata da una singola organizzazione, ma con un livello di supervisione eseguito dalla blockchain pubblica, che è necessaria per eseguire determinate convalide delle transazioni.
  + Utilizza le caratteristiche di entrambi i tipi di blockchain, ovvero può avere un sistema privato basato sui permessi e un sistema pubblico senza permessi. Con questa rete ibrida, gli utenti possono controllare chi può accedere a quali dati memorizzati nella blockchain.
  + Solo una sezione selezionata di dati o record della blockchain può essere resa pubblica, mantenendo il resto riservato nella rete privata. Il sistema ibrido di blockchain è flessibile e consente agli utenti di unire facilmente una blockchain privata a più blockchain pubbliche. Una transazione in una rete privata di una blockchain ibrida viene solitamente verificata all'interno di quella rete.

### Sidechain

* Le Sidechain, che sono delle blockchain che validano dati da altri blocchi ed è interoperabile con la blockchain principale. ­Esse permettono la creazione di nuove criptovalute o token basati su blockchain preesistenti. In pratica, le sidechain sono catene laterali che si collegano alla blockchain principale, ma che operano in modo indipendente da essa, con proprie regole e caratteristiche.



Riferimento: <https://www.coindesk.com/learn/an-introduction-to-sidechains/>

* + Esse esistono in quanto la costruzione sulle blockchain pubbliche potrebbe non gestire bene introduzione di nuove caratteristiche senza il consenso tra tutta la comunità, difficilmente gestendo grandi quantità di dati e la base di codice della catena principale
  + Le sidechain beneficiano della decentralizzazione e della sicurezza della blockchain principale sottostante e mantengono la flessibilità necessaria per risolvere casi d'uso altamente specifici.
  + L'aggiunta e la rimozione di funzionalità su una sidechain non dipende dal consenso della comunità della mainchain, poiché le nuove funzionalità interessano solo gli utenti della sidechain. Inoltre, le nuove funzionalità possono essere aggiunte a registri separati della sidechain, riducendo lo stress sulla mainchain.
* Esse possono essere usato per eseguire applicazione blockchain decentralizzate (dApp). Una tipica implementazione della sidechain crea una transazione sulla prima blockchain (la mainchain) bloccando gli asset, quindi crea una transazione sulla seconda blockchain (la sidechain) e fornisce prove crittografiche alla transazione che gli asset sono stati bloccati correttamente sulla prima blockchain.
* Le sidechain hanno una funzione di trasferimento bidirezionale con la blockchain principale, definita come *2-way peg*, che consente di bloccare un certo quantitativo di asset sulla blockchain principale, che poi vengono "trasferiti" sulla sidechain, dove vengono utilizzati per eseguire transazioni specifiche. Questa mantiene la compatibilità, dato che come detto gli asset rimangono bloccati sulla blockchain principale fino a quando non vengono restituiti. Questa deve assumere che gli attori che vi operano, definiti come *validatori*, stiano agendo onestamente.
* Esse operano in maniera *cross-chain* (*cross-chain interoperability*), che consente a diverse blockchain di comunicare tra loro e di trasferire asset e valori da una blockchain all'altra, senza dover passare attraverso la blockchain principale. Per ottenerla si utilizzano diversi approcci, per esempio l’*atomic swap*, che consente lo scambio diretto di asset tra blockchain diverse senza la necessità di intermediari. In questo modo, anche con cripto diverse, si ha subito uno scambio senza passare per la main chain.
  + Questo utilizzo due sottoprotocolli; uno per le transazioni in avanti (dalla mainchain alla sidechain) e l’altro all’indietro (dalla sidechain alla mainchain). Per il primo si ha *full referencing*, implicando che i blocchi sidechain contengono tutti i riferimento ai blocchi della mainchain. Questi sono normalmente delle hash oppure dei riferimenti tramite Merkle Tree. Ad ogni modo, occorre modificare la logica della mainchain per mantenere i riferimenti. Entrambi i trasferimenti sono asincroni
  + Un altro approccio utilizzato per la cross-chain interoperability è quello del *token wrapping*. In pratica, il token wrapping consente di creare un token rappresentativo di un asset su una blockchain diversa. Ad esempio, un token ERC-20 potrebbe essere creato per rappresentare Bitcoin su una blockchain Ethereum. Questo token potrebbe poi essere utilizzato per eseguire transazioni su Ethereum, ma rimane comunque collegato al Bitcoin reale sulla blockchain principale.
* A livello di vantaggi:
  + *Scalabilità*: Una sidechain può offrire transazioni più veloci e meno costose grazie a molte ottimizzazioni, ad esempio spostando un certo tipo di transazione su un'altra catena il cui protocollo è costruito appositamente per quel tipo di transazione. Questo dovrebbe decongestionare la prima catena, rendendola anch'essa più veloce ed economica. Le catene laterali possono anche utilizzare tecniche più veloci e più efficienti.
  + *Sperimentazione/aggiornamento*: L'aggiornamento di una blockchain consolidata con diversi soggetti interessati può essere difficile. Raggiungere il consenso può essere lento, se non impossibile. Le sidechain consentono di testare e diffondere nuove idee senza un ampio consenso.
  + *Diversificazione e governance*: Gli asset di altre blockchain possono essere resi accessibili a più persone. Applicazioni come i prestiti e i mutui nella DeFi (Decentralized Finance) possono avere accesso ad asset di altre catene. Disaccoppiandosi dalla catena principale, si possono testare anche caratteristiche indipendenti e anche creare token personalizzati
* Un esempio di sidechain è *Plasma*, un framework per la creazione di una sidechain (nota anche come child chain o plasma chain) che interagisce con la blockchain di Ethereum. L'architettura plasma può essere vista come un albero gerarchico di sidechain. Poiché ogni catena laterale opera in modo indipendente ed è parallela alla mainchain e alle altre sidechain, la velocità e l'efficienza sono ottimizzate. Inoltre, ogni sidechain può essere utilizzata per elaborare applicazioni uniche nello stesso ecosistema sicuro. Plasma utilizza il proof-of-stake come meccanismo di consenso invece del proof-of-work per garantire un'esecuzione più rapida delle transazioni.
  + Questa viene spesso utilizzata per i cosiddetti *smart contract*, che sono creati e immessi nella catena principale. Questo smart contract contiene le regole, il tasso di scambio dei token e gli hash di stato della sidechain. Una sidechain invia periodicamente informazioni sullo stato alla blockchain madre.
  + Gli impegni dei blocchi scorrono verso il basso e le uscite possono essere inviate a qualsiasi catena madre, per essere infine impegnate nella blockchain principale. La rete di illuminazione può essere implementata in cima allo strato di plasma (cioè le sidechain) per facilitare le transazioni istantanee.

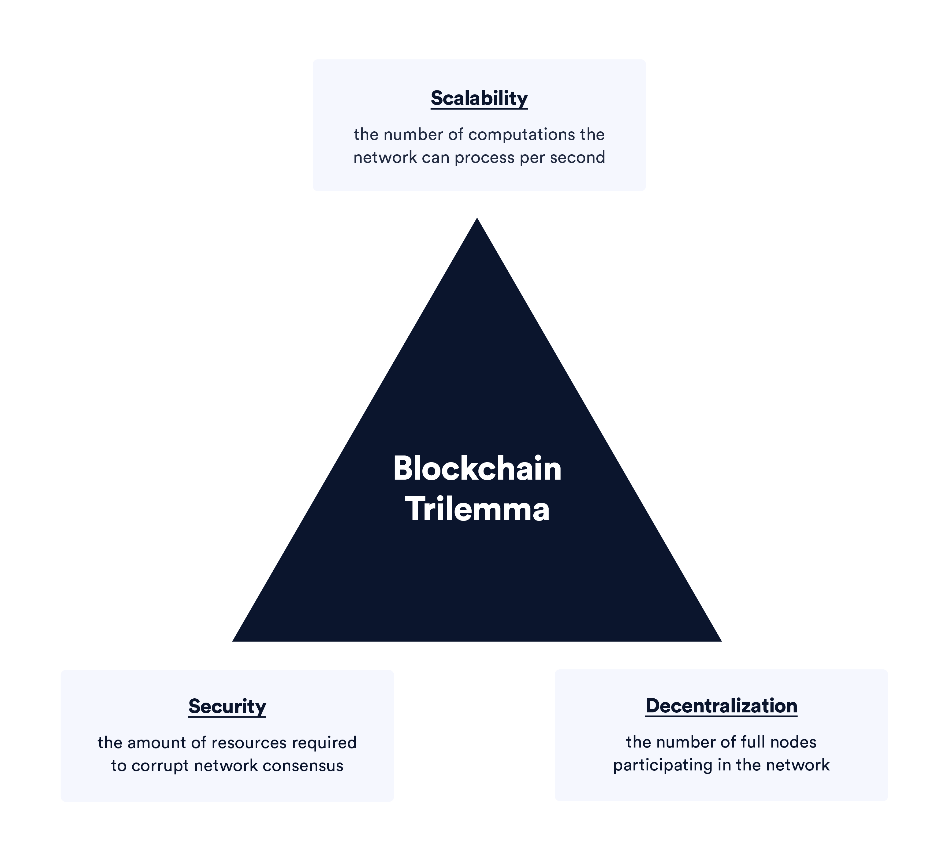
Immagine che contiene diagramma

Descrizione generata automaticamente

Riferimento: <https://crypto.com/university/what-are-sidechains-scaling-blockchain>

* A livello di svantaggi:
  + *Limitata sicurezza,* perché le sidechain non sono protette dal Layer 1 e dipendono invece dai propri validatori per la sicurezza. Attirare un gran numero di validatori interessati può essere difficile, soprattutto per le catene che non dispongono di gettoni nativi che possono servire da incentivo. Essendo catene più piccole, sono più vulnerabili a potenziali attacchi.
  + *Minore decentralizzazione*, in quanto le sidechain sono considerate più centralizzate delle catene principali. Rispetto alle blockchain principali, si affidano a un numero minore di minatori, rendendo l'elaborazione delle transazioni meno decentralizzata. Ciò solleva questioni di compatibilità con la natura decentralizzata della blockchain.

## Blockchain trilemma



Riferimento: <https://blog.chain.link/blockchain-scalability-approaches/>

Prima di parlare di come scalare le blockchain, è importante capire perché l'informatica delle blockchain è fondamentalmente diversa da quella tradizionale. In generale, le blockchain sono preziose per tre motivi:

* Calcolo deterministico: la logica codificata predefinita viene eseguita esattamente come scritta, con un livello di certezza molto elevato.
* Neutralità credibile: non c'è un amministratore centrale e non ci sono privilegi di rete speciali, il che significa che chiunque può inviare transazioni senza temere censure o discriminazioni.
* Verifica da parte dell'utente finale: lo stato storico e attuale del libro mastro della blockchain e il codice alla base del software client sono verificabili da chiunque nel mondo.

A livello più tecnico, le blockchain hanno il compito di gestire un registro interno di dati, che possono rappresentare la proprietà di beni, lo stato dei contratti o semplicemente informazioni grezze. La maggior parte delle reti blockchain è gestita da due gruppi di partecipanti sovrapposti ma distinti: i produttori di blocchi e i nodi completi.

* I produttori di blocchi raccolgono le transazioni non confermate inviate dagli utenti, ne verificano la validità e le inseriscono in strutture di dati chiamate blocchi. I produttori di blocchi sono generalmente indicati come minatori nelle catene Proof-of-Work (PoW) o validatori nelle catene Proof-of-Stake (PoS), con PoW e PoS che agiscono come meccanismi di resistenza alle sibille per garantire che i libri mastri della blockchain rimangano vivi e immuni alla censura.
* I blocchi inviati dai produttori di blocchi vengono poi accettati o rifiutati dai nodi completi, entità che memorizzano in modo indipendente una copia completa del libro mastro della catena e convalidano continuamente i nuovi blocchi, ma non sono tenuti a partecipare alla produzione dei blocchi. I nodi completi sono gestiti dalla maggior parte dei produttori di blocchi, ma includono anche gli utenti finali e gli attori economici chiave, come le borse, i fornitori di RPC e gli emittenti di stablecoin.
* In definitiva, i nodi completi mantengono l'onestà dei produttori di blocchi rifiutando i blocchi non validi, anche in una situazione in cui la maggior parte dei produttori di blocchi è malintenzionata. Questo rende la creazione di blocchi non validi uno spreco di tempo e denaro, supponendo che esista un numero sufficiente di nodi completi onesti.
* Inoltre, la separazione di tutti i nodi dai produttori di blocchi previene i validatori. Questo fa parte di un sistema di pesi e contrappesi in cui i produttori di blocchi hanno solo il potere di ordinare transazioni, ma non di dettare le regole della blockchain. Le regole sono governate dall'intera comunità dei nodi, alla quale idealmente chiunque può facilmente partecipare.
* Si parla di *trilemma*, perché allo stesso tempo si intende ridurre l’hardware necessario, al fine di abbassare la barriera di ingresso per un interno nodo, che è storicamente il modo in cui le blockchain sono rimaste decentralizzate, una componente critica per generare la minimizzazione della fiducia. Tuttavia, la decentralizzazione è anche la proprietà che generalmente rende le blockchain lente, poiché la rete può operare per lo più solo alla velocità del nodo più lento.
* Di fatto, occorre almeno sacrificare una di queste componenti

#### Decentralizzazione

* La blockchain consente di distribuire i dati contenuti nel database tra diversi nodi della rete in varie località. Questo non solo crea *ridondanza*, ma mantiene anche la fedeltà dei dati memorizzati: se qualcuno tenta di alterare un record in un'istanza del database, gli altri nodi non verrebbero alterati, impedendo così a un malintenzionato di farlo. Se un utente manomette il registro delle transazioni di Bitcoin, tutti gli altri nodi fanno un *riferimento incrociato* e individuano facilmente il nodo con le informazioni errate. Questo sistema aiuta a stabilire un ordine preciso e trasparente degli eventi. In questo modo, nessun singolo nodo della rete può alterare le informazioni in essa contenute.
* Ogni nodo ha la propria copia della blockchain e la rete deve approvare algoritmicamente ogni nuovo blocco estratto affinché la catena sia aggiornata, affidabile e verificata. Poiché le blockchain sono trasparenti, ogni azione nel libro mastro può essere facilmente controllata e visualizzata, creando una sicurezza intrinseca della blockchain. A ogni partecipante viene assegnato un numero di identificazione alfanumerico unico che indica le sue transazioni.
* La combinazione di informazioni pubbliche con un sistema di controlli e bilanciamenti aiuta la blockchain a mantenere l'integrità e a creare fiducia tra gli utenti. In sostanza, le blockchain possono essere considerate come la scalabilità della fiducia attraverso la tecnologia. Per questo motivo, le informazioni e la storia (ad esempio delle transazioni di una criptovaluta) sono *irreversibili*. Un record di questo tipo potrebbe essere un elenco di transazioni (come nel caso di una criptovaluta), ma è anche possibile che una blockchain contenga una serie di altre informazioni come contratti legali, identificazioni statali o l'inventario dei prodotti di un'azienda.
* Poniamo alcuni concetti base sulla decentralizzazione:
  + Rete distribuita: Una rete blockchain è costituita da molti nodi o computer che lavorano insieme per convalidare e registrare le transazioni sul libro mastro della blockchain. Ogni nodo mantiene una copia del libro mastro, che viene aggiornato in tempo reale quando vengono aggiunte nuove transazioni. Poiché il libro mastro è distribuito tra molti nodi, non esiste un singolo punto di errore o di controllo.
  + Meccanismo di consenso: Per garantire che tutti i nodi della rete concordino sullo stato del libro mastro, le reti blockchain utilizzano un meccanismo di consenso. Si tratta di un insieme di regole che disciplinano il modo in cui le nuove transazioni vengono convalidate e aggiunte alla blockchain. Ad esempio, la blockchain Bitcoin utilizza un meccanismo di consenso proof-of-work, in cui i nodi competono per risolvere complessi puzzle matematici per aggiungere nuovi blocchi alla catena.
  + Sistema senza fiducia (trustless system): In una rete blockchain decentralizzata, le transazioni vengono convalidate e registrate senza la necessità di un intermediario fidato, come una banca o un governo. La rete si basa invece sulla crittografia e sul consenso per garantire la sicurezza e l'affidabilità delle transazioni. Ciò significa che chiunque può partecipare alla rete senza doversi fidare di un'autorità centrale.
  + Aperta e trasparente: Le blockchain decentralizzate sono spesso aperte e trasparenti, il che significa che chiunque può visualizzare e verificare le transazioni e i dati registrati sulla blockchain. Ciò garantisce un elevato grado di trasparenza e responsabilità, particolarmente importante in settori come la finanza, la gestione della catena di approvvigionamento e le votazioni.
  + Sfide di scalabilità: Una delle sfide della decentralizzazione nella blockchain è la scalabilità. Quando un numero maggiore di utenti si unisce alla rete e vengono aggiunte più transazioni, la rete può diventare più lenta e più costosa da gestire. Questo ha portato allo sviluppo di nuove tecnologie blockchain, come lo sharding e le sidechain, che mirano a migliorare la scalabilità delle reti decentralizzate.

### Scalabilità

La scalabilità della blockchain si riferisce alla capacità di una rete blockchain di gestire un numero crescente di transazioni senza sacrificare prestazioni, velocità o sicurezza. Si tratta di un aspetto critico della tecnologia blockchain perché determina il numero di transazioni che possono essere elaborate in un determinato periodo di tempo. Nel corso del tempo si sono cercate varie soluzioni per aumentare il numero di transazioni mantenendo i costi o quantomeno diminuendoli.

I principali fattori che influenzano la scalabilità di un protocollo blockchain per quanto riguarda il throughput delle transazioni sono:

* La dimensione di una singola transazione (byte)
* La dimensione del blocco (byte -> transazioni per blocco)
* Tempo di blocco, l'intervallo in cui vengono creati i blocchi (tempo -> blocchi per unità di tempo -> transazioni per unità di tempo)
* Ritardo di propagazione sulla rete (secondi)

#### Fattori di scalabilità

La scalabilità della blockchain può essere suddivisa in tre categorie generali.

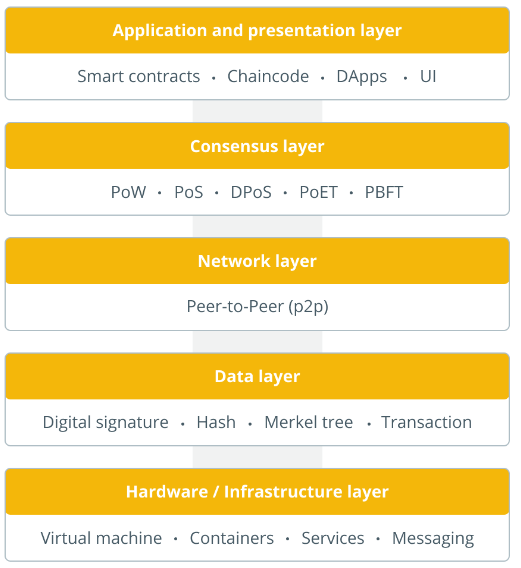
* *Esecuzione*
  + L'esecuzione della blockchain è il calcolo necessario per eseguire le transazioni e i cambiamenti di stato. L'esecuzione delle transazioni comporta il controllo della validità delle transazioni (ad esempio, la verifica delle firme e dei saldi dei token) e l'esecuzione della logica on-chain necessaria per calcolare i cambiamenti di stato. I cambiamenti di stato avvengono quando i nodi completi aggiornano la loro copia del libro mastro per riflettere i nuovi trasferimenti di token, gli aggiornamenti del codice degli smart contract e l'archiviazione dei dati.
  + La scalabilità dell'esecuzione della blockchain è comunemente considerata in termini di transazioni al secondo (TPS), ma a livello più generale si riferisce al numero di calcoli al secondo, poiché le transazioni possono variare in termini di complessità e costo. Più transazioni fluiscono attraverso una rete, più calcoli devono essere eseguiti in un dato momento.
* Quando si scala il livello di esecuzione, il problema principale da risolvere è come ottenere un maggior numero di calcoli al secondo senza aumentare sostanzialmente i requisiti hardware dei singoli nodi completi che convalidano le transazioni in blocchi.
* *Storage* (immagazzinamento)
  + L'archiviazione della blockchain si riferisce ai requisiti di archiviazione dei nodi completi, che mantengono e memorizzano una copia del libro mastro. Le blockchain hanno due forme generali di archiviazione:
    - I dati storici comprendono tutte le transazioni grezze e i dati dei blocchi. I dati delle transazioni comprendono gli indirizzi di origine e di destinazione, l'importo inviato e la firma di ogni singola transazione. I dati dei blocchi comprendono l'elenco delle transazioni e i metadati di un blocco specifico, come la radice Merkle, il nonce, l'hash del blocco precedente, ecc. I dati storici non richiedono in genere un accesso rapido e basta che ci sia almeno un'entità onesta che li renda disponibili per il download.
    - Lo stato globale è un'istantanea di tutti i dati che gli smart contract possono leggere o scrivere, come i saldi dei conti e le variabili all'interno di tutti gli smart contract. Lo stato globale può essere generalmente considerato come il database di una blockchain, necessario per convalidare le transazioni in entrata. Lo stato è comunemente memorizzato all'interno di strutture ad albero (ad esempio gli alberi di Merkle), dove l'accesso e le modifiche possono essere effettuate facilmente e rapidamente da un nodo completo.
  + I nodi completi hanno bisogno di accedere ai dati storici per sincronizzarsi per la prima volta con la blockchain e allo stato globale per convalidare i nuovi blocchi ed eseguire le nuove modifiche allo stato. Man mano che il libro mastro e la memoria associata crescono, il calcolo dello stato diventa più lento e costoso perché i nodi richiedono più tempo e calcoli per leggere e scrivere sullo stato. Se la memoria di un nodo si riempie, dovrà utilizzare lo spazio di archiviazione su disco, il che rallenta ulteriormente la computazione poiché i nodi devono passare da un ambiente di archiviazione all'altro durante l'esecuzione.
* Quando si scala il livello di archiviazione, il problema principale da risolvere è come consentire alle blockchain di elaborare e convalidare un maggior numero di dati senza aumentare i requisiti di archiviazione per i nodi completi; cioè, dove si possono archiviare i dati a lungo termine senza modificare in modo sostanziale i presupposti di fiducia delle blockchain?

* *Consenso*
  + Il consenso della blockchain è il metodo con cui i nodi di una rete decentralizzata raggiungono un accordo sullo stato attuale della blockchain. Il consenso si occupa principalmente di raggiungere una maggioranza onesta di fronte a una certa soglia di attori malintenzionati e di raggiungere la definitività, ossia che le transazioni siano elaborate con precisione e che sia altamente improbabile che vengano annullate. Il consenso della blockchain è generalmente progettato per ridurre al minimo l'overhead di comunicazione, al fine di aumentare il limite superiore della decentralizzazione per una maggiore tolleranza ai guasti bizantini e ridurre il tempo necessario per raggiungere la finalità per una liquidazione più rapida.
* Quando si scala il livello di consenso, il problema principale da risolvere è come raggiungere il risultato finale in modo più veloce, più economico e con una maggiore minimizzazione della fiducia, il tutto in modo prevedibile, stabile e accurato.

Gli approcci alla scalabilità sono generalmente tre:

* *Tecnologie di secondo livello (second layer technologies)*, come i canali di pagamento o di stato utilizzati nella Lightning Network, che portano le transazioni fuori dalla catena con l'opzione di regolare la catena in qualsiasi momento.
* *approcci Sidechain*, che possono essere meglio descritti come una parallelizzazione delle blockchain. Le attività possono essere trasferite da una catena all'altra tramite un peg (cioè, il prezzo specifico a cui punta un token, agganciando il riferimento ad un’altra valuta) bidirezionale (2WP) e il carico transazionale è condiviso tra la mainchain e le sue sidechain.
* I grafi aciclici diretti (DAG) hanno una struttura di dati diversa dalle blockchain. Molti blocchi possono essere prodotti in parallelo, il che richiede modifiche al meccanismo di consenso.

#### Blockchain e layers



Riferimento:

<https://cointelegraph.com/learn/a-beginners-guide-to-understanding-the-layers-of-blockchain-technology>

Quando parliamo degli strati della blockchain, è importante notare che ci sono due modi per comprendere la tecnologia blockchain. Possiamo infatti distinguere dei layer a livello fisico e a livello di protocollo

##### Livello fisico

Partendo dal basso:

* Livello di infrastruttura hardware (hardware layer)
  + Il primo livello della blockchain è costituito da hardware, come le connessioni di rete, i computer all'interno della rete e i server di dati. I dati memorizzati all'interno di una blockchain sono ospitati dai server di dati e i computer della rete blockchain possono condividere questi dati tra loro. Questo porta alla creazione di una rete P2P in cui le informazioni vengono convalidate dai singoli nodi (o computer) della rete.
* Livello dati (data layer)
  + In questo livello vengono gestite le informazioni memorizzate sulla rete. Questo livello è costituito da blocchi di informazioni, ogni blocco è collegato al precedente. L'unico blocco che non è collegato a un altro è il blocco genesi (il primo blocco della rete).
  + Ogni transazione scritta su questi blocchi è protetta da una chiave privata e da una chiave pubblica. La chiave privata è una firma digitale conosciuta solo dal proprietario per autorizzare una transazione; la chiave pubblica viene utilizzata per verificare chi ha firmato per la transazione. In parole povere, se qualcuno vi invia una criptovaluta, dovrà conoscere la vostra chiave pubblica; per ricevere la criptovaluta, dovrete usare la vostra chiave privata per verificare la transazione e dimostrare la vostra proprietà al vostro portafoglio blockchain.
* Livello di rete (network layer/propagation layer/P2P layer)
  + Il livello di rete, comunemente chiamato livello P2P, è responsabile della comunicazione tra i nodi. La scoperta, le transazioni e la propagazione dei blocchi sono tutte gestite dal livello di rete. Il livello di propagazione è un altro nome per questo livello.
  + Questo livello P2P garantisce che i nodi possano trovarsi l'un l'altro e interagire, diffondersi e sincronizzarsi per mantenere la rete blockchain in uno stato legittimo. Una rete P2P è una rete di computer in cui i nodi sono distribuiti e condividono il carico di lavoro della rete per raggiungere uno scopo comune. Le transazioni della blockchain sono effettuate dai nodi.
* Livello di consenso (consensus layer)
  + Il livello di consenso è essenziale per l'esistenza delle piattaforme blockchain. Il livello di consenso è il livello più necessario e critico di qualsiasi blockchain, che si tratti di Ethereum, Hyperledger o altro. Il livello di consenso ha il compito di convalidare i blocchi, ordinarli e garantire che tutti siano d'accordo.
  + Gli elementi essenziali di questo livello sono diversi, in quanto questo strato stabilisce un chiaro se di accordi tra i nodi, assicurandosi che la rete resti decentralizzata, che si segua solo una catena stabilendo a quali regole i nodi devono aderire, ottenendo unanimità di accettazione e aiutando ad ottenere affidabilità.
* Livello applicativo (application layer)
  + I contratti intelligenti, i chaincode e le applicazioni decentralizzate (DApp) costituiscono il livello applicativo. I protocolli del livello applicativo sono ulteriormente suddivisi in livello applicativo e livello di esecuzione. Il livello applicativo comprende i programmi che gli utenti finali utilizzano per comunicare con la rete blockchain. Ne fanno parte script, interfacce di programmazione delle applicazioni (API), interfacce utente e framework.
  + La rete blockchain funge da tecnologia back-end per queste applicazioni, che comunicano con essa tramite API. I contratti intelligenti, le regole sottostanti e il codice a catena fanno tutti parte del livello di esecuzione.
  + Anche se una transazione passa dal livello applicativo al livello di esecuzione, viene convalidata ed eseguita nel livello semantico. Le applicazioni danno istruzioni al livello di esecuzione, che esegue le transazioni e garantisce la natura deterministica della blockchain.

###### Scalabilità sull’execution layer

Qui parliamo di scaling verticale, quindi aggiungere più potenza di calcolo ai nodi esistenti, attraverso nuove risorse. Questo migliora i costi, l’efficienza e la manutenzione, ma rende più vulnerabile il tutto a interruzioni e a single point of failure.

L'esecuzione della blockchain può essere scalata aumentando i requisiti hardware dei produttori di blocchi. Requisiti hardware più elevati fanno sì che ogni validatore sia in grado di eseguire più calcoli al secondo.

Da una parte:

* la presenza di un’unica rete decentralizzata composta da validatori porta a blockchain con che possono supportare blocchi più grandi, tempi di blocco più rapidi e costi di transazione più bassi, pur mantenendo la componibilità sulla catena tra gli smart contract e una minimizzazione della fiducia potenzialmente più elevata rispetto ai modelli di calcolo tradizionali.

D’altro canto:

* La scalabilità verticale dei validatori limiterà la decentralizzazione della rete, dato il costo più elevato di gestione di un validatore o di un nodo completo. I costi dei nodi aumenteranno spesso nel tempo, rendendo difficile la partecipazione della maggior parte degli utenti. Il mantenimento della decentralizzazione dipenderà dalla legge di Moore, secondo la quale il numero di transistor su un microchip raddoppia ogni due anni circa, mentre il costo dei computer si dimezza. L'aumento dei costi dei nodi completi può anche aumentare i costi per gli utenti finali che vogliono verificare direttamente le attività che avvengono sulla catena, riducendo la minimizzazione della fiducia.

Ora parliamo di scaling orizzontale, quindi aggiungere nuovi nodi o nuove macchine all’infrastruttura presente. Qui è meglio da un punto di vista di downtime, aumentando la resilienza, performance e tolleranza, richiedendo però più complessità di manutenzione, operazioni e costi iniziali.

Un'alternativa allo scaling verticale è lo scaling orizzontale, attraverso l'uso di più blockchain o sidechain indipendenti all'interno di un singolo ecosistema. Lo scaling orizzontale distribuisce il calcolo delle transazioni in un ecosistema su molte blockchain indipendenti, con ogni catena che ha i propri produttori di blocchi e la propria capacità di esecuzione.

In questo caso parliamo di *scaling vertical con ecosistemi multicatena (multi-chain ecosystems).*

Da una parte:

* Gli ecosistemi multi-catena consentono al livello di esecuzione di ogni singola catena di avere caratteristiche completamente personalizzabili, come i requisiti hardware dei nodi, le caratteristiche di privacy, l'uso dei token di gas, la scelta della macchina virtuale (VM), le impostazioni dei permessi e altro ancora.
* Questo design è il motivo per cui gli ecosistemi multi-catena a volte si traducono in catene di dApp, in cui le singole blockchain si specializzano nel supporto di singole dApp o di piccole collezioni di dApp. Le blockchain auto-sovrane possono anche contribuire a isolare i rischi per la sicurezza, in quanto la scelta di una catena per la sicurezza non influisce sempre sulle altre catene dell'ecosistema.

D’altro canto:

* Gli ecosistemi multi-catena richiedono che ogni blockchain avvii la propria sicurezza attraverso un token nativo emesso in modo inflazionistico. Sebbene questo sia uno standard nelle prime fasi di crescita delle blockchain, potrebbe rivelarsi difficile passare a un modello economico meno diluitivo e più sostenibile, basato sulle tariffe degli utenti della catena, poiché le tariffe degli utenti saranno distribuite su molte blockchain indipendenti. Ci sono anche problemi di componibilità, poiché le dApp e i token che vogliono interoperare non sempre esistono sulla stessa blockchain.

Un altro meccanismo di scaling verticale è lo sharding, implementato all’interno di Ethereum, al fine di garantire il concetto di rete distribuita o trustless e mantenere il throughput. In questo caso, l’aggiornamento è noto come “eth2” sulla rete Ethereum.

Il concetto di sharding consiste nella suddivisione della rete in più sezioni, in grado di operare in maniera parzialmente indipendente, facendo in modo che le sezioni trattino ciascuna operazioni differenti.

Tale meccanismo raddoppia il volume delle transazioni eseguibili, poiché ora il limite è dovuto alla capacità elaborativa dei due nodi in contemporanea. Suddividendo dunque la blockchain in diverse sezioni è possibile aumentare linearmente il numero di operazioni supportate. L’implementazione dello sharding opera sul primo layer del network. Sarà quindi necessario eseguire un fork per aggiornare il protocollo della rete ed inserire le funzionalità di frammentazione.

Lo scopo dello sharding quindi è quello di tagliare i dati e partizionare la rete. La divisione della blockchain di Ethereum consentirebbe a ciascun nodo di memorizzare solo una parte della rete completa, mentre i nodi sarebbero in grado di convalidare la rete attraverso la matematica sottostante e la comunicazione reciproca.

In questo caso si parla di *scaling orizzontale tramite execution sharding*.

Un approccio simile, ma unico, alla scalabilità multi-catena consiste nell'avere una singola blockchain che supporta l'esecuzione parallela su molti shard diversi. Ogni shard agisce essenzialmente come una propria blockchain, il che significa che molte blockchain possono essere eseguite in parallelo. Esiste anche una singola catena principale che ha l'unico scopo di mantenere sincronizzati tutti gli shard.

Nell'execution sharding, c'è un pool di validatori che viene suddiviso tra gli shard per eseguire le transazioni. I nodi vengono ruotati in modo casuale e regolare in modo da non eseguire/convalidare sempre lo stesso shard, con un numero di shard configurato in modo da rendere statisticamente insignificante il rischio di corruzione di un singolo shard.

Da una parte:

* Tutti gli shard di esecuzione attingono dallo stesso pool di nodi, quindi non è necessario eseguire il bootstrap della sicurezza su nuovi shard. Supponendo che ci sia un ampio pool di nodi, ogni ambiente di esecuzione può raggiungere lo stesso livello di sicurezza. Lo sharding dell'esecuzione non richiede inoltre un aumento dei requisiti hardware dei nodi, poiché questi ultimi eseguono l'esecuzione su un solo shard alla volta. Gli shard possono anche operare con la stessa macchina virtuale o utilizzare configurazioni diverse per soddisfare i requisiti unici di alcuni casi d'uso.

D’altro canto:

* Ogni shard ha una flessibilità limitata, dato che tutti i nodi devono essere in grado di supportare la computazione di ogni shard. In genere esiste anche un limite al numero di shard che una blockchain può supportare, a causa dei crescenti requisiti di calcolo richiesti alla catena principale e del rischio di avere un numero troppo basso di nodi per shard. Inoltre, ci sono attriti quando si tratta di bilanciamento del carico e rischi di implementazione, dato che i modelli di sicurezza condivisi significano che tutti gli shard possono essere soggetti alla stessa vulnerabilità.

Un altro approccio di scaling è lo *scaling orizzontale tramite modularità*.

Questo è ottenuto dalle *blockchain modulari*, in cui l'architettura delle blockchain è separata in più livelli diversi, isolando cioè i componenti di esecuzione, disponibilità dei dati (DA) e consenso. Il modo più diffuso di eseguire l'esecuzione nelle implementazioni di blockchain modulari è quello dei rollup, che spostano la computazione e lo stato fuori dalla catena in reti fuori dalla catena, mentre memorizzano i dati delle transazioni nella catena. I cambiamenti di stato calcolati fuori dalla catena sono poi dimostrati proattivamente sulla catena come validi usando prove a conoscenza zero (zk-rollup) o non validi retroattivamente usando prove di frode (optimistic rollup).

Da una parte:

* Le blockchain modulari scaricano l'esecuzione delle transazioni e lo stato su un ambiente di elaborazione più economico, più snello e a più alto rendimento, pur ereditando la sicurezza della blockchain sottostante utilizzata per il regolamento.
* Questo perché il processo di consenso, in cui viene verificata la validità dei calcoli fuori catena eseguiti dal livello di esecuzione, viene eseguito da una blockchain decentralizzata esistente (cioè L1). Intuitivamente, ciò significa che la larghezza di banda computazionale di una blockchain baselayer può essere utilizzata in modo più efficiente perché i nodi completi non hanno bisogno di eseguire ogni transazione. I nodi completi devono solo verificare le prove succinte e memorizzare una piccola quantità di dati delle transazioni.

D’altro canto:

* Le blockchain modulari potrebbero non essere così veloci o economiche come le sidechain o le catene autonome, poiché la maggior parte degli approcci sfrutta l'uso dello spazio di blocco limitato e talvolta costoso di una blockchain baselayer per la sicurezza.
* Gli attuali approcci alle reti modulari comportano anche rischi di aggiornamento che richiedono l'intervento della governance (al di fuori dei rollup immutabili) e possono portare alla frammentazione della liquidità e a problemi di componibilità se alcune dApp rimangono su una blockchain baselayer mentre altre vengono eseguite su diversi livelli di esecuzione off-chain.
* Infine, l'implementazione di un rollup o di altri progetti di blockchain modulari è un processo più nuovo e più complesso rispetto al lancio di una nuova blockchain autonoma.

###### Scalabilità sul data layer

Analogamente al vertical scaling dell'esecuzione della blockchain, il vertical scaling dell'archiviazione della blockchain comporta l'aumento dei requisiti hardware per l'esecuzione di un nodo completo.

Da una parte:

* Le blockchain con limiti di archiviazione più elevati per i nodi completi possono offrire un ampio volume di archiviazione a basso costo, consentendo un accesso più semplice ai dati della catena, non avendo livelli/dipendenze aggiuntive.

D’altro canto:

* Poiché i dati da memorizzare sono sempre più numerosi nel tempo, la decentralizzazione della blockchain diventa sempre più a rischio con l'aumento dei costi di gestione di un nodo completo. Con una minore decentralizzazione, si possono fornire agli utenti meno garanzie di fiducia sulla disponibilità e sulla correttezza dei dati. I blocchi potrebbero inoltre eseguire più lentamente, aumentando la pressione sulla rete.

Un altro approccio per scalare l'archiviazione dei dati delle blockchain è il *data sharding*, effettuato su blockchain di livello 1. Il data sharding suddivide l'archiviazione del libro mastro e/o dei dati utilizzati per ricreare il libro mastro su molti shard, riducendo i requisiti di archiviazione di un singolo nodo in qualsiasi momento a quelli di un singolo shard o di un piccolo insieme di shard.

* Questo permette maggiore decentralizzazione, fornendo inoltre maggiore capacità per memorizzare transazioni; d’altro canto, potrebbero esserci dei limiti al numero di shard una blockchain può supportare, in base al carico della mainchain. È inoltre necessario un campionamento della disponibilità dei dati (DAS), che dimostri che i dati storici necessari per ricostruire una parte del libro mastro erano disponibili in un determinato momento (cioè quando è stato prodotto il blocco) senza che i nodi debbano effettivamente scaricare tutti i dati.

SI parla inoltre di memorizzazione dei dati *in modo compresso con le blockchain modulari*. Le blockchain modulari eseguono i calcoli fuori dalla catena e poi memorizzano i dati delle transazioni o le differenze di stato sia sulla catena che fuori dalla catena. I dati consentono ad altri nodi o utenti di ricostruire lo stato attuale o storico del libro mastro. Quando i rollup utilizzano l'archiviazione dei dati on-chain, i dati delle transazioni sono spesso compressi off-chain prima di essere archiviati on-chain.

* L'archiviazione compressa dei dati sulla catena è la forma più sicura di archiviazione dei dati per le blockchain modulari, perché i dati sono archiviati da tutti i nodi completi della rete. Inoltre, riduce il costo dell'archiviazione dei dati sulla blockchain di livello 1.
* La disponibilità di storage on-chain è più costosa rispetto allo storage off-chain, il che potrebbe inibire la capacità delle blockchain modulari di eguagliare la scalabilità di opzioni di storage meno decentralizzate. La compressione dei dati può anche eliminare parti dei dati non strettamente necessarie per la convalida, inibendo potenzialmente un'analisi più granulare dell'attività della catena basata su tali dati.

Possiamo inoltre avere il *data pruning*, tecnica che consente ai nodi completi della blockchain di scartare i dati storici oltre una specifica altezza di blocco. La potatura dei dati è spesso abbinata ai checkpoint Proof-of-Stake, in cui le transazioni nei blocchi oltre il checkpoint sono considerate definitive, cioè non possono essere annullate senza un grande consenso sociale o un hard fork.

* Il data pruning riduce la quantità di dati che un nodo deve memorizzare o a cui fare riferimento quando partecipa al consenso: il libro mastro è più piccolo perché i dati storici sono già stati convalidati e quindi è sicuro che vengano eliminati. Poiché i dati storici sono già stati convalidati, non sono più necessari se l'intento di un nodo completo è solo quello di convalidare i blocchi futuri, invece di offrire anche uno sguardo storico.
* Tuttavia, la potatura dei dati si basa su terze parti (ad esempio, borse, esploratori di blocchi, ecc.) per memorizzare i dati storici in modo permanente al fine di ricostruire lo stato fino al blocco genesi.
  + Si tratta di un modello di fiducia 1-ad-n, quindi solo una terza parte deve memorizzare i dati in modo onesto affinché un nodo completo sia in grado di ricreare tutto lo stato storico. Con Proof-of-Stake che offre checkpoint e una debole soggettività, questa ipotesi diventa meno rilevante. Tuttavia, tali dati sono ancora importanti per l'analisi della catena e per gli esploratori di blocchi.

Esistono anche metodi incentrati sulla limitazione della quantità di stato che i nodi completi devono memorizzare, in particolare attraverso la scadenza dello stato, la statelessness o le implementazioni di state rent.

* I progetti di scadenza dello stato (state expiry designs) consentono ai nodi di eliminare lo stato a cui non si è acceduto per un certo periodo di tempo, ma utilizzano un tipo di prova di merkle (chiamata "testimone") per far rivivere lo stato scaduto, se necessario.
* I progetti *stateless* sono quelli in cui i nodi completi non sono tenuti a memorizzare lo stato. I nodi completi devono solo convalidare i nuovi blocchi con l'inclusione di testimoni. La statelessness debole si ha quando solo i produttori di blocchi sono tenuti a memorizzare lo stato globale, mentre tutti gli altri nodi possono verificare i blocchi senza memorizzare lo stato.
* I progetti di affitto dello stato (*state rent*) richiedono che gli utenti paghino per mantenere un'archiviazione limitata dello stato. Lo stato che non viene più pagato viene riciclato e affittato a nuovi utenti.

I metodi per limitare i requisiti di memorizzazione dello stato aiutano a limitare la quantità di stato che i singoli nodi devono memorizzare. Ciò contribuisce ad alleviare il gonfiore dello stato, anche in presenza di un libro mastro in crescita o di un numero crescente di transazioni sulla catena. La limitazione della memorizzazione dello stato è fondamentale per mantenere la verifica a lungo termine da parte dell'utente finale, pur mantenendo requisiti hardware pratici.

* Tuttavia, La limitazione dell'archiviazione dello stato è un approccio piuttosto nuovo ed elimina l'idea che gli utenti paghino una sola volta per far sì che ogni singolo nodo completo della rete memorizzi il proprio stato in perpetuo, in netto contrasto con il modo in cui le blockchain gestiscono lo stato oggi.
* Inoltre, l'aggiornamento di una blockchain che utilizza un modello di memorizzazione dello stato tradizionale a un modello di memorizzazione dello stato più limitato è difficile e può rompere le applicazioni che durante lo sviluppo hanno fatto ipotesi specifiche sul fatto che lo stato sia sempre accessibile. I nuovi modelli di memorizzazione dello stato possono anche rendere determinate applicazioni più costose di quanto non fossero in precedenza.

###### Scalabilità sul consensus layer

Di seguito sono riportati quattro obiettivi generali quando si cerca di scalare i meccanismi di consenso della blockchain, che riguardano una maggiore frequenza dei blocchi, una maggiore velocità di finalizzazione e una maggiore robustezza contro i tempi di inattività o gli attacchi malevoli. Si noti che la scalabilità del consenso non riguarda solo la velocità, ma anche l'accuratezza, la stabilità e la sicurezza.

* Aumentare la capacità di esecuzione e di memorizzazione
  + Ciò consentirà a un maggior numero di nodi di partecipare al consenso o almeno impedirà ai nodi esistenti di abbandonare la rete man mano che il libro mastro cresce, contribuendo a mantenere forti garanzie di consenso in termini di tempo di attività, resistenza alla censura, accuratezza e sicurezza.
  + Se la capacità di esecuzione e di archiviazione viene aumentata a un livello significativo senza un impatto significativo sui nodi completi, le blockchain potrebbero persino essere in grado di supportare tempi di blocco più rapidi e/o dimensioni di blocco più grandi in modo stabile senza sacrificare la loro proprietà fondamentale di decentralizzazione.
* Ridurre la larghezza di banda della rete
  + Un altro modo di affrontare la scalabilità del meccanismo di consenso di una blockchain è quello di ridurre la larghezza di banda di rete, cioè l'overhead di comunicazione (invio e ricezione di messaggi) richiesto tra i nodi completi per raggiungere il consenso.
  + Invece di richiedere che i nodi siano in grado di comunicare tra tutti gli altri nodi (cioè il voto da tutti a tutti), il consenso delle blockchain può essere progettato in modo che i nodi debbano comunicare solo con una piccola parte degli altri nodi in qualsiasi momento (ad esempio, il sottocampionamento).
  + Alcuni progetti di consenso non utilizzano schemi di voto o di comunicazione a più turni, in modo che l'unica comunicazione richiesta sia la propagazione dei blocchi, ma questo generalmente va a scapito della finalità probabilistica.
* Aumentare la latenza della rete
  + Esistono anche metodi che cercano di ridurre la latenza della rete durante il consenso, in particolare per quanto riguarda l'abbassamento del tempo di finalizzazione. Alcuni meccanismi di consenso delle blockchain hanno una finalizzazione istantanea, sia attraverso un sottocampionamento a più turni, sia attraverso turni di voto "tutti contro tutti".
  + Altre blockchain implementano checkpoint garantiti da un consenso alla grande maggioranza dei validatori dopo un certo periodo di tempo, il che significa che i blocchi sono considerati definitivi dopo il checkpoint, poiché non possono più esserci riorganizzazioni in-protocollo al di là di esso. Spesso è necessario trovare un compromesso tra latenza e larghezza di banda della rete.
* Aumentare il budget per la sicurezza
  + La minimizzazione della fiducia del consenso può essere scalata anche aumentando il budget di sicurezza che finanzia i nodi che partecipano al consenso. In genere, ciò avviene ottenendo un premio monetario, con ricompense in token inflazionate, e/o aumentando i ricavi delle commissioni di transazione dato che la domanda di spazio per i blocchi supera l'offerta.
  + Budget di sicurezza più elevati aprono maggiori entrate potenziali per i partecipanti, che possono quindi aumentare la decentralizzazione della rete, poiché più nodi sono incentivati a partecipare. Le blockchain possono anche richiedere ai nodi di mettere a disposizione una maggiore partecipazione o potenza di calcolo per partecipare al consenso, anche se questo rischia di aumentare la centralizzazione della rete se i requisiti diventano troppo elevati.

##### Livello logico (protocolli)

Possiamo distinguere 4 tipi principali di layer, partendo dal:

* Layer 0, che è la blockchain stessa. I componenti necessari per rendere reale la blockchain sono internet, l'hardware e molte altre connessioni. La blockchain di livello zero è lo stadio iniziale della blockchain che permette a varie reti di funzionare, come Bitcoin, Ethereum e molte altre. Il livello 0 fornisce inoltre alla blockchain una struttura di comunicazione interoperabile tra le catene dall'alto verso i diversi livelli. Il livello 0 fornisce l'infrastruttura di base della blockchain.
* Layer 1, che è il livello di base e la cui sicurezza si basa sulla sua immutabilità. La rete Ethereum, o primo livello, è ciò a cui si allude quando si parla di Ethereum. Questo livello è responsabile dei processi di consenso, dei linguaggi di programmazione, della durata dei blocchi, della risoluzione delle controversie e delle regole e dei parametri che mantengono la funzionalità di base di una rete blockchain. È anche noto come livello di implementazione. Bitcoin è un esempio di blockchain di primo livello.
* Layer 2, noto anche come livello di esecuzione. Quando una blockchain cresce, aumenta il numero di transazioni eseguite su di essa. Per supportare l'aumento del numero di transazioni, abbiamo bisogno di soluzioni di livello 2 scalabili (in grado di gestire l'aumento del carico). Spesso vengono implementate soluzioni fuori catena (o di terze parti) per risolvere eventuali problemi nel primo livello del protocollo. Queste soluzioni non ostacolano le caratteristiche del primo livello, ma piuttosto le aggiungono.
* Layer 3, anche noto come application layer. Il compito principale di questo livello è quello di ospitare le DApp e molti altri protocolli che abilitano altre applicazioni. In questo caso, il protocollo della blockchain è suddiviso in due livelli secondari significativi, vale a dire l'applicazione e l'esecuzione. Si tratta della soluzione più potente per separare le blockchain con funzionalità cross-chain per raggiungere l'obiettivo di una reale interoperabilità.

###### Scalabilità sul layer 1

Le soluzioni blockchain di livello 1 contribuiscono a migliorare i protocolli di base (ad esempio il proof-of-work o PoW di Bitcoin) modificando il loro funzionamento per quanto riguarda l'elaborazione dei dati. Ad esempio, la rete Ethereum sta passando a un algoritmo di consenso proof-of-stake (PoS), definita come *Ethereum 2.0*.

* Ethereum è forse la migliore rete blockchain di livello 1 da utilizzare come caso di studio. È passata da proof-of-work a proof-of-stake. Nel primo caso poteva elaborare solo circa 10 transazioni al secondo. Ma la seconda ha aumentato il carico di elaborazione dei dati di Ethereum a circa 32 blocchi. Sebbene Ethereum abbia ancora molta strada da fare in termini di scalabilità, il passaggio a PoS è un passo monumentale in questa direzione.

Per aumentare il throughput di una blockchain lineare a livello di protocollo si può diminuire la dimensione delle transazioni stesse o l'intervallo in cui vengono creati i blocchi. Alcuni cambiamenti sono fatti per i canbiamenti sulla chain principale, con due soluzioni principali: Segwit e Sharding.

* Esiste la proposta Segwit (Segregated Witness), che ha creato un aggiornamento soft fork (cioè, un aggiornamento retrocompatibile) che ha permesso il cambio dei blocchi e delle loro dimensioni; i blocchi vecchi non comprenderanno le regole, ma potranno comunque convalidare blocchi e transazioni. Questa soluzione è stata implementata nel protocollo Bitcoin nel 2017, cercando di aumentare la dimensione massima dei blocchi, allora di 1 MB.
  + Senza SegWit, i dati relativi alla firma possono arrivare a occupare fino al 65% di un blocco. Con SegWit, i dati relativi alla firma vengono rimossi dall'input della transazione. Questo porta le dimensioni effettive del blocco da 1 MB a circa 4 MB.
* Invece di includere le informazioni di transazione (come mittente, destinatario, importo, definito come *scriptSig*) all'interno del blocco, SegWit crea una struttura di dati separata chiamata "Witness", che contiene solo le informazioni di autenticazione delle transazioni. Questo consente di separare le informazioni di autenticazione delle transazioni dalle informazioni di transazione reali, permettendo di ridurre il peso delle transazioni e di aumentare il numero di transazioni che possono essere incluse in un blocco.
* I creatori di Segwit avrebbero potuto lasciare che i blocchi di Segwit fossero grandi o piccoli quanto volevano e Segwit sarebbe stato comunque un soft fork, a patto che i blocchi inviati ai nodi Legacy fossero sempre di 1.000.000 di byte o meno. Una restrizione di 1MB per i blocchi Segwit non aumenterebbe affatto le dimensioni dei blocchi, mentre una restrizione di 1GB per i blocchi Segwit aprirebbe un vettore di attacco molto ovvio. Per limitare i blocchi Segwit, i creatori di Segwit hanno invece ideato una restrizione diversa dalla dimensione.
* I blocchi Segwit sono limitati da qualcosa chiamato Block Weight. Il Block Weight è un nuovo concetto introdotto in Segwit e viene calcolato per ogni transazione. Ogni transazione ha un "peso" che è definito in questo modo:
  + (dimensione della transazione con i dati del testimone rimossi) \* 3 + (dimensione della transazione)
* Le transazioni non-Segwit non hanno dati degli witnesses, quindi il peso di una transazione non-Segwit è esattamente 4 volte la dimensione. Le transazioni Segwit hanno alcuni dati witness, quindi il peso sarà inferiore a 4 volte la dimensione. Si noti che le transazioni Segwit vengono trasmesse ai nodi Legacy senza dati witness, quindi questa formula darà sempre come risultato blocchi comunicati ai nodi Legacy che sono inferiori o uguali a 1.000.000 di byte. Ancora una volta, questo è il motivo per cui Segwit è un soft fork.
* Segwit non solo aumenta la scalabilità della blockchain di Bitcoin, ma offre anche importanti vantaggi in termini di sicurezza. Poiché le informazioni di autenticazione delle transazioni sono separate dalle informazioni di transazione reali, la blockchain diventa più resistente agli attacchi di tipo malevolo come gli attacchi di double-spending. Inoltre, SegWit consente di introdurre nuove funzionalità sulla blockchain di Bitcoin, come le transazioni a pagamento multi-firma.

Un’altra soluzione pensata in questo senso è Blockchain Unlimited, il cui obiettivo è sviluppare soluzioni per aumentare la dimensione massima dei blocchi della blockchain di Bitcoin, attualmente limitata a 1 MB. Blockchain Unlimited propone di aumentare questa dimensione, ad esempio a 2 MB o 8 MB, per permettere alla blockchain di elaborare un maggior numero di transazioni in un dato periodo di tempo. Inoltre, Blockchain Unlimited sostiene che un aumento della dimensione dei blocchi della blockchain di Bitcoin può favorire la decentralizzazione della rete, permettendo a un maggior numero di nodi di partecipare alla validazione delle transazioni.

* Una delle soluzioni proposte da Blockchain Unlimited per aumentare la dimensione dei blocchi della blockchain di Bitcoin è Bitcoin Unlimited, un software alternativo al software ufficiale di Bitcoin che permette di scegliere la dimensione dei blocchi. Con Bitcoin Unlimited, i nodi possono scegliere di elaborare blocchi di qualsiasi dimensione, a patto che almeno il 75% dei nodi aderisca alla stessa dimensione di blocco. Questo sistema, secondo Blockchain Unlimited, permette una maggiore flessibilità e adattabilità della blockchain di Bitcoin, e permette di aumentare la scalabilità della rete.
* Tuttavia, la proposta di Blockchain Unlimited ha incontrato diverse critiche da parte della comunità Bitcoin, soprattutto per quanto riguarda i possibili rischi per la sicurezza della rete e la decentralizzazione della stessa. L'aumento della dimensione dei blocchi della blockchain di Bitcoin potrebbe infatti comportare un maggior rischio di attacchi di tipo malevolo, e potrebbe favorire la centralizzazione della rete, con un maggior potere di controllo dei grandi miner sulla validazione delle transazioni.

Altra soluzione citata è lo sharding, dividendo come detto la rete blockchain in vari segmenti, ciascuno governato da un gruppo specifico di nodi. In questo modo, le performance saranno aumentata e si ridurrà il tempo di risposta.

* Nello sharding, una rete blockchain è divisa in diversi segmenti, chiamati shard, e ogni shard è governato da determinati nodi che gli sono stati assegnati. Ciò consente alla rete di elaborare più transazioni contemporaneamente, aumentando il throughput complessivo e la capacità della rete.

* Questo approccio può essere utile in situazioni in cui il numero di transazioni sulla blockchain è elevato e la rete fatica a tenere il passo con la domanda. Lo sharding può anche contribuire ad aumentare la scalabilità della blockchain di una rete, in quanto è possibile aggiungere più nodi a uno shard man mano che la rete cresce. Tuttavia, lo sharding comporta anche una serie di sfide, tra cui il mantenimento della sicurezza e dell'integrità della rete e la garanzia di una comunicazione efficiente e sicura tra i diversi shard.

###### Scalabilità sul layer 2

Le soluzioni off-chain, note anche come soluzioni di scalabilità della blockchain di secondo livello, sono metodi utilizzati per aumentare l'efficienza e la velocità delle transazioni su una rete blockchain. Queste soluzioni aggiungono un livello secondario alla rete blockchain principale, progettato specificamente per gestire un volume elevato di transazioni.

Le transazioni vengono scaricate dalla blockchain principale al livello secondario per risparmiare spazio e ridurre la congestione della rete, con tempi di conferma più rapidi e una rete più efficiente. Le soluzioni off-chain offrono anche ulteriori vantaggi in termini di sicurezza e privacy, poiché il livello secondario gestisce la maggior parte delle transazioni, lasciando la blockchain principale per le transazioni più importanti o complesse; le transazioni sul livello secondario potrebbero non essere visibili all'intera rete.

Possiamo citare varie soluzioni:

* i canali di stato (state channels), che facilitano la comunicazione bidirezionale tra una blockchain e i canali transazionali off-chain e migliorano la capacità e la velocità complessiva delle transazioni. Un canale di stato non richiede la convalida da parte dei nodi della rete Layer-1.
  + Si tratta invece di una risorsa adiacente alla rete che viene sigillata utilizzando un meccanismo di firma multipla o di smart contract. Quando una transazione o un lotto di transazioni viene completato su un canale di stato, lo "stato" finale del "canale" e tutte le sue transizioni inerenti vengono registrate sulla blockchain sottostante.
* le nested blockchain, essenzialmente una blockchain all'interno - o meglio, in cima - a un'altra blockchain. L'architettura della blockchain annidata prevede in genere una blockchain principale che stabilisce i parametri per una rete più ampia, mentre le esecuzioni vengono effettuate su una rete interconnessa di catene secondarie. Su una catena principale possono essere costruiti più livelli di blockchain, con ogni livello che utilizza una connessione genitore-figlio. La catena madre delega il lavoro alle catene figlie che lo elaborano e lo restituiscono alla catena madre una volta completato. La blockchain di base non partecipa alle funzioni di rete delle catene secondarie, a meno che non sia necessaria la risoluzione di controversie.
* i rollup, che sono contratti intelligenti regolari sulla mainnet di Ethereum che fungono da relay tra la mainchain e il Layer 2, dove avvengono i calcoli. In un sistema di rollup, gli hash dello stato del contratto sono memorizzati sulla catena, insieme alle chiamate alla transazione e agli argomenti registrati come calldata. Il calcolo della transazione vera e propria avviene fuori dalla catena. Le prestazioni del rollup si ottengono raggruppando molte transazioni off-chain in un'unica transizione di stato on-chain. La sicurezza del rollup è garantita dalla capacità del contratto on-chain di verificare la correttezza delle transizioni di stato.
* Domanda di base: Come può Ethereum proteggere i rollup senza eseguire le transazioni in L1?
  + Questo è il fattore che rende i rollup diversi da una sidechain. Ethereum garantisce la genuinità degli aggiornamenti di stato. Questa approvazione è rappresentata sotto forma di prova. I rollup dimostrano la loro correttezza a Ethereum utilizzando particolari prove che consentono a Ethereum di verificare l'elaborazione delle transazioni senza eseguirle effettivamente.
* Vi sono due approcci principali:
  + In Optimistic Rollup, gli aggregatori inviano blocchi di rollup che riassumono le transazioni off-chain; chiunque può verificare la validità dei blocchi di rollup ed emettere una controversia per consentire al contratto on-chain di eseguire i calldata per verificare i blocchi di rollup.
    - Optimistic Rollup può fornire un throughput più elevato, costi più bassi, sicurezza di livello mainchain e un ambiente di esecuzione simile a quello della mainchain (ad esempio, "EVM-compatibile"), a scapito di una maggiore latenza delle transazioni dovuta alla natura del batching.
* le sidechains, semplicemente delle blockchain separate collegate alla blockchain principale. Gli asset sulla mainchain e sulle sidechain (ref) possono essere scambiati a tassi predeterminati utilizzando un peg bidirezionale (2-way peg), banalmente definito come trasferimento tra la catena principale e quella secondaria. Ciò consente di scaricare alcune transazioni e applicazioni dalla mainchain alla sidechain per ridurre la congestione della rete e aumentare la scalabilità della blockchain.
  + Più sidechain possono essere collegate alla mainchain e ogni sidechain può avere la propria architettura. È possibile creare una rete di sidechain con una mainchain in cui la mainchain può agire come una rete di relay e le sidechain rappresentano una rete di blockchain. *Plasma (ethereum)* e *Parachain (polkadot)* sono soluzioni di scala popolari che utilizzano sidechain e relay.
  + Uno sviluppatore potrebbe adottare una sidechain rispetto a un canale di pagamento o di stato se ha bisogno di maggiore flessibilità e controllo sull'infrastruttura sottostante.

Potrebbe trattarsi del lancio di un token o di una DApp, pur sfruttando i bassi costi e la maggiore velocità consentiti dal fatto di non implementare gli smart contract direttamente sulla L1.

* + Le side chain operano utilizzando il proprio meccanismo di consenso, il che significa che gli sviluppatori possono ottimizzare la loro rete per la scalabilità, la sicurezza o la decentralizzazione, e possono anche scegliere di rendere le loro reti private e autorizzate o pubbliche e senza autorizzazioni.
* i payment channels, che sono delle reti off-chain che eseguono parallelamente alla blockchain principale. L'idea è quella di stabilire un canale tra due parti che vogliono effettuare una transazione.
  + Tutte le transazioni che avvengono all'interno del canale sono fuori dalla catena e non è richiesto il consenso globale. Di conseguenza, queste transazioni vengono eseguite rapidamente tramite uno smart contract, con commissioni molto più basse e alla velocità della luce.
  + Un tipico canale di pagamento prevede tre fasi:
    - Fase uno: stabilire un canale firmando e finanziando un canale
    - Fase due: transazioni peer to peer su un canale di pagamento
    - Fase tre: chiusura del canale e trasmissione dello stato finale alla blockchain principale

## Soluzioni di scalabilità

Layer 1:

Algorand

Cardano

Layer 2:

Polygon

Plasma: <https://vitalik.ca/general/2021/01/05/rollup.html>

Lightning Network

#### Immagine che contiene grafico Descrizione generata automaticamenteDirected Acyclic Graphs (DAG)

I grafi aciclici diretti (Directed Acyclic Graphs o DAG) rappresentano una tecnologia alternativa alle blockchain tradizionali, utilizzata per la gestione dei registri digitali distribuiti. A differenza delle blockchain, basate su una struttura lineare a catena di blocchi, le DAG utilizzano una struttura ad albero, in cui i nodi rappresentano le transazioni e i rami rappresentano le connessioni tra le transazioni.

Un grafo aciclico diretto non ammette relazioni cicliche tra i nodi, come quella che si può vedere nella parte a forma di diamante del grafo diretto al centro. In termini tecnici, si direbbe che il grafo G = (V, E) è definito come l'insieme dei vertici V e degli spigoli E.

* In una DAG, ogni transazione viene aggiunta come nodo nella struttura e si connette con i nodi precedenti che hanno influenzato il suo stato. Questo processo di connessione dei nodi forma una rete di transazioni che rappresenta l'intera storia delle transazioni all'interno della DAG. La connessione tra le transazioni viene gestita in modo dinamico, in modo che le transazioni più recenti possano essere validate più velocemente, riducendo i tempi di conferma. La differenza principale è che qui transazioni multiple possono essere riferite, piuttosto che una sola alla volta
* Il DAG introduce la bidimensionalità nella struttura dei dati altrimenti lineare o unidimensionale della blockchain ed è un approccio promettente per rendere scalabili le reti decentralizzate. Rispetto a una blockchain, cambiamo solo la struttura dei dati, ma manteniamo lo stesso meccanismo di consenso, Proof-of-Work, per far sì che la rete si accordi su un'unica cronologia delle transazioni.
* Una delle caratteristiche principali delle DAG è la possibilità di eseguire operazioni di transazione in parallelo, consentendo di elaborare più transazioni contemporaneamente, senza l'attesa di una conferma per la transazione precedente. Questa capacità rende le DAG molto adatte per l'elaborazione di transazioni ad alta velocità, come ad esempio le micro-transazioni.

Esistono vari modi per impostare un DAG da utilizzare per una criptovaluta.

* Scegliere un algoritmo di DAG; di questi ne esistono molteplici:
  + IOTA's Tangle: IOTA è una criptovaluta basata su DAG che utilizza una struttura di dati chiamata Tangle. Nel Tangle, ogni transazione è rappresentata da un nodo e ogni nodo è collegato a due nodi precedenti. Quando viene effettuata una nuova transazione, questa deve convalidare due transazioni precedenti per essere confermata. Questo processo è chiamato "selezione della punta". IOTA si avvale anche di un "Coordinatore" per prevenire alcuni tipi di attacchi e garantire il buon funzionamento della rete.
  + SPECTRE (Shortest Path First Tangle): Essa è una criptovaluta basata su DAG, progettata per risolvere alcuni dei problemi di scalabilità e sicurezza del Tangle. In SPECTRE, ogni transazione è rappresentata da un nodo e i nodi sono collegati tra loro in modo da creare un grafo aciclico diretto. SPECTRE utilizza un meccanismo di consenso chiamato "greedy heaviest observed subtree" (GHOST) per confermare le transazioni e mantenere la rete.
  + Phantom: Essa è una criptovaluta basata su DAG che utilizza una struttura di dati chiamata "unità phantom" per rappresentare le transazioni. L'unità fantasma è una combinazione di due transazioni precedenti ed è collegata ad altre unità fantasma per creare un DAG. Phantom utilizza un meccanismo di consenso chiamato "Casper FFG" (Friendly Finality Gadget) per confermare le transazioni e mantenere la rete.
* Definire la struttura delle transazioni
  + Le criptovalute basate su DAG utilizzano una struttura di transazioni diversa da quella delle blockchain tradizionali. Ogni transazione è rappresentata da un nodo del DAG, collegato ad altri nodi che rappresentano transazioni precedenti. È importante definire la struttura delle transazioni e le regole per collegarle.
* Impostare meccanismo di consenso
* Determinare gli incentivi per i singoli nodi
  + In una criptovaluta basata su DAG, i nodi sono responsabili della conferma delle transazioni e della manutenzione della rete. È importante determinare gli incentivi per i nodi a partecipare alla rete, come le commissioni per le transazioni o le ricompense per i blocchi.
* Sviluppare un wallet e una UI
  + Come le criptovalute tradizionali, le criptovalute basate su DAG richiedono un portafoglio e un'interfaccia utente che permetta agli utenti di interagire con la rete. È importante sviluppare un'interfaccia facile da usare che renda semplice per gli utenti inviare e ricevere transazioni.
* Testare e lanciare

Solo poche criptovalute usano le DAG; di fatto è un modello che abbatte i costi energetici e di transazione legati al mining collegati, in quanto siamo nelle prime fasi di utilizzo; normalmente, vengono usati nella fase di test della rete e non di suo stabile utilizzo.

##### blockDAG

In un blockDAG, le transazioni sono organizzate in blocchi, che vengono poi collegati per formare un DAG. Ciò consente tempi di elaborazione delle transazioni più rapidi e una migliore scalabilità rispetto a una blockchain tradizionale, pur mantenendo la sicurezza e l'affidabilità di una blockchain.

Ecco alcune differenze tra i blockDAG e le blockchain tradizionali:

* Struttura: In una blockchain tradizionale, le transazioni sono raggruppate in blocchi collegati in una catena lineare e cronologica. In un blockDAG, le transazioni sono raggruppate in blocchi collegati in una struttura DAG.
* Consenso: Le blockchain tradizionali utilizzano meccanismi di consenso come Proof of Work o Proof of Stake per convalidare le transazioni e mantenere la rete. I BlockDAG spesso utilizzano una combinazione di meccanismi di consenso per ottenere una migliore scalabilità e sicurezza.
* Biforcazioni: Le biforcazioni possono verificarsi sia nelle blockchain tradizionali che nei blockDAG, ma il modo in cui vengono risolte è diverso. In una blockchain tradizionale, i fork vengono risolti scegliendo la catena più lunga. In un blockDAG, le biforcazioni possono essere risolte scegliendo il ramo più pesante, che tiene conto sia della lunghezza che del peso di ciascun ramo.

Immagine che contiene diagramma

Descrizione generata automaticamente

Riferimento: <https://ancapalex.medium.com/an-introduction-to-the-blockdag-paradigm-50027f44facb>

* In un libro mastro blockDAG, i nuovi blocchi fanno riferimento a tutte le punte del grafo (blocchi che non sono ancora stati referenziati) che i loro minatori vedono localmente. Come in una blockchain, i blocchi vengono pubblicati immediatamente.
* Tuttavia, a differenza di una blockchain che, per costruzione, preserva la coerenza (ogni blocco della catena aggiunge transazioni che sono coerenti con i suoi predecessori nella catena), un blockDAG incorpora blocchi provenienti da diversi "rami" e quindi può contenere molte transazioni in conflitto. Per questo motivo, un DAG, o blockDAG, non può essere considerato una "soluzione" o un "nuovo protocollo" di per sé. Invece, un blockDAG è un quadro di riferimento per l'ideazione di protocolli di consenso che possono essere (o meno) sicuri e più scalabili dei protocolli basati sulle catene.

I protocolli BlockDAG come SPECTRE e PHANTOM evitano i problemi associati agli alti tassi di orfanità. Questo comporta numerosi vantaggi:

* Permette tempi di conferma dell'ordine dei secondi, almeno quando sono visibili doppi pagamenti e conflitti.
* Consente un'ampia velocità di transazione, limitata solo dalla capacità della dorsale di rete e degli endpoint; ciò implica anche commissioni ridotte.
* Contribuisce alla decentralizzazione del mining consentendo circa 100.000 blocchi al giorno, il che riduce l'incentivo a unirsi a un pool di mining.
* Evita il rischio di orfanizzazione, con molti vantaggi aggiuntivi (come la compatibilità con il Layer Two).
* Elimina il selfish mining premiando tutti i blocchi senza discriminare tra blocchi on-chain e off-chain.

### Sicurezza

Un problema rilevante è il double spending problem (detto anche *problema della doppia spesa*), quindi occorre fare in modo che i soldi vengano spediti una volta sola senza essere copiati più volte come accade per altri file. Come detto, si riferisce alla possibilità che un utente spenda due volte la stessa unità di criptovaluta, creando due transazioni diverse che utilizzano la stessa unità di criptovaluta. Questo problema è dovuto al fatto che, se fosse permesso, porterebbe a frodi e alla svalutazione della criptovaluta.



Riferimento: <https://www.bitpanda.com/academy/en/lessons/what-is-double-spending-and-why-is-it-such-a-problem/>

* + La soluzione al problema della doppia spesa si ottiene attraverso l'uso di un *sistema di registro distribuito* (*distributed ledger system*), gestito da una rete di nodi o computer che convalidano e registrano collettivamente le transazioni. Questo sistema di libro mastro è progettato per prevenire la possibilità di doppia spesa garantendo che ogni transazione sia convalidata e confermata da più nodi della rete.
  + Quando viene avviata una transazione, questa viene trasmessa all'intera rete e ogni nodo della rete la convalida in modo indipendente per assicurarsi che il mittente abbia fondi sufficienti per completare la transazione. Una volta che un numero sufficiente di nodi ha convalidato la transazione, questa viene aggiunta alla blockchain, che è un registro permanente e a prova di manomissione di tutte le transazioni avvenute sulla rete.
  + Una volta che una transazione è confermata e registrata sulla blockchain, diventa praticamente impossibile spendere due volte la stessa unità di criptovaluta, perché qualsiasi tentativo di farlo richiederebbe la modifica della blockchain da parte di un aggressore, che verrebbe immediatamente rilevato dalla rete e rifiutato.

Un altro problema rilevante è il selfish mining, tecnica di attacco alla sicurezza delle blockchain basate sul Proof of Work (PoW). In pratica, il selfish mining consente a un miner di ottenere una maggiore ricompensa rispetto agli altri miner senza dover effettivamente eseguire un lavoro maggiore.

* Il selfish mining funziona in modo simile ad un attacco di double-spending. In pratica, il miner che utilizza il selfish mining raccoglie informazioni sulle transazioni e sullo stato corrente della blockchain, e cerca di costruire un blocco in segreto, formandosi un branch segreto.

Immagine che contiene testo, orologio, manometro

Descrizione generata automaticamente

Riferimento: <https://decentralizedthoughts.github.io/2020-02-26-selfish-mining/>

* Una volta che il miner ha costruito un blocco, lo tiene segreto invece di condividerlo con la rete. In questo modo, il miner può continuare a lavorare sul blocco successivo, aumentando le sue possibilità di risolvere il puzzle crittografico e di aggiungere il nuovo blocco alla blockchain. In pratica, continua ad estendere il suo branch segreto finché la catena pubblica non è ad un passo indietro. Allora viene pubblicato; essendo più lungo della catena principale, viene considerata come tale e i blocchi degli altri utenti vengono ignorati. Deve però essere il primo a pubblicare il blocco quando la catena è più lunga, altrimenti la strategia non funziona.
* Soluzioni possibili: cercare di assegnare in maniera casuale minatori ai rami della blockchain quando avviene un branch oppure settare limiti di soglia alle pool di mining della rete che preverrebbero ai selfish minera di avere vantaggi significativi sulla rete.

L'attacco del 51% (51 percent attack) è un attacco informatico in cui un attaccante controlla più della metà della potenza di calcolo della rete. In pratica, ciò significa che l'attaccante potrebbe confermare le transazioni malevoli, riscrivere la storia della blockchain e prendere il controllo della rete. Per effettuare un attacco del 51%, l'attaccante deve possedere una quantità significativa di risorse, come potenza di calcolo, energia elettrica e denaro per sostenere l'attacco.

* + In questo caso, si parla di *hashrate* per definire la velocità con cui un dispositivo calcola e completa l’operazione. Essendo la rete bitcoin alla base decentralizzata ed essendo basato principalmente sulla potenza piuttosto che sulla parità, potrebbe ben accadere che un dispositivo generi blocchi più velocemente degli altri, risultando quindi un problema per gli utenti onesti.
  + L'attacco Finney (Finney Attack) può essere definito come un'estensione dell'attacco selfish mining. L'attaccante estrae un blocco furtivamente e invia la transazione non confermata all'altro nodo, eventualmente a un nodo commerciante. Se il nodo commerciante accetta la transazione, l'attaccante può aggiungere un nuovo blocco alla catena in un lasso di tempo ridotto, invertendo la transazione e inducendo un attacco di doppia spesa. La finestra di attacco nel caso di un attacco Finney è notevolmente ridotta, ma può causare molti danni se il valore della transazione è sufficientemente grande.

Altro tipo di attacco comune è l’attacco Sybil, un attaccante crea molteplici identità o nodi sulla rete per aumentare la propria potenza di calcolo. In questo modo, l'attaccante può aumentare le proprie possibilità di risolvere i problemi matematici e prendere il controllo della rete. Una debolezza in un nodo può essere un motivo per generare un attacco in un qualsiasi momento, così come punti in cui si ha avuto molto guadagno.

* Ne distinguiamo due tipi:
  + *Direct Sybil Attack*, che si verifica quando un nodo maligno interagisce direttamente con i nodi onesti del protocollo. Questo tipo di attacco è particolarmente efficace perché i nodi autentici non sono in grado di rilevare che i nodi falsi non sono legittimi. Di conseguenza, l'attaccante può manipolare i nodi autentici inducendoli a intraprendere azioni in linea con i propri interessi.
  + *Indirect Sybil Attack,* che si verifica quando i nodi falsi entrano in contatto con uno o più nodi collegati ai nodi autentici. A differenza dell'attacco sibillino diretto, gli hacker utilizzano nodi intermediari/proxy per lanciare un attacco indiretto. I nodi intermedi, che sono posizionati tra il nodo sibilo e quelli onesti, rappresentano uno o più punti di fallimento. Sfruttando questi nodi intermedi, gli hacker possono sferrare un attacco ai nodi onesti che non è facilmente individuabile come un attacco sibillino diretto.
  + Oltre ad essere un problema di privacy e che potenzialmente può portare al dominio della rete con un attacco simile a quello precedente. Similmente, alcuni nodi falsi possono rifiutarsi di aggiungere nuovi blocchi alla catena (“block withholding attack”); in questo modo, si manderà in bancarotta un pezzo della rete, dato che declina l’invio di nuovi profitti.
  + Un modo per contrastare questo tipo di attacco può essere l’autenticazione diretta fornendo dettagli reali (diretta) oppure validando le proprie informazioni personali a nodi considerati verificatori (indiretta). Un altro modo può essere lo stabilire un sistema gerarchico (dato che nuovi nodi o identità possono essere nodi Sybil) oppure creare dei grafi di fiducia sociale
* Un estensione del Sybil Attack è Timejacking Attack (Attacco a tempo), in cui ogni nodo mantiene un contatore di tempo che si basa sull'ora mediana dei suoi pari; se l'ora mediana differisce dall'ora del sistema di un certo valore, il nodo ritorna all'ora del sistema. Un aggressore può inondare la rete di nodi che riportano timestamp imprecisi, il che può causare un rallentamento o un'accelerazione della rete, portando a una desincronizzazione.

Un altro tipo di attacco comune è l’Eclipse attack (attacco Eclipse), che è un tipo di attacco che prende di mira una rete blockchain isolando un nodo specifico dal resto della rete. L'attacco prende il nome dall'evento astronomico che vede un corpo celeste bloccato o oscurato da un altro oggetto, come il sole oscurato durante un'eclissi solare.

* In un attacco Eclipse, un attaccante prende di mira un nodo specifico della rete blockchain e lo inonda di nodi falsi (*sybil nodes*, come visto sopra) controllati dall'attaccante. Questi nodi sibillini formano quindi una rete attorno al nodo preso di mira, isolandolo di fatto dal resto della rete. Una volta isolato il nodo preso di mira, l'aggressore può manipolare le informazioni che scorrono tra il nodo preso di mira e il resto della rete.
* Ad esempio, se una catena ha un nodo che ha solo otto connessioni in uscita e può supportare al massimo 128 thread in qualsiasi momento, ogni nodo ha accesso alla vista solo dei nodi ad esso collegati. La vista della catena per il nodo vittima può essere modificata se un attaccante attacca un nodo specifico e ottiene il controllo degli otto nodi ad esso collegati. Questo può portare a un'ampia varietà di danni, tra cui il doppio esborso di monete ingannando la vittima che una particolare transazione non si è verificata, e anche gli attacchi contro i protocolli di secondo livello.



Riferimento: <https://wesecureapp.com/blog/attacks-on-blockchain/>

* Per prevenire un attacco Eclipse, le reti blockchain possono implementare misure quali la limitazione del numero di connessioni che un nodo può avere, l'uso di reti peer-to-peer affidabili e l'utilizzo di più fonti per verificare l'autenticità delle informazioni. È inoltre importante che i nodi siano aggiornati con le ultime patch di sicurezza e che gli utenti evitino di utilizzare reti Wi-Fi pubbliche o connessioni non sicure per accedere alle reti blockchain.

#### Immutabilità

Il concetto di immutabilità viene naturalmente ottenuto tramite la combinazione del processo crittografico con il processo di hash. Infatti:

* Ogni transazione verificata dalla rete blockchain viene registrata con un timestamp e incorporata in un "blocco" di informazioni, protetto crittograficamente da un processo di hashing che si collega e incorpora l'hash del blocco precedente e si unisce alla catena come aggiornamento cronologico successivo.
* Il processo di hashing di un nuovo blocco include sempre i meta-dati provenienti dall'output dell'hash del blocco precedente. Questo legame nel processo di hashing rende la catena "infrangibile": è impossibile manipolare o cancellare i dati dopo che sono stati convalidati e inseriti nella blockchain, perché se si tentasse di farlo, i blocchi successivi della catena rifiuterebbero il tentativo di modifica (poiché i loro hash non sarebbero validi). In altre parole, se i dati vengono manomessi, la blockchain si rompe e il motivo può essere facilmente identificato. Questa caratteristica non si riscontra nei database tradizionali, dove le informazioni possono essere modificate o cancellate con facilità.

La blockchain è essenzialmente un registro di fatti in un determinato momento. Nel caso di Bitcoin, questi fatti riguardano le informazioni sui trasferimenti di Bitcoin tra indirizzi. L'immagine sottostante mostra come la somma di controllo dei dati delle transazioni venga aggiunta come parte dell'intestazione, che a sua volta viene sottoposta a hash e diventa la somma di controllo dell'intero blocco.

Perché è importante?

* Integrità completa dei dati - I libri mastri che utilizzano la tecnologia blockchain possono garantire l'intera storia e la traccia dei dati di un'applicazione: una volta che una transazione entra nella blockchain, vi rimane come rappresentazione del libro mastro fino a quel momento. L'integrità della catena può essere convalidata in qualsiasi momento semplicemente ricalcolando gli hash dei blocchi: se esiste una discrepanza tra i dati del blocco e l'hash corrispondente, significa che le transazioni non sono valide. Ciò consente alle organizzazioni e alle autorità di regolamentazione del settore di rilevare rapidamente la manipolazione dei dati.
* Audit semplificato - La possibilità di produrre la storia completa e indiscutibile di un libro mastro transazionale consente un processo di audit facile ed efficiente. Dimostrare che i dati non sono stati manomessi è un vantaggio importante per le aziende che devono rispettare le normative di settore. Alcuni casi d'uso comuni sono la gestione della catena di approvvigionamento, la finanza e la gestione delle identità.
* Aumento dell'efficienza - Il mantenimento di un record storico completo non è solo un vantaggio per la revisione contabile, ma offre anche nuove opportunità di interrogazione, analisi e processi aziendali in generale. FlureeDB, ad esempio, sfrutta il concetto di viaggio nel tempo per le applicazioni aziendali: le query possono essere specificate a partire da qualsiasi blocco, o punto nel tempo, e riprodurre immediatamente la versione del database di quel momento.
  + FlureeDB è un database di tipo blockchain, progettato per archiviare e gestire dati in modo sicuro, trasparente e decentralizzato.
  + FlureeDB utilizza una tecnologia chiamata "blocktree", che combina i vantaggi della blockchain con quelli di una struttura ad albero. Ciò significa che i dati possono essere archiviati in modo sicuro e immutabile, ma con una maggiore flessibilità rispetto alla tradizionale catena di blocchi.
  + Inoltre, FlureeDB offre anche funzionalità come il controllo degli accessi, la gestione dei dati, le query e le analisi avanzate. Ciò consente agli sviluppatori di creare applicazioni decentralizzate e di gestire facilmente i dati in modo sicuro e affidabile.

Questa capacità consente una serie di risparmi di tempo e di costi, tra cui il monitoraggio della provenienza dei principali bug, la verifica di dati specifici dell'applicazione, il backup e il ripristino delle modifiche allo stato del database per recuperare le informazioni. L'immutabilità può rendere irrilevanti i problemi di dati più moderni che affliggono le applicazioni aziendali.

* Prova dell'errore (Proof of Fault) - Le controversie sull'errore nelle aziende sono fin troppo comuni. L'industria delle costruzioni conta 1 trilione di dollari di perdite a causa di controversie non risolte. Anche se la blockchain non eliminerà del tutto questa enorme categoria di procedimenti legali, potrebbe essere sfruttata per prevenire la maggior parte delle controversie relative alla provenienza e all'integrità dei dati (in sostanza, dimostrare chi ha fatto cosa e in quale momento).

Le blockchain, in fin dei conti, sono un meccanismo per rilevare le falsità, non di una magica macchina della verità.

La blockchain non rende i dati intrinsecamente, automaticamente o magicamente veritieri: la sua implementazione si limita a proteggerli crittograficamente in modo che non possano mai essere alterati o cancellati senza conseguenze. Misure come la condivisione dei risultati dell'hash direttamente con le parti interessate (clienti, revisori, ecc.) o la creazione di una rete decentralizzata di nodi di convalida sono un buon complemento all'immutabilità storica fornita dal processo di hashing della blockchain, per garantire una componente di convalida spesso necessaria.

Inoltre: più forti sono le regole di applicazione, più affidabili sono i dati sulla blockchain (esempio: la proof of work di Bitcoin).

* È possibile, ma incredibilmente difficile, ricreare una blockchain con nuovi dati.

Ciò richiede alcune cose: la collusione tra le parti e il ricalcolo completo degli hash. Per una blockchain pubblica come Bitcoin, entrambe le cose sono costose e richiedono un'incredibile quantità di lavoro di calcolo per ri-minizzare i blocchi fino a quando i loro hash possono essere spacciati per validi.

* Quanto più piccola è la cerchia di nodi indipendenti, tanto più facile è ottenere questo risultato.

In termini di svantaggi, avere una storia inalterabile delle transazioni sembra la risposta a molti problemi aziendali moderni - e lo è, per molti versi. Ma cosa succede quando dati sensibili, come gli indirizzi di casa dei dipendenti, vengono accidentalmente pubblicati su una blockchain? Si spera che questo non sia un problema, poiché le decisioni di progettazione standard nella costruzione di ambienti blockchain richiedono una separazione tra informazioni sensibili e informazioni di identificazione personale.

* Se si gestisce una blockchain privata e federata, l'azienda deve convincere le altre parti ad accettare una "biforcazione/fork" della blockchain - dove una blockchain si divide in due percorsi e il nuovo database designato continua. Tutte o la maggior parte delle parti coinvolte in questa blockchain dovranno concordare i termini, compreso il blocco in cui effettuare la biforcazione ed eventuali regole aggiuntive del nuovo database. Se la blockchain è veramente pubblica, è praticamente impossibile rimuovere queste informazioni (anche in questo caso è necessario un hard fork, ma è molto più improbabile che si riesca a convincere le altre parti della rete a conformarsi).
* Per sua natura immutabile, la blockchain è certamente sensibile ad attacchi come il 51 Percent dove un attaccante può acquisire un'enorme potenza di calcolo rispetto agli altri membri della rete. Una blockchain è una forma decentralizzata di rete e nessuna singola entità è al comando. I minatori, tuttavia, possono ancora scrivere la morte per l'immutabilità del sistema blockchain attraverso la creazione di potenza di hashing.

Un altro grosso problema alla blockchain sono i computer quantistici. L'informatica quantistica è considerata una potenziale minaccia per l'immutabilità della tecnologia blockchain. Questo perché i computer quantistici sono potenzialmente in grado di rompere la crittografia che protegge le chiavi private utilizzate per firmare le transazioni su una blockchain.

* In un sistema blockchain tradizionale, le chiavi private vengono utilizzate per firmare le transazioni e dimostrare la proprietà dei beni digitali. Queste chiavi sono protette da algoritmi di crittografia difficili da decifrare con i metodi di calcolo tradizionali. Tuttavia, i computer quantistici sono in grado di eseguire alcuni calcoli molto più velocemente dei computer tradizionali, il che potrebbe rendere possibile la violazione di questi algoritmi di crittografia.
* Se un computer quantistico fosse in grado di rompere la crittografia che protegge una chiave privata, potrebbe potenzialmente consentire a un aggressore di accedere e manipolare i dati memorizzati su una blockchain. Ciò potrebbe compromettere l'integrità e l'immutabilità della blockchain, in quanto le transazioni potrebbero essere alterate o cancellate senza essere rilevate.
* Per mitigare questa minaccia, ricercatori e sviluppatori stanno esplorando modi per creare algoritmi di crittografia resistenti ai quanti, in grado di resistere agli attacchi dei computer quantistici. Questi sforzi includono lo sviluppo di nuove tecniche crittografiche come la crittografia a reticolo e le firme basate su hash, che si ritiene siano più resistenti agli attacchi quantistici. Implementando queste nuove tecniche, potrebbe essere possibile garantire l'immutabilità e la sicurezza dei sistemi blockchain anche di fronte alle minacce dell'informatica quantistica.

#### Usi nelle banche

All’interno delle blockchain si sta sempre più spingendo per meccanismi di verifica degli utenti similari a quelli tradizionali già presenti. Addirittura, si prevede una sua implementazione all’interno di meccanismi di controllo finanziario tradizionale, data la sua natura immutabile, sicura e pubblica.

Possiamo citare ad esempio Il KYC (Know Your Customer), che è un processo attraverso il quale le banche ottengono informazioni sull'identità e l'indirizzo degli acquirenti. Si tratta di un processo di due diligence regolato dalle autorità di vigilanza per verificare l'identità dei clienti. Questo processo aiuta a garantire che i servizi delle banche non vengano utilizzati in modo improprio.

* Le banche sono responsabili del completamento della procedura KYC durante l'apertura dei conti. Le banche sono inoltre tenute ad aggiornare periodicamente i dati KYC dei propri clienti. La procedura KYC può essere manuale, dispendiosa in termini di tempo e ridondante tra le varie istituzioni. La condivisione delle informazioni KYC su Blockchain consentirebbe agli istituti finanziari di ottenere migliori risultati in termini di conformità, aumentare l'efficienza e migliorare l'esperienza dei clienti.
* Il processo di per sé, tradizionalmente, risulta essere ridondante, inefficiente a livello di tempo e ben poco specifico; l'architettura blockchain e la DLT ci permettono di raccogliere le informazioni provenienti da diversi fornitori di servizi in un unico database crittograficamente sicuro e immutabile che non necessita di una terza parte per verificare l'autenticità delle conoscenze. Ciò consente di creare un sistema in cui l'utente dovrà sottoporsi alla procedura KYC una sola volta per verificare la propria identità.

Il processo che implementa la verifica KYC tra banche e blockchain è il seguente:

* Per la procedura KYC, l'utente presenta i documenti a una delle banche in cui vuole ottenere un prestito o utilizzare un altro servizio.
* I singoli partecipanti sono responsabili della raccolta dei dati personali (banche, enti governativi, aziende o utenti stessi) e li memorizzano in una rete decentralizzata.
* La banca controlla e conferma il passaggio del KYC se tutto è normale.
* La banca è responsabile dell'inserimento dei dati dell'utente nella piattaforma blockchain, alla quale hanno accesso altre banche, organizzazioni e strutture statali. Tutte le parti possono controllare e regolare il processo KYC. Il sistema monitorerà le modifiche e gli aggiornamenti dei dati dell'utente e, se qualcuno viola le regole, lo sapranno tutte le parti.
* Quando un utente vuole utilizzare i servizi di un'altra banca, questa seconda banca accede al sistema e conferma così l'identità dell'utente.
* L'accesso ai dati dell'utente sarà basato esclusivamente sul suo consenso. L'utente deve effettuare il login con le transazioni in criptovaluta, cioè utilizzare la chiave privata per avviare l'operazione di scambio di informazioni.

I programmi KYC sono generalmente costituiti da tre componenti: identificazione del cliente, due diligence e monitoraggio continuo.

* Customer Identification Program (CIP)
  + Un programma di identificazione del cliente o "CIP" utilizza dati affidabili e indipendenti per garantire che il cliente sia chi dice di essere. Per le persone fisiche, ciò potrebbe includere il nome legale del cliente, la data di nascita, l'indirizzo e la documentazione di verifica come la patente di guida o il passaporto. Per i clienti aziendali, le licenze commerciali e gli atti costitutivi sono requisiti comuni.
* Customer Due Diligence (CDD)
  + La Customer Due Diligence o "CDD" è una valutazione dei rischi presentati da un nuovo cliente o da un rapporto commerciale. I fornitori di servizi finanziari si avvalgono di controlli di base, indagini sui clienti e analisi della cronologia delle transazioni dei clienti per assegnare valutazioni di rischio che determinano l'intensità del monitoraggio di un conto.
* Monitoraggio continuo
  + Il monitoraggio continuo consiste nell'esame continuo delle transazioni alla ricerca di attività criminali. Quando vengono rilevate attività sospette, i VASP sono obbligati a presentare i rapporti sulle attività sospette (SAR) alla FinCEN o ad altre agenzie di polizia competenti.

Allo stesso tempo vengono fatti dei controlli AML (Anti Money Laundering), quindi antiriciclaggio.

* Il Financial Action Task Force (FATF) stabilisce gli standard per le leggi antiriciclaggio a livello globale. Il FATF ha iniziato a pubblicare linee guida sull'antiriciclaggio delle criptovalute nel 2014 e i responsabili politici delle giurisdizioni membri del GAFI si sono rapidamente attivati; oggi, il FinCEN (Financial Crimes Enforcement Network), la Commissione Europea e decine di altri enti normativi hanno codificato in legge la maggior parte delle raccomandazioni del GAFI in materia di antiriciclaggio delle criptovalute.
* Da qui, il testimone passa ai fornitori di servizi di asset virtuali (VASP), un gruppo che il GAFI definisce come comprendente gli exchange di criptovalute, gli emittenti di stablecoin e, caso per caso, alcuni protocolli DeFi e marketplace NFT. Queste aziende fanno il lavoro pesante per fermare il riciclaggio di denaro impiegando funzionari di conformità antiriciclaggio, richiedendo controlli per conoscere i propri clienti e monitorando continuamente le transazioni di criptovaluta alla ricerca di attività sospette.
* Per valutare i rischi dei clienti e rispettare le normative antiriciclaggio, il KYC (Know Your Customer) è una funzione cruciale. L'identità dei clienti, le minacce che essi rappresentano e le attività finanziarie sono tutti elementi che rientrano nel KYC. Qualsiasi azienda o istituto finanziario deve intraprendere le seguenti azioni per il KYC:
  + Identificare il cliente
  + Esaminare il comportamento del cliente
  + Analizzare il potenziale di riciclaggio di denaro associato a ciascun cliente
* Memorizzando i dati e le informazioni su KYC e AML su un libro mastro decentralizzato, la creazione di una piattaforma AML/KYC abilitata alla blockchain può semplificare le procedure AML/KYC. Le informazioni inserite in un libro mastro blockchain non possono essere modificate o cancellate e sono sempre visibili a tutti i partecipanti alla rete. La gestione dei dati AML/KYC sulla blockchain può quindi aiutare le organizzazioni finanziarie a mantenere i dati senza problemi.
* Le proprietà intrinseche della tecnologia blockchain hanno il potenziale per fermare il riciclaggio di denaro. Ogni transazione effettuata sulla blockchain lascia una traccia cartacea permanente di registrazioni che non possono essere modificate. Di conseguenza, per le autorità è più semplice trovare la fonte originale del denaro.
* La storia completa di ogni transazione può essere monitorata, verificata e registrata in un registro pubblico della blockchain. I minatori di criptovalute e i lettori del libro mastro pubblico vengono avvisati non appena avviene una transazione. La transazione viene immediatamente bloccata se una qualsiasi delle fasi della transazione, tra cui il portafoglio di destinazione, il portafoglio di partenza, il tipo di valuta e l'importo, non viene verificata.
* Anche l'analisi del rischio di riciclaggio di denaro e i meccanismi di segnalazione sono resi possibili dalla blockchain. Consente un'analisi a livello di sistema, invece di concentrarsi solo sui punti di ingresso e di uscita. Inoltre, è tutto automatizzato. Alcuni benefici:
  + Libro mastro immutabile per il monitoraggio normativo
    - In una rete decentralizzata, la Blockchain richiede che ogni utente o nodo convalidi le modifiche. Grazie a questa caratteristica, la rete è estremamente sicura. Ogni nodo mantiene una copia dell'intero libro mastro ed è in grado di confrontare le modifiche e di identificare quelle non approvate. La Blockchain garantisce l'integrità e l'immutabilità dei dati, eliminando le modifiche non autorizzate.
  + Sviluppa la fiducia
    - Il libro mastro incorpora automaticamente la fiducia degli utenti perché è verificabile e immutabile. La blockchain rende incredibilmente semplice e sicuro stabilire la fiducia per qualsiasi transazione, facilitando il trasferimento internazionale di denaro o di altri dati sensibili.
  + Tracciamento delle transazioni
    - Una piattaforma antiriciclaggio basata su Blockchain che incorpora contratti intelligenti automatizza il processo di rilevamento delle frodi del sistema. Gli algoritmi incorporati monitorano continuamente ogni transazione, bloccano istantaneamente quelle che appaiono sospette e generano automaticamente un avviso. Utilizzando la blockchain nell'AML, è possibile ottenere il controllo su tutte le transazioni.

Alcune applicazioni utili della tecnologia blockchain:

* Apertura di un conto bancario
  + Ogni banca deve eseguire la due diligence KYC per aprire un conto bancario e confermare l'identità della persona per valutare il rischio di terrorismo o riciclaggio di denaro. Lavorare con la blockchain presenta molti vantaggi, tra cui l'eliminazione dei silos di dati, la categorizzazione del rischio e i record con data e ora. La piattaforma AML/KYC verifica se i proprietari autorizzati dell'azienda sono persone fisiche o giuridiche quando viene aperto un conto aziendale. Per completare questa verifica, sono necessari i documenti di identificazione personale.
* Richiesta di prestito
  + Secondo la normativa, il KYC è essenziale quando si richiede un prestito. Vengono effettuate valutazione del rischio di criminalità finanziaria, riciclaggio di denaro e affidabilità creditizia. I vari servizi dell'istituto possono accedere rapidamente ai dati del cliente e concludere il processo di richiesta di prestito grazie al libro mastro Blockchain.
* Miglioramenti e correzioni al meccanismo tradizionale KYC
  + Gli istituti finanziari non dovranno più chiedere ai clienti che ritornano i loro documenti per completare il processo di correzione KYC. Il libro mastro sarà utilizzato per memorizzare tutte le informazioni necessarie, compresi documenti, dati, azioni e analisi. Il processo di correzione KYC automatizzato estrae le date di scadenza delle licenze e notifica automaticamente al cliente interessato di caricare documenti più recenti e aggiornati.

# Blockchain: concetti avanzati

Tutti i concetti seguenti fanno parte del cosiddetto Web3, in cui viene data importanza alla rivendicazione della proprietà dei beni digitali, dell’identità autonoma digitale e la decentralizzazione, basandosi su blockchain. Il Web3 si propone di dare una base a tutte le tecnologie esistenti, facendo in modo che il servizio sia costruito e posseduto dagli stessi utenti. Dall’ipertesto (Web1) alla condivisione di contenuti attivi (Web2) alla decentralizzazione. Il paradigma è “lettura, scrittura, proprietà”.

La rete sicura si basa su citati meccanismi di consenso, mentre la protezione dei dati è in mano agli utenti stessi, sostituendo i server con nodi (computer) distribuiti e interagendo con applicazioni decentralizzate chiamate DApp (Decentralized Application), gestite liberamente da una comunità di utenti ma attualmente (per diverse di queste) non funzionanti con i browser tradizionali ed interagendo solo tramite criptovalute, comportando dunque l’apertura di un wallet.

Alcuni dei contenuti della successiva parte riguardano:

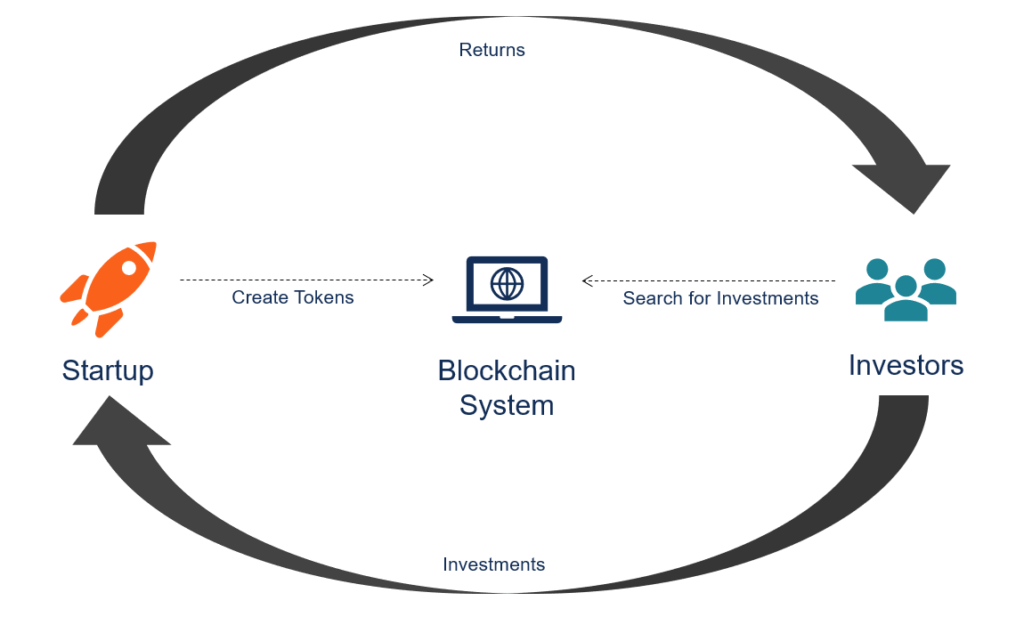
* I *token*, rappresentazioni digitali di un valore o di un'attività che possono essere scambiate e trasferite sulla rete blockchain. I token possono essere utilizzati come moneta digitale, rappresentare asset come titoli azionari, immobili o altre forme di proprietà, o essere utilizzati come strumenti per accedere a servizi o prodotti.
* Gli *smart contract*, software basati su blockchain che permettono di tracciare e gestire lo scambio di criptovalute, effettuando transazioni senza intermediare e in maniera immutabile sulla blockchain
* Le dette *applicazioni decentralizzate (dApp)*, applicazione open source e basate su smart contract, in cui l’utente detiene il pieno controllo delle proprie attività.
* Le *DAO*, organizzazioni che consentono ai creator di avere pari autorità nel contesto di creazione e distribuzione di beni e valori.

## Token

Possiamo definire i token come unità di valore accettate da una comunità e costruite su una blockchain pre-esistente (dunque, non possono essere minati), per esempio quella di Ethereum. Possono essere usati per rappresentare asset fisici come l'oro o l'immobiliare, oppure per rappresentare beni digitali come i dati o i diritti di accesso a servizi. Essendo registrati su una blockchain, sono immutabili e possono essere trasferiti in modo sicuro e trasparente da un proprietario all'altro.

Per utilizzare i token, gli utenti devono prima avere un portafoglio digitale, ovvero un'interfaccia che consente loro di accedere alla blockchain e interagire con essa. Una volta che un utente ha un portafoglio, può ricevere e inviare token. Per ricevere i token, l'utente deve fornire il proprio indirizzo del portafoglio al mittente, che poi invia i token all'indirizzo fornito. Per inviare i token, l'utente deve avere abbastanza token nel proprio portafoglio e deve conoscere l'indirizzo del destinatario a cui inviare i token.

* Viene considerato *asset* in quanto è non riproducibile e non falsificabile, dunque assumendo valore unico. Essi esistono solo in forma digitale ed è tutto ciò che ha valore per un certo stakeholder. Di fatto, viene sottoscritto dalla chiave privata del debitore, poi connesso ad una chiave pubblica del creditore, si ha una sorta di obbligazione a favore del creditore e viene giuridicamente considerato bene mobile.
* In questo contesto parliamo di ICO (Initial Coin Offering) è un evento di raccolta fondi in cui una società o un progetto emette e vende i propri token, utilizzando una blockchain come base tecnologica. Gli investitori possono partecipare all'ICO acquistando questi token con criptovalute o valuta tradizionale.
  + Nel contesto dei token blockchain, gli ICO sono spesso utilizzati come meccanismo di finanziamento per progetti che utilizzano una blockchain come base tecnologica e che intendono emettere un token per rappresentare un valore o un'utilità all'interno del loro ecosistema. L’uso di questa permette di raccogliere fondi eliminando intermediari nel processo di raccolta dei capitali



Riferimento: <https://corporatefinanceinstitute.com/resources/cryptocurrency/initial-coin-offering-ico/>

* I due tipi principali sono:
  + ICO privata
    - Nelle offerte iniziali di moneta private, solo un numero limitato di investitori può partecipare al processo. In genere, solo gli investitori accreditati (istituzioni finanziarie e persone con un elevato patrimonio netto) possono partecipare alle ICO private e una società può decidere di fissare un importo minimo di investimento.
  + ICO pubblica
    - Le offerte iniziali di moneta pubbliche sono una forma di crowdfunding che si rivolge al grande pubblico. L'offerta pubblica è una forma di investimento democratizzata perché quasi tutti possono diventare investitori. Tuttavia, a causa delle preoccupazioni normative, le ICO private stanno diventando un'opzione più praticabile rispetto alle offerte pubbliche.
* Per poterne avviare una, tutto inizia da un *whitepaper*, che permette di descrivere il progetto e il suo stato di avanzamento. Successivamente, occorre identificare gli investimenti e i possibili fondi da acquisire è la creazione dei token, dipendenti dalla piattaforma blockchain utilizzata. Allo stesso tempo, occorre una vasta campagna promozionale e poi avviene la fase di offerta agli investitori.
* Le ICO sono spesso paragonate alle offerte pubbliche iniziali (IPO), un'offerta di nuove azioni da parte di una società privata. Sia le ICO che le IPO consentono alle aziende di raccogliere fondi.

La differenza principale tra le ICO e le IPO è che le IPO comportano la vendita di titoli e sono soggette a normative molto più severe. Una società che vuole condurre un'IPO deve depositare una dichiarazione di registrazione presso la Securities and Exchange Commission degli Stati Uniti e ottenere la sua approvazione. La dichiarazione di registrazione deve includere un prospetto che fornisca i rendiconti finanziari e i potenziali fattori di rischio.

* Le ICO sono in gran parte non regolamentate. Negli Stati Uniti non esistono regolamenti specifici per le ICO. Tuttavia, se una ICO rientra nella classificazione di offerta di titoli, allora ricade sotto la giurisdizione della SEC ed è regolata dalle leggi federali sui titoli. Alcuni Paesi hanno adottato una posizione rigida e hanno vietato completamente le ICO. Tra i Paesi che hanno vietato le ICO vi sono Cina, Nepal, Bangladesh, Macedonia, Bolivia ed Ecuador.
* A livello di vantaggi:
  + Offrono un elevato potenziale di profitto se si è in grado di determinare quale criptovaluta è un buon investimento. Poiché si acquista in anticipo, i prezzi sono spesso più bassi e alcune ICO offrono token a prezzi scontati.
  + Le ICO sono accessibili a tutti. A differenza di alcune IPO, non ci sono restrizioni su chi può investire.
  + È un modo rapido ed efficiente per le start-up di raccogliere fondi.
* A livello di svantaggi:
  + Poiché i progetti di criptovaluta sono volatili, c'è un rischio significativo che il token perda valore o finisca per fallire del tutto.
  + La mancanza di regolamentazione porta a un numero maggiore di truffe e progetti mediocri. Cercare un progetto di qualità tra le prossime ICO può sembrare come cercare un ago in un pagliaio.
  + Di solito è necessaria una certa conoscenza dei portafogli di criptovalute per investire nelle ICO. Per coloro che sono alle prime armi con le criptovalute, spesso è più facile attenersi alle azioni delle criptovalute o alle monete quotate in borsa.
* Ecco alcuni esempi delle principali ICO nel corso degli anni:
  + Ethereum: Molti appassionati di criptovalute erano entusiasti di Ethereum e della sua blockchain programmabile quando si è svolta l'ICO nel luglio 2014. Alla fine ha raccolto 18,4 milioni di dollari ed è diventata la seconda criptovaluta più grande.
  + Cardano (CRYPTO:ADA): Cardano ha migliorato alcuni aspetti di Ethereum e ha avuto un'ICO di successo ancora maggiore. Nel gennaio 2017 ha raccolto 62,2 milioni di dollari. Alla fine entrerà nella top five delle criptovalute per capitalizzazione di mercato.
  + Tezos (CRYPTO:XTZ): Tezos ha raccolto 232 milioni di dollari attraverso la sua ICO nel luglio 2017, ma non è stato un successo completo. Ci sono stati numerosi ritardi nella distribuzione dei token venduti attraverso l'ICO, che hanno portato a un'azione legale collettiva. Tezos ha raggiunto un accordo di 25 milioni di dollari con tutte le parti nel 2020.
  + Dragon Coins (CRYPTO:DRG): Ci sono state molte ICO fallite, e Dragon Coins è uno degli esempi di più alto profilo. Nel marzo 2018 ha raccolto 320 milioni di dollari. Una serie di controversie ha causato un calo quasi immediato del prezzo quando è stata resa disponibile per il commercio pubblico. Nel 2021, la sua capitalizzazione di mercato è scesa sotto il milione di dollari.

### Tipologie di token

* *Payment Tokens,* più comunemente noti come "crypto". Come si può intuire, questi token sono utilizzati per acquistare e vendere beni e pagare le commissioni delle transazioni basate sulla blockchain senza la necessità di un intermediario.
  + Tecnicamente, la maggior parte delle criptovalute potrebbe essere classificata come token di pagamento, a patto che vengano utilizzate come forma di scambio. Più comunemente, però, i token di pagamento non vengono utilizzati per altro che per i pagamenti. Pertanto, non forniscono ai titolari alcuna utilità aggiuntiva oltre al loro valore monetario.
  + Un loro esempio è rappresentato dai token Ethereum o Bitcoin nelle rispettive blockchain.
* *Stablecoins,* che sono è un tipo di criptovaluta agganciata a un bene stabile come il dollaro USA. Lo scopo di una stablecoin è quello di fornire un'alternativa più stabile e meno volatile alle altre criptovalute, come Bitcoin ed Ethereum, che spesso subiscono notevoli fluttuazioni di prezzo.
  + Le monete stabili sono spesso utilizzate come pagamento per beni e servizi e offrono agli utenti commissioni di transazione più basse rispetto ad altre criptovalute e persino ad alcuni popolari servizi di trasferimento di denaro come PayPal. Le Stablecoin sono particolarmente convenienti per l'esecuzione di transazioni internazionali, in quanto le commissioni non aumentano solo perché qualcuno vive dall'altra parte del mondo.
  + Alcuni esempi:
    - Tether (USDT) è forse la più nota stablecoin agganciata al dollaro USA. Tether è stata creata per essere un'alternativa stabile e trasparente alle tradizionali valute fiat da utilizzare in una serie di transazioni di criptovalute.
    - USDC è un'altra stablecoin legata al dollaro USA. È stata progettata per fornire una moneta stabile trasparente e completamente garantita ed è emessa da numerose società, tra cui Circle e Coinbase.
    - Bacon Protocol (bHOME) è un token sostenuto da mutui reali per case e proprietà negli Stati Uniti. Questa moneta può effettivamente crescere di valore man mano che i prestiti vengono rimborsati e gli interessi vengono pagati nel contratto bHome.
* *Defi Tokens,* sono creati appositamente per essere utilizzati in combinazione con piattaforme decentralizzate che operano tramite contratti intelligenti. I token Defi hanno diversi utilizzi, come fornire garanzie per i prestiti, liquidità per il trading e consentire agli utenti di partecipare all'agricoltura di rendimento. Questi token consentono inoltre agli utenti di accedere a informazioni che esistono al di fuori della blockchain, integrando i dati nella rete.
  + Alcuni esempi:
    - Chainlink (LINK) consente agli smart contract di connettersi in modo sicuro a fonti di dati esterne, permettendo di integrare nella blockchain dati ed eventi del mondo reale.
    - Uniswap (UNI) consente agli utenti di utilizzare un modello di market maker automatizzato (AMM) per acquistare e vendere una varietà di criptovalute su uno scambio decentralizzato.
    - Aave (AAVE) consente agli utenti di prestare e prendere in prestito una varietà di criptovalute attraverso una piattaforma decentralizzata di prestiti e prestiti che impiega un modello unico di pool di liquidità.
  + In questo contesto, viene impiegata una tattica di guadagno nota come *yield farming*. Al suo livello di base, lo yield farming consente agli investitori in criptovalute di trarre profitto dai loro investimenti. Lo yield farming è un metodo per guadagnare interessi dalle commissioni di trading depositando unità di criptovaluta in un meccanismo di prestito consentendo consente agli investitori in criptovalute di bloccare le proprie attività in cambio di ricompense.
    - Essa è analoga all'accensione di un prestito bancario. Quando una banca vi presta del denaro, dovete ripagarlo con gli interessi. Lo Yield farming funziona in modo simile, ma questa volta le banche sono investitori in criptovalute come voi.
    - In questo modo, si stabilizza la liquidità e il prezzo del token, allo stesso tempo creando maggiore partecipazione e coinvolgimento sulla rete
* *Governance Tokens,* che sono un tipo di criptovaluta che consente ai titolari di token di votare sulla direzione di un progetto blockchain. Lo scopo principale dei token di governance è quello di decentralizzare il processo decisionale e di dare ai titolari voce in capitolo sulla gestione del progetto.
  + I titolari di token di governance sono di solito più investiti nel successo del progetto, poiché possono guadagnare o perdere di più, a seconda del risultato. I membri della comunità possono utilizzare i token per influenzare direttamente la direzione e le caratteristiche di un protocollo blockchain. In questo modo, è possibile implementare cambiamenti relativi all'interfaccia utente, votare sulle commissioni e sulla distribuzione delle ricompense o persino modificare il codice sottostante di un progetto.
  + Invece, il possesso di token di governance conferisce al titolare il diritto di partecipare alla governance della piattaforma. Ciò può includere il voto sulle proposte di modifica del codice della piattaforma, la determinazione delle modalità di allocazione delle risorse e altre decisioni relative al funzionamento della piattaforma.
  + In molti casi, i token di governance sono utilizzati anche per incentivare la partecipazione al processo di governance: i titolari vengono ricompensati con token aggiuntivi o altri vantaggi per aver partecipato attivamente al processo decisionale della piattaforma.
* *Utility Tokens*, che consentono di acquistare un determinato bene o servizio e conferisce al titolare un diritto di opzione per l’acquisto o somministrazione di cose o per la fornitura di servizi (attuali o futuri). Spesso definiti "coupon digitali" o "redeemable", i token di utilità consentono ai titolari di accedere a prodotti e servizi specifici.
  + Sebbene alcuni dei servizi forniti da un token di utilità siano offerti all'interno dell'ecosistema digitale della blockchain, molti dei benefici potrebbero includere vantaggi della vita reale come prodotti fisici, accesso esclusivo a eventi o persino programmi di mentorship.
  + A differenza degli NFT, i token di utilità sono fungibili, ovvero possono essere scambiati con un altro token dello stesso tipo o con un altro token di pari valore.
  + Alcuni esempi:
    - ApeCoin (APE) è un token di utilità creato appositamente per l'ecosistema Bored Ape Yacht Club. Permette ai possessori di partecipare alle votazioni della governance del progetto, oltre a fornire l'accesso a funzioni riservate ai membri, come giochi, eventi e altri servizi.
    - Decentraland (MANA) è un token utilizzato come pagamento nel mondo virtuale di Decentraland. Viene utilizzato principalmente per acquistare LAND e altri beni virtuali offerti nel gioco.
    - Enjin Coin (ENJ) è un popolare token di utilità sostenuto da miliardi di asset basati sulla blockchain. È noto soprattutto per il suo coinvolgimento nell'ecosistema dei giochi e supporta il commercio di beni di gioco.
* *Security Tokens*, che rappresentano la titolarità di un’attività finanziaria e conferisce un diritto di credito a cui per legge o per contratto possono accedere altri diritti o altre situazioni giuridiche soggettive, come per esempio il diritto di voto che spetta al titolare di un’azione societaria o il diritto alle cedole nelle obbligazioni.
  + Questi token, in sostanza, sostituiscono gli strumenti di investimento tradizionali come le azioni o le obbligazioni, offrendo agli investitori una proprietà frazionata tramite token digitali.
  + Utilizzando la tecnologia blockchain, l'entità definisce i criteri di investimento prima di tokenizzarli e offrirli agli investitori. Non è molto diverso da quando le società distribuivano agli investitori certificati azionari cartacei.
  + La differenza principale è che, poiché questi token vivono sulla blockchain, esiste un record pubblico e immutabile che chiunque può visualizzare e a cui fare riferimento.
  + Alcuni esempi:
    - BCAP è un token per smart contract basato su Ethereum ed è il primo fondo di venture tokenizzato di Blockchain Capital.
    - EXOD è la rappresentazione digitale di un'azione ordinaria di Classe A di Exodus Movement. Questo token consente sia agli investitori al dettaglio che agli investitori accreditati di partecipare alle azioni.
    - VEVU è un token creato appositamente per una società del Wyoming che consente agli investitori internazionali di investire con azioni senza diritto di voto (il che significa che i titolari non hanno diritto di voto per l'elezione degli amministratori).
* *Privacy Coins*, le quali consentono di inviare denaro in modo anonimo. Utilizzando vari metodi, le transazioni completate con le monete privacy rendono difficile per chiunque verificare chi ha inviato o ricevuto il pagamento.
  + Inoltre, i dettagli del pagamento, compresi l'importo e le date della transazione, rimangono anonimi. L'anonimato si ottiene generalmente utilizzando un indirizzo segreto unico e mescolando numerose transazioni in una sola.
  + Alcuni esempi:
    - Zcash (ZEC) è una criptovaluta veloce e riservata che offre agli utenti commissioni ridotte. Utilizza un protocollo di sicurezza per garantire che ogni parte coinvolta sia verificata senza rivelare alcuna informazione sulla parte o sulla rete.
    - Monero (XMR) è utilizzata per pagare beni e servizi in tutto il mondo in tutta riservatezza e con commissioni di transazione estremamente basse.
    - Dash (DASH) è una valuta che consente di inviare denaro a chiunque, ovunque, in modo anonimo e per meno di un centesimo.
* *Exchange Tokens*, tipo di criptovaluta emesso da una borsa. Questi token sono generalmente creati per aumentare la liquidità di un exchange e incentivare le persone a utilizzare la sua piattaforma. Sebbene efficaci, i token di scambio sono uno dei tipi di criptovaluta più volatili sul mercato.
  + I token di scambio sono comunemente utilizzati per pagare le commissioni di transazione sullo scambio. Inoltre, anche se meno comunemente, possono essere utilizzati come token di governance per offrire ai titolari un potere di voto per lo sviluppo futuro della piattaforma.
  + Alcuni esempi:
    - Binance USD (BUSD) è emesso dall'exchange Binance e sostenuto da una riserva di dollari USA. È completamente garantito e soggetto a verifiche trimestrali per garantire l'integrità della riserva. Il BUSD è spesso utilizzato per il trading, i pagamenti, i trasferimenti e come riserva di valore.
    - FTX (FTT) era il token principale dell'exchange FTX prima del suo fallimento. Veniva utilizzato come forma di pagamento con l'ulteriore vantaggio di ridurre le commissioni di trading e di offrire agli utenti diverse altre utilità sulla piattaforma.
    - Crypto.com (CRO) è il token nativo di Crypto.com creato per facilitare le transazioni all'interno della sua blockchain e del suo ecosistema finanziario.
* *Meme Coins,* criptovaluta che si ispira ad alcuni dei meme più popolari al mondo e ad altre barzellette di internet. Questi token di solito non hanno un valore iniziale né sono sostenuti da un bene o da un'utilità sottostante, ma il loro valore è completamente dipendente e derivato dalle comunità che circondano le barzellette.
  + Al di là del sentimento della comunità verso la moneta, non c'è alcun valore da ottenere dal possesso di una moneta meme. Soprattutto se si considera che molte monete meme hanno un'offerta illimitata e non hanno un meccanismo di distruzione, per cui il numero totale di monete in circolazione continua ad aumentare, facendo scendere ulteriormente il valore.
  + Alcuni esempi:
    - Dogecoin (DOGE) è di gran lunga la moneta meme più popolare. È stata originariamente creata per scherzo e ha finito per guadagnare un po' di trazione quando Elon Musk l'ha lanciata. Può essere utilizzata per i pagamenti, tuttavia, nonostante la sua popolarità, rimane estremamente volatile.
    - Shiba Inu (SHIB) è una parodia del Dogecoin. Può essere utilizzata per acquistare beni e servizi nell'ecosistema, ma al di là di questo, è praticamente priva di valore.
    - SafeMoon (SFM) è tecnicamente classificata come una moneta meme a causa della sua comunità online fortificata chiamata SafeMoon Army, tuttavia la società spera che il suo utilizzo come valuta preferita per gli scambi online diventi una realtà.

### Standard per i token

La creazione di token usa le blockchain per trasmetterli in maniera sicura senza intermediari ed utilizza gli smart contract per crearne di nuovi.

La loro creazione segue alcuni standard, di seguito listando i principali:

* ERC-20, uno standard tecnico utilizzato per creare e gestire token su blockchain Ethereum. La sigla "ERC" significa "Ethereum Request for Comment" e il numero "20" rappresenta l'identificatore univoco dell'implementazione del protocollo. Lo standard generale dei token offre la funzionalità di base per il trasferimento dei token, oltre a supportare l'approvazione dei token per le spese da parte di altre terze parti della catena.
* I token ERC-20 sono token fungibili e Le aree di applicazione più popolari sono le Stablecoin, i token di governance e le ICO. Esso richiede una singola operazione per ogni transazione, senza verifiche KYC (Know Your Customer), con uno smart contract comune e richiedendo più spazio; esso trasferisce 1/2 token alla volta.
* I token ERC-20 hanno le seguenti funzionalità standard:
  + Saldo del token: ogni account può avere un saldo di token associato ad esso.
  + Trasferimento di token: i token possono essere trasferiti da un account all'altro.
  + Approvazione di token: gli account possono approvare altri account per prelevare un determinato numero di token.
  + Ottenere il saldo di token di un account: gli account possono controllare il saldo di token di un altro account.
  + Ottenere il totale di token disponibile: gli account possono controllare il numero totale di token in circolazione.
* In questo standard sono presenti 9 regole, di cui 6 obbligatorie e 3 opzionali.
  + Regole e norme obbligatorie
    - allowance (controllo della transazione e ne permette l’esecuzione se il bilancio è sufficiente)
    - approve (approvazione del contratto per ottenere soldi)
    - transferFrom (per automatizzare alcune transazioni e renderle fisse)
    - transfer (trasferimento dei token ad un altro indirizzo)
    - balanceOf (numero totale di token che un indirizzo ha)
    - totalSupply (numero totale di token da creare)
  + Regole/standard opzionali
    - Token Name
    - Decimal (massimo: 18)
    - Immagine che contiene testo

      Descrizione generata automaticamenteSymbol

L’utilizzo di ERC20 permette lo sviluppo delle DApp standard, rendendo l’implementazione semplice per i blockchain developer, offrendo un protocollo di ampia adozione, personalizzabile e con ampia liquidità. Per contro. Tuttavia, possono essere soggetti a problemi di sicurezza (sfruttando ad esempio il bug sulla funzione *transfer* per cui il recipiente non riceve notifica dei fondi e se non riceve il token, nessuno può usarlo), richiedendo maggiore centralizzazione e regolamentazione per utilizzo.

Occorre usare per il bug di prima la combinazione "approve" e "transferFrom". Se si inviano token a uno smart contract usando la funzione "transfer", si vedrà una transazione riuscita, ma il contratto non riceverà mai i token. Questo bug brucia i token in modo permanente e non è possibile recuperarli.

A livello di codice si presenta come a lato.

Riferimento: <https://www.leewayhertz.com/erc-20-vs-erc-721-vs-erc-1155/>

* ERC-721, specificamente legato agli NFT. A differenza dei token ERC-20, che sono intercambiabili con altri token dello stesso tipo, i token ERC-721 sono unici e non possono essere sostituiti uno con l'altro. Ogni token ERC-721 rappresenta un oggetto digitale unico e può essere utilizzato per rappresentare qualsiasi cosa di valore, come ad esempio un'opera d'arte, un oggetto collezionabile, un immobile o un diritto d'autore.
  + Gli ERC-721 sono utilizzati principalmente in applicazioni decentralizzate (dApp) che richiedono la proprietà univoca e la trasferibilità di oggetti digitali, come i mercati NFT, i giochi blockchain e le piattaforme di condivisione di contenuti.
  + Si noti che ogni NFT ha una variabile uint256 nota come *tokenId*. Pertanto, per ogni contratto EBR-721, la coppia indirizzo del contratto - tokenId uint256 deve essere unica.
  + Inoltre, le dApp devono disporre di un "convertitore" per regolare il processo di ingresso e uscita degli NFT. Ad esempio, il convertitore considera il tokenId come input e produce token non fungibili come un'immagine di zombie, uccisioni, oggetti da collezione del gioco, ecc.

A livello di codice si presenta così:

Immagine che contiene testo

Descrizione generata automaticamente

Riferimento: <https://www.leewayhertz.com/erc-20-vs-erc-721-vs-erc-1155/>

* ERC-1155­, combinando le capacità di ERC-20 e ERC-720, Witek Radomski (CTO di Enjin) ha introdotto un token standard onnicomprensivo per gli smart contract di Ethereum. Si tratta di un'interfaccia standard che supporta lo sviluppo di token fungibili, semi-fungibili, non fungibili e altre configurazioni con un contatto intelligente comune. Questa permette operazioni multiple in una singola transazione, richiedendo un solo smart contract per infiniti token
  + Ora è possibile soddisfare tutte le esigenze di sviluppo dei token e affrontare i problemi utilizzando un'unica interfaccia, rendendo ERC-1155 un vero e proprio cambiamento. L'idea di questo standard unico per i token è stata quella di sviluppare una robusta interfaccia per smart contract che rappresenti e gestisca diverse forme di token ERC.
  + Un altro aspetto positivo di ERC-1155 è che migliora la funzionalità complessiva dei precedenti standard di token ERC, rendendo l'ecosistema Ethereum più efficiente e scalabile. Questo rappresenta quindi un mezzo intermedio per poter usufruire di tutti i token, fungibili e no.

Immagine che contiene testo

Descrizione generata automaticamente

Riferimento: <https://www.leewayhertz.com/erc-20-vs-erc-721-vs-erc-1155/>

Ulteriori standard da citare:

* ERC-165, uno standard di interfaccia che definisce un metodo per determinare se un contratto supporta determinate funzionalità. Questo standard consente ai contratti di verificare se un altro contratto implementa determinate funzioni e in tal modo consentire loro di interagire con esso in modo appropriato. ERC-165 utilizza l'identificatore dell'interfaccia (Interface ID) per consentire ai contratti di identificare le funzionalità supportate da un altro contratto.
  + Si tratta in realtà di uno standard per un metodo, anziché per i token. Tuttavia, è importante per un altro standard di token, ERC721, che non può essere implementato senza di esso.

Uno smart contract deve interagire con i token crittografici. Anche se tutti i contratti possono interagire con i token ERC 20, è diverso da altri standard ERC per i token, come ERC 721. I contratti intelligenti devono contratti intelligenti devono implementare interfacce specifiche per interagire con i token che seguono altri standard.

* + Ora, la comunità di sviluppo di Ethereum ha bisogno di sapere quali interfacce implementa uno smart contract ed è necessario pubblicare queste informazioni. Questo perché devono sapere in quali modi possono interagire con quel contratto.

Non esisteva un metodo standard per rilevare le interfacce utilizzate da uno smart contract e per pubblicarle. L'ERC 165 standardizza un metodo per questo, oltre a standardizzare l'identificazione delle interfacce.

* ERC-223, che non si tratta di un token digitale onnicomprensivo, ma fornisce una soluzione ad alcuni dei problemi di UX associati ad altri token ERC. In molti casi, le monete vengono inviate all'indirizzo sbagliato del portafoglio o a uno smart contract, con il risultato di essere perse per sempre.
  + Alcuni sviluppatori cinici potrebbero dire che gli utenti finali possono essere incolpati di aver inviato i token all'indirizzo sbagliato, ma questo potrebbe limitare l'accettazione da parte del pubblico. L'interfaccia poco intuitiva può allontanare le persone che non sono tecnicamente preparate.
  + Una caratteristica fondamentale della proposta ERC-223 è che essenzialmente dice agli utenti quando inviano token a indirizzi errati di contratti smart e li annulla. In cambio del pagamento del gas, gli utenti risparmiano il loro Ethereum. In cambio del pagamento della benzina, gli utenti salvano la loro piattaforma Ethereum. Il software offre funzionalità UX di facile utilizzo che proteggono gli utenti dai fallimenti dei fondi, compatibilità con lo standard dei token ERC-20 e costi operativi inferiori a quelli di Ethereum.
  + Gli sviluppatori possono accettare o rifiutare i token che arrivano agli indirizzi dei loro smart contract. Specifica le funzioni che un contratto può codificare in modo che se non può accettare i token, il trasferimento fallisce. Nessun token lo utilizza e gli scambi di criptovalute potrebbero non essere preparati per questo standard.
* ERC-621, estensione dell'ERC 20, questo standard consente di aumentare o diminuire la fornitura totale di token utilizzando due funzioni, "increaseSupply" e "decreaseSupply". La proposta raccomanda che solo il proprietario del contratto o gli utenti fidati possano utilizzarle.
  + Le criptovalute sono denaro matematico, a differenza delle valute fiat in cui le banche centrali possono aumentare o diminuire l'offerta. L'offerta totale di token è uno dei tanti fattori che determinano l'apprezzamento dei token e influisce sull'economia di un token.
  + Questa proposta è una bozza e dobbiamo vedere se la comunità di Ethereum la implementerà, visto l'ampio impatto.
* ERC-725, uno standard di identità basato su Blockchain sviluppato da Fabian Vogelsteller, lo sviluppatore del noto standard ERC-20. Gli smart contract definiti da ERC-725, che definisce i contratti proxy, possono avere più chiavi ed essere associati a diversi smart contract. Individui, macchine e oggetti possono essere definiti con smart contract di identità.
  + La governance e la reputazione decentralizzate sarebbero facilitate da uno standard di identità aperto e portatile. Le persone dovrebbero poter utilizzare la propria identità su più piattaforme e applicazioni.
* ERC-777, standard proposto che include una funzione per identificare la ricezione dei token e avviare uno smart contract subito dopo la prima transazione. Se da un lato riduce l'overhead delle transazioni, dall'altro consente all'utente di rifiutare i token in arrivo da un indirizzo inserito nella blacklist. L'inserimento di un indirizzo nella lista nera può essere dovuto a vari motivi, come hacking o attività illegali. La possibilità di rifiutare il pagamento da un tale indirizzo migliora la posizione di sicurezza di una DApp Ethereum.
  + L'ERC 777 mantiene tutti i punti di forza dell'ERC 20. OpenZeppelin (libreria per lo sviluppo di smart contract sicuri) ha già implementato ERC777 per costruire, automatizzare e gestire applicazioni decentralizzate.
* ERC-827, che permette il trasferimento di token a terzi per spenderli. Si tratta di un'altra estensione dell'ERC 20, ma intende risolvere il problema che l'ERC 223 cerca di risolvere, in modo migliorato. Se questo standard viene implementato, il titolare di un token può trasferire i token e allo stesso tempo approvare che una terza parte li spenda.
  + Con questo standard ERC, i portafogli e gli scambi possono riutilizzare i token, perché entrambe le parti concordano su criteri specifici per la spesa di una terza parte per un importo dinamico. La comunità di Ethereum non l'ha ancora implementato.
* ERC-865, uno standard di interfaccia che definisce un metodo per ridurre i costi di transazione sui token Ethereum. Questo standard consente agli utenti di pagare i costi di transazione in token invece che in Ether, il che può ridurre significativamente i costi di transazione. ERC-865 definisce una serie di funzioni per effettuare pagamenti e consente ai contratti di delegare i costi di transazione agli utenti che effettuano le transazioni.
  + Quando un utente nuovo alle criptovalute cerca di trasferire i token basati su Ethereum, scopre improvvisamente di dover pagare le commissioni dei minatori in Ether. Questo aumenta l'attrito per l'utente.
  + Tra gli standard ERC proposti dagli sviluppatori della comunità di Ethereum, l'ERC 865 intende semplificare la vita dei nuovi utenti di criptovalute.
* ERC-884, uno standard di interfaccia per token Ethereum che consente la creazione di token non fungibili (NFT) che rappresentano il possesso di beni fisici o digitali. Questo standard definisce un insieme di funzioni per creare, trasferire e gestire i NFT, consentendo a diverse applicazioni di interagire con questi token in modo standardizzato.
  + Per rispettare le normative, ERC 884 include anche quanto segue:
    - Verifica dell'identità e whitelisting obbligatorio dei titolari di token;
    - La società può preparare un elenco degli azionisti in base ai requisiti normativi;
    - Registrazione delle informazioni mandato dei regolatori;
    - Registrazione del trasferimento di azioni in base ai requisiti normativi;
    - Solo il valore intero dei token, cioè nessun valore parziale;
    - Gli azionisti che hanno perso le chiavi private o i token devono recuperarli a un nuovo indirizzo.
  + L'implementazione dell'ERC 884 richiede un database fuori dalla catena per vari requisiti "Know Your Customer" (KYC). Si tratta ancora di una bozza di proposta

### Fungible Tokens

I token fungibili sono criptovalute che presentano tutte le caratteristiche standard degli asset fungibili. In altre parole, queste criptovalute dovrebbero essere non uniche, divisibili e avere un chiaro valore di mercato. Non dovrebbe essere difficile per i trader determinare il valore dei loro token fungibili o fare trading con essi sugli scambi di criptovalute.

#### Tipologie

Alcune delle tipologie listate di seguito sono già state citate sopra, pertanto qui sintetizzate:

* Stablecoin: è un tipo di token fungibile che mira a mantenere un valore stabile, spesso agganciato a una valuta tradizionale o a un altro asset stabile. Gli stablecoin possono essere utilizzati come riserva di valore o come mezzo di scambio, senza le fluttuazioni di prezzo tipiche di altre criptovalute.
* Utility token: è un tipo di token fungibile che conferisce al possessore il diritto di accedere a un servizio o a una funzionalità specifica all'interno di una piattaforma o di un'applicazione. Questo tipo di token può essere utilizzato come forma di pagamento all'interno della piattaforma che lo emette.
* Security token: è un tipo di token fungibile che rappresenta un investimento in un'azienda o in un progetto. A differenza degli utility token, i security token sono considerati titoli finanziari e sono soggetti alle normative sui mercati finanziari.
* Asset token: è un tipo di token fungibile che rappresenta la proprietà di un asset fisico o di un bene virtuale. Gli asset token possono rappresentare, ad esempio, l'acquisto di un bene immobiliare o di una quota di una società.
* Commodity token: è un tipo di token fungibile che rappresenta la proprietà di una materia prima, come oro, argento, petrolio, ecc. Questo tipo di token permette agli investitori di acquistare una quota di un bene fisico senza la necessità di possederlo fisicamente.

Ora passiamo ad altre tipologie di token fungibili:

* CDBC (Central Bank Digital Currency), utilizzabile qualora emessa da una banca centrale per accumulare valore ed effettuare pagamenti, vista come valuta fiduciaria affinché possa essere utilizzata (meglio in futuro) come sistema accessibile e disponibile da tutti i cittadini. Esse vengono supportate dai governi del paese, dando la possibilità di effettuare trasferimenti diretti; per contro, non hanno un valore pienamente stabile (dipendente dal mercato, non dal contesto stesso del paese) ed occorre chiaramente un wallet per la gestione delle chiavi e dei propri dati.
* Branded currency, che può essere definita come - una riserva di valore e un mezzo di scambio per beni e servizi di un marchio o di un commerciante specifico; ciò include qualsiasi forma fisica o digitale di pagamento da parte di un commerciante o di un marchio, come coupon, punti fedeltà e carte regalo. Vengono viste come un mezzo di connessione tra utenti reali e brand, sia come mezzo di maggiore connessione verso di lui che come ulteriore trasformazione digitale
  + Le criptovalute di marca cercano di risolvere i problemi di rottura e di valore inutilizzato all'interno dell'economia delle valute di marca tradizionali, eliminando gli attriti e gli altri punti dolenti inerenti all'infrastruttura attuale
  + Tuttavia, alcune valute di marca come i punti e le carte regalo sono ancora considerate pagherò e non sono esplicitamente garantite. Possono corrispondere a un valore in dollari, ma non sono direttamente garantite dai dollari associati.
  + Aziende come Walmart e Facebook stanno esplorando questo settore in quanto l’accesso alle blockchain potrebbe consentire a circa 1.7 miliardi di persone di accedere ai sistemi finanziari, permettendo direttamente agli utenti di utilizzare una valuta considerata come asset facilmente integrabile in molti altri sistemi reali

### NFT (Non Fungible Tokens)

I token non fungibili (NFTs) non possono essere interscambiati in quanto costituiscono la rappresentazione digitale di oggetti unici o rari ed hanno valore di certificato di proprietà digitale. Gli NFT non possono essere scambiati o negoziati come gli altri token crittografici, poiché sono tutti tecnicamente unici, quindi hanno un valore proprio.

* Un NFT può rappresentare una serie di cose sia nel regno fisico che in quello digitale. Arte, musica, oggetti di gioco, biglietti e naturalmente oggetti da collezione sono tutti esempi di ciò che un NFT può rappresentare in forma digitale. Essi possono essere aggiornati in qualsiasi momento, aggiungendo nuove caratteristiche in ogni momento.
* I non fungible-token vengono comunemente utilizzati per la gestione dell’identità digitale, i progetti di tracciabilità (in questo caso si parla di digital twin) e dell’automazione dei processi di supply chain, il voto elettronico, i collezionabili intesi come bene nei giochi online.

Il processo di pubblicazione di un token unico su una blockchain viene definito come minting (conio), generalmente utilizzato per descrivere il processo di creazione di un NFT. Si usa anche per descrivere l'atto di acquistare un token non fungibile direttamente da uno smart contract. Il processo impiega 5 minuti per un singolo NFT fno ad alcune settimane per un intera collezione tramite uno smart contract

* Come creatore, si conia un NFT per renderlo disponibile per la vendita ad altri. In questo modo, si pubblica un singolo NFT oppure una collezione di essi.
* Come consumatore, si può coniare un NFT da un contratto, che viene poi rivelato dopo l'acquisto. Da questo punto di vista, si intende comprare un NFT, permettendo di diventare il possessore unico della stessa

Prima di effettuare il minting, è necessario considerare:

* La piattaforma blockchain opportuna, affinché la supporti e sia popolare, oltre che compatibilità con la stessa. Si realizza un processo definito come *wrapping*, per il quale la NFT ha funzionalità economiche, di prestito, anti-frode e protetto da svalutazione. Questo permette di spostarle tra blockchain diverse.
* I costi di creazione, da un punto di vista di sicurezza, efficienza e funzionalità.

Per effettuarlo, occorre:

* Creare un file digitale che verrà pubblicato sulla blockchain. Questo file può essere un'immagine, un video, un audio o una combinazione di questi. Tuttavia, è bene tenere presente che ci sono limitazioni di upload.
* Comprare alcune cripto, in quanto per creare un asset digitale è necessario utilizzare la criptovaluta per pagare le spese di transazione relative alla blockchain. Ogni blockchain utilizza la propria criptovaluta nativa, pertanto è necessario assicurarsi di acquistare la valuta corretta per la blockchain in uso.
* Usare un wallet di qualsiasi tipo
* Pubblicare l’asset nella blockchain; è possibile crearlo anche gratuitamente, eseguendo il minting solo quando l’oggetto viene acquistato e il compratore pagherà il gas necessario per pubblicarlo su di essa.

L'acquisto di un NFT presso la zecca (at mint) presenta molti vantaggi e svantaggi. A livello di vantaggi:

* Profitto dall'essere tra i primi ad acquistare: Può essere vantaggioso essere tra i primi ad acquistare nuovi token. Infatti, i primi investitori possono ottenere gli NFT al prezzo più basso possibile. Di conseguenza, siete in una posizione eccellente per aumentare il rendimento del vostro investimento in NFT.
* Ottenere un accesso esclusivo alle DAO e altri vantaggi di essere i primi: l'acquisto di un NFT può darvi un vantaggio nell'aspetto della comunità. Ad esempio, i casi in cui gli NFT possono fornire ai possessori di token l'ingresso nella DAO del progetto possono far sì che gli investitori ricevano delle partecipazioni nella direzione in cui il progetto potrebbe andare.

A livello di svantaggi:

* Rischio di diminuzione del valore dopo il conio: Il mercato speculativo è uno dei motivi principali per cui il conio di nuovi NFT comporta un rischio significativo. Non c'è alcuna garanzia che il valore del vostro prodotto aumenti nel tempo. Tra l'altro, potreste anche trovarvi di fronte a circostanze in cui il prezzo scende vertiginosamente subito dopo il conio.
* Elevate commissioni di gas: Ci sono momenti in cui il prezzo d'ingresso potrebbe salire notevolmente a causa dell'aumento dell'attività della rete. A volte, ci possono essere troppe persone contemporaneamente che cercano di ottenere un posto nella zecca. Di conseguenza, la tassa sul gas può aumentare rapidamente.

Il processo di distruzione è definito come burning e, come già visto, si intende inviare il token ad un indirizzo non spendibile che nessuno può accedere, pagando una tassa di *gas*.

La principale classificazione dei tipi di NFT si riferisce alle categorie generali. I tre tipi comuni di NFT comprendono:

* Originale o copia di un'opera, documentata su una rete blockchain o DLT
* NFT nativi digitali, che hanno diritti di proprietà sull'opera che li costituisce
* metadati NFT, che prevede che l'NFT fornisca la rappresentazione della proprietà per i file di metadati relativi a Internet.

I tipi comuni di token non fungibili offrono un'ampia descrizione dei criteri utilizzati nella classificazione degli NFT. Nel caso degli NFT originali, essi sono creati su una rete blockchain e gli NFT rimangono sulla rete blockchain. I NFT nativi digitali sono i beni che comportano l'emissione di NFT a più persone con diritti di proprietà del bene. I metadati dell'NFT sono un'altra indicazione importante per la classificazione dei token non fungibili, in quanto si tratta essenzialmente di un collegamento ai metadati dell'NFT. Di conseguenza, non si ottiene la proprietà dell'NFT, ma solo i diritti di utilizzo.

#### Tipologie

Le voci di spicco di un elenco di token non fungibili sono le seguenti:

* Oggetti da collezione (collectibles)
  + Il principale esempio di NFT è emerso con lo sviluppo dei Cryptokitties, che sono oggetti da collezione online. Di fatto, i Cryptokitties sono il primo caso di utilizzo di NFT. È interessante notare che nel 2017 i Cryptokitties sono diventati così popolari da congestionare la rete Ethereum. I Cryptokitties sono una delle aggiunte più sorprendenti alla lista dei token non fungibili nella categoria degli oggetti digitali da collezione. Si tratta fondamentalmente di gattini digitali con caratteristiche distinte che li rendono più popolari e favorevoli di altri.
* Opere d'arte (artworks)
  + Anche le opere d'arte sono un altro candidato importante per i NFT. I tipi comuni di gettoni non fungibili in questo settore si riferiscono all'arte programmabile, che presenta una combinazione unica di creatività e tecnologia. Attualmente sono in circolazione molte opere d'arte in edizione limitata che possono essere programmate a determinate condizioni. L'uso di oracoli e contratti intelligenti potrebbe aiutare gli artisti a creare immagini rappresentate su reti blockchain. Le NFT per l'arte digitale hanno anche incoraggiato le possibilità di partecipazione dell'industria artistica tradizionale.
  + La tokenizzazione di beni e opere d'arte del mondo reale potrebbe contribuire a incoraggiare l'adozione delle NFT. La possibilità di combinare blockchain e IoT potrebbe offrire prospettive interessanti per la scansione di un codice o di un adesivo sui beni. I tipi di NFT nelle opere d'arte potrebbero garantire che gli utenti possano facilmente registrare la proprietà di opere d'arte reali su una rete blockchain. Successivamente, gli utenti potrebbero anche scoprire la storia completa di un'opera d'arte, come i precedenti proprietari e i prezzi per cui è stata venduta in passato.
* Biglietti per eventi
  + Un'altra promettente aggiunta tra i tipi di NFT riguarda i biglietti per eventi. Questi tipi di NFT consentono alle persone che partecipano a eventi come festival musicali e concerti di verificare la loro identità e i loro biglietti. I gestori degli eventi potrebbero coniare un numero specifico di biglietti NFT su una piattaforma blockchain selezionata. I clienti potrebbero acquistare i biglietti attraverso un'asta e conservarli nei loro portafogli, facilmente accessibili attraverso i dispositivi mobili.
* Musica e media
  + Anche il settore della musica e dei media sta sperimentando le NFT, dando vita a un'altra categoria di NFT. È possibile collegare i file musicali e multimediali agli NFT, consentendo così a un individuo con un vero diritto di proprietà di accedere ai file. Le due principali piattaforme che aiutano gli artisti a coniare le loro canzoni come NFT sono Rarible e Mintbase.
  + Mentre gli artisti ottengono il vantaggio di raggiungere direttamente i propri follower e il nuovo pubblico, gli ascoltatori ottengono un'esperienza premium. Il senso di esclusività nell'acquisto di musica NFT è uno dei motivi principali per cui i dischi in vinile vintage sono stati infusi. La crescita dei NFT musicali nell'elenco dei gettoni non fungibili potrebbe offrire prospettive affidabili per affrontare i problemi della pirateria musicale e degli intermediari.
* Videogiochi (Gaming)
  + I tipi comuni di gettoni non fungibili nel settore dei giochi si concentrano principalmente sugli oggetti di gioco. I NFT hanno suscitato un grande interesse tra gli sviluppatori di giochi. Possono offrire la funzionalità di record di proprietà per gli oggetti di gioco, favorendo così la crescita delle economie di gioco. Ma soprattutto, gli NFT nel settore dei giochi si concentrano sull'introduzione di un'ampia gamma di vantaggi per i giocatori.
  + Mentre gli oggetti da collezione in-game erano requisiti comuni per una migliore esperienza di gioco, gli NFT hanno il potenziale per cambiare il loro valore. Gli oggetti di gioco come NFT potrebbero facilmente aiutare a recuperare denaro vendendoli al di fuori del gioco. D'altra parte, gli sviluppatori di giochi o i creatori che emettono NFT potrebbero guadagnare una royalty per ogni vendita di oggetti nel mercato aperto.
* Asset del mondo reale
  + Anche se non è possibile trovare molti tipi di NFT che fungono da gettoni per oggetti del mondo reale, i progressi nel campo degli NFT possono far sì che ciò accada. Ad esempio, molti progetti NFT si stanno attualmente concentrando sulla tokenizzazione di beni immobili e di beni di lusso. Gli NFT sono fondamentalmente atti di proprietà e possono introdurre la flessibilità di acquistare un'auto o una casa con un atto di proprietà NFT. Pertanto, gli NFT che rappresentano beni del mondo reale possono sfruttare le opportunità offerte dalla prova crittografica della proprietà.
* Identità
  + Una delle caratteristiche fondamentali dei token non fungibili è la scarsità. Ogni NFT è unico e non può essere scambiato con nessun altro token. Nel caso degli NFT di identità, il funzionamento è simile a quello degli NFT di biglietti per eventi. Possono fungere da identificatori unici e quindi da supporto affidabile per i sistemi di gestione dell'identità.
  + Le applicazioni comuni degli NFT basati sull'identità possono essere evidenti nelle certificazioni e nelle licenze. La creazione di certificazioni e licenze e di NFT per dimostrare e verificare i dati di un individuo potrebbe cambiare il settore della gestione dell'identità. Inoltre, gli NFT basati sull'identità potrebbero anche garantire che gli individui possano conservare le prove della loro identità senza rischiare di perderle.
* Memes
  + Il progresso più significativo nel settore delle NFT negli ultimi tempi è la vendita di meme come NFT. Oltre a far parte della cultura popolare e a essere immediatamente preferiti dagli utenti di Internet, i meme sono stati associati alle NFT. La vendita di meme come NFT mostra il potenziale dei creatori di meme unici di partecipare a un ecosistema futuristico in evoluzione.
* Nomi di dominio
  + Questi ultimi sono diventati popolari di recente. I servizi di nomi di dominio decentralizzati, come Unstoppable Domains e Ethereum Name Service (ENS), sono i principali esempi di NFT di nomi di dominio. ENS può aiutare a tradurre indirizzi utente lunghi e complessi in un'esperienza flessibile e amichevole per gli utenti, con una curva di apprendimento più semplice

## Tokenizzazione

La tokenizzazione è il processo di rappresentazione di un bene o di un'attività in forma digitale attraverso l'emissione di un token su una blockchain. In altre parole, si tratta di convertire un bene fisico o immateriale in un asset digitale che può essere negoziato e scambiato in modo decentralizzato.

* Fondamentalmente, i token blockchain forniscono una rappresentazione digitale della proprietà completa o condivisa di qualsiasi entità con un valore specifico. Le applicazioni comuni dei token blockchain sono evidenti nei pagamenti e nel regolamento delle transazioni tra i partecipanti. I token rappresentano anche la proprietà di più parti per beni indivisibili come opere d'arte o video musicali. Inoltre, i token consentono un più facile scambio di proprietà di beni indivisibili attraverso una rete blockchain.
* La combinazione tokenizzazione - blockchain potrebbe aprire nuove prospettive per l'ottimizzazione dei processi aziendali, che includono più partner, e l'introduzione di nuovi modelli di business. Secondo IDC, il mercato globale degli asset tokenizzati raggiungerà una valutazione di 500 miliardi di dollari. Sebbene si possano notare promettenti opportunità per la tokenizzazione di asset sulla blockchain, il concetto di token non è nato qui.
* Infatti, i token sono stati utilizzati come meccanismo unico di sicurezza dei dati nei servizi finanziari per salvaguardare informazioni riservate come numeri di carte di credito, informazioni di identificazione personale e rendiconti finanziari.

Esistono diversi tipi di asset da considerare, i quali saranno poi successivamente convertiti in token:

* *Beni immateriali (intangibles)*
  + I beni immateriali esistono solo in base a precedenti legali, senza alcun oggetto fisico che li rappresenti. Esempi di beni immateriali sono i diritti d'autore e i brevetti. Quando si desidera tokenizzare i beni immateriali, è importante assicurarsi che il modello di trasferimento dei beni della rete blockchain sia uguale a quello del mondo reale. I beni immateriali sono facili da convertire in token senza problemi di stoccaggio o spedizione. Al contrario, le differenze giurisdizionali potrebbero creare profonde difficoltà per il trasferimento di token che rappresentano beni immateriali.
* *Fungible Assets (asset fungibili)*
  + I beni fungibili sono quelli che possono essere sostituiti da un bene simile. Gli esempi più comuni di beni fungibili sono il grano o l'oro. Convertire i beni fungibili in gettoni è più facile, in quanto è possibile dividerli facilmente in unità più piccole. Inoltre, un gettone può servire come rappresentante di un gruppo di beni fungibili, come un mucchio d'oro. L'algoritmo di tokenizzazione per gli asset fungibili deve anche includere un livello di astrazione. Inoltre, un insieme di token è correlato a una collezione di componenti di asset intercambiabili.
* Un asset fungibile ha due caratteristiche:
  + È intercambiabile, in quanto ogni unità ha lo stesso valore di mercato e validità
  + È divisibile, in quanto può essere suddivisa nel numero di cifre decimali che sono state configurate al momento della sua emissione. Ogni unità avrà lo stesso valore e la stessa validità.
* *Non-fungible Assets (asset non fungibili)*
  + Gli asset non fungibili sono quelli che non possono essere scomposti in pezzi più piccoli. La tokenizzazione aiuta a scomporre gli asset non fungibili in azioni digitali, che possono essere scambiate completamente o in modo limitato. L'esempio di tokenizzazione di opere d'arte e immobili mostra le possibilità di conversione di beni non fungibili in token.
  + La conversione di opere d'arte in token inizia con l'introduzione di una firma digitale immutabile. La firma digitale o token fornisce una rappresentazione dell'opera d'arte mantenendo l'unicità. Successivamente, è possibile suddividere il token in sub-token più piccoli che possono essere firmati digitalmente. È quindi possibile vendere i token come azioni dell'opera d'arte originale.
* Un asset non fungibile ha tre caratteristiche:
  + Non intercambiabili: Gli NFT non possono essere sostituiti con token dello stesso tipo, perché ogni token rappresenta un valore unico.
  + Non divisibili: Gli NFT non sono tipicamente divisibili, anche se gli F-NFT (fractional NFT, un inter NFT che è stato diviso in frazioni più piccole permettendo a varie persone di prendere proprietà dello stesso NFT. Questi offrono la proprietà frazionaria degli NFT, come nel caso di costose opere d'arte o immobili commerciali.).
  + Unico: Ogni token si differenzia da un altro token dello stesso tipo e possiede informazioni e attributi unici.

### Cosa tokenizzare, tipi e benefici

Le possibilità sono infinite, poiché la tokenizzazione consente sia la proprietà frazionaria che la prova di proprietà. Dai beni tradizionali come i fondi di capitale di rischio, le obbligazioni, le materie prime e le proprietà immobiliari ai beni esotici come le squadre sportive, i cavalli da corsa, le opere d'arte e le celebrità, le aziende di tutto il mondo utilizzano la tecnologia blockchain per tokenizzare quasi tutto. Tuttavia, si possono distinguere quattro categorie principali:

* Asset: qualsiasi oggetto di valore che può essere trasformato in denaro. Si divide in due classi: personale e aziendale. Gli asset personali possono includere denaro e proprietà. Gli asset aziendali comprendono le attività presenti nel bilancio.
* Patrimonio netto (Equity): I titoli azionari (shares) possono essere tokenizzati; tuttavia, le attività rimangono sotto forma digitale di token di sicurezza memorizzati online in un portafoglio. Gli investitori possono tipicamente acquistare azioni in borsa.
* Fondi: Un fondo di investimento è un tipo di attività che gli investitori possono tokenizzare: questi token rappresentano la quota del fondo detenuta dagli investitori. Ogni investitore riceve dei token che rappresentano la sua quota del fondo.
* Servizi: Un'azienda può offrire beni o servizi come mezzo per raccogliere fondi o condurre affari. Gli investitori possono utilizzare i token per acquistare i beni o i servizi forniti dal fornitore.

I tipi di tokenizzazione della blockchain includono:

* Tokenizzazione fungibile. Si tratta di gettoni blockchain standard. Hanno valori identici, quindi possono facilmente sostituirsi l'un l'altro - si pensi a scambiare un dollaro con un altro dollaro.
* Tokenizzazione non fungibile. Si tratta di gettoni blockchain meno comuni che non hanno un valore prestabilito. Rappresentano invece la proprietà di un bene, come un'arte digitale o un immobile, che determina il valore del token.
* Tokenificazione della governance. Questi token rappresentano diritti di voto e possono essere utilizzati per votare e collaborare su un sistema blockchain.
* Tokenizzazione di utilità. Questi token sono utilizzati per dare accesso a determinati prodotti e servizi su una specifica blockchain, quindi possono essere utilizzati per completare azioni come il pagamento delle commissioni di transazione o il funzionamento di un sistema di mercato decentralizzato.

I tipi di tokenizzazione non-blockchain includono:

* Tokenizzazione del caveau. È il tipo standard di tokenizzazione per proteggere le informazioni di pagamento, in cui il token viene utilizzato per elaborare i pagamenti senza fornire numeri di carta o altri dati.
* Tokenizzazione senza caveau. È un tipo di tokenizzazione utilizzata per l'elaborazione dei pagamenti che non richiede un caveau per l'archiviazione dei token. Utilizza invece dispositivi e algoritmi crittografici per convertire i dati in token.
* Tokenizzazione con elaborazione del linguaggio naturale. Questo tipo di tokenizzazione scompone le informazioni in termini più semplici per renderle più comprensibili ai computer. Comprende la tokenizzazione di parole, sottoparole e caratteri.

Vi sono alcuni vantaggi della tokenizzazione degli asset dal punto di vista del proprietario di un bene:

* Aumento della liquidità
  + Facciamo l'esempio di una persona che richiede 50.000 dollari per un appartamento del valore di 500.000 dollari. Questo individuo potrebbe aver tokenizzato il suo appartamento in 500.000 token di sicurezza, ognuno dei quali vale lo 0,0002%. Potrebbe vendere 50.000 token, invece di vendere l'intera proprietà e perdere la sua utilità come spazio abitabile, assicurandosi così un bene più liquido.
* Prezzi equi
  + Le attività che non possono essere liquidate hanno spesso un prezzo di mercato non stabilito. In questo caso, i proprietari degli asset forniscono agli acquirenti incentivi come sconti per illiquidità, che riducono il prezzo dell'asset. La tokenizzazione degli asset aumenterebbe la liquidità di un asset, in quanto facilita la proprietà frazionata, che elimina gli sconti di illiquidità. Inoltre, la vendita di piccole frazioni di proprietà consente ai proprietari di applicare un prezzo di mercato equo.
* Riduzione dei costi di gestione
  + Se oggi si trasferisce la proprietà di un bene, è necessario l'intervento di avvocati che fungano da intermediari per gestire le pratiche burocratiche e creare fiducia tra il proprietario e l'acquirente, con conseguenti tempi e costi aggiuntivi. Se si sceglie di tokenizzare lo stesso bene e di utilizzare una piattaforma o un mercato decentralizzato, si automatizzano molte parti di questo processo, risparmiando tempo e costi.

Similmente, vi sono alcuni vantaggi della tokenizzazione degli asset dal punto di vista degli investitori:

* Aumento della liquidità
  + Considerando l'esempio precedente di una proprietà tokenizzata, è ora possibile per gli investitori al dettaglio investire piccole somme di denaro in una proprietà. Gli investitori hanno la possibilità di diversificare il loro portafoglio investendo, ad esempio, una somma di 10.000 dollari - cosa che storicamente non era possibile senza un sacco di scartoffie, che costano denaro e richiedono più tempo. Gli investitori beneficiano della maggiore liquidità degli asset grazie alla tokenizzazione.
* Periodi di blocco più brevi
  + I periodi di lock-up impediscono agli investitori di vendere i propri asset. A volte ciò è dovuto al fatto che l'asset è grande e illiquido. La tokenizzazione degli asset ha il potenziale di accorciare il periodo di lock-up, in quanto gli investitori possono vendere facilmente i loro token in un mercato liquido. In questo scenario, gli investitori non devono più aspettare anni per ottenere profitti o perdite.
* Processo trasparente
  + Poiché la tecnologia alla base della tokenizzazione degli asset, la blockchain, è immutabile, i proprietari non possono modificare la storia di un asset per farlo apparire più interessante. Ciò consente agli investitori di vedere la storia di una partecipazione e di prendere decisioni più informate.
* Identità sicura
  + Con i dati relativi alla proprietà e all'identità decentralizzata (DID) conservati sulla blockchain, la coppia di chiavi private e pubbliche dell'acquirente forma una firma digitale che garantisce che si tratti davvero di lui. Inoltre, ci sono identificatori DID decisi da organizzazioni di standard, come il w3c, che garantiscono l'accettazione in molte reti e piattaforme diverse.

Comunque, in linea di massima, possiamo evidenziare certamente:

* Maggiore accessibilità
  + L'accessibilità è anche uno dei principali vantaggi della tokenizzazione degli asset su blockchain.
  + L'adozione della tokenizzazione degli asset su blockchain consente di frammentare gli asset al minimo possibile sotto forma di token e incoraggia gli investitori a ottenere una piccola frazione di azioni. Questo apre le porte a diversi investitori e riduce il periodo e l'importo minimo di investimento.
* Assenza di intermediari e transazioni automatizzate e senza intermediari, in quanto eseguito tramite smart contract

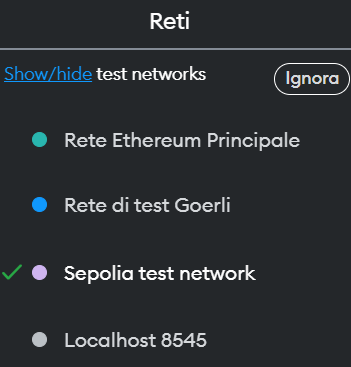
### Ambiti applicativi

Sono molti i settori che stanno abbracciando il concetto di tokenizzazione degli asset:

* Finanza
  + La tokenizzazione, una delle applicazioni della blockchain nel settore fintech, sta cambiando il panorama in diversi modi, che si tratti di prestiti a margine, strutturazione di prodotti, investimenti o pagamenti.
  + Questo concetto offre alle organizzazioni finanziarie l'opportunità di trasformare tutti gli asset in criptovalute digitali che possono essere scambiate senza soluzione di continuità. Offre ai commercianti la possibilità di evitare di memorizzare i numeri delle carte di credito nelle macchine POS e in altri sistemi. Questo, nel complesso, introduce liquidità nel mercato e riduce le violazioni della sicurezza dei dati.
  + Allo stesso modo, la tokenizzazione delle azioni fornisce agli utenti più di un token emesso per la stessa carta di credito. In altre parole, anche se si utilizza un token su un sito web online e il portale viene violato, non sarà più facile decodificare il codice e conoscere il numero effettivo della carta di credito.
  + Un esempio reale di questa tendenza emergente nelle strategie di investimento è la partnership di AlgoZ con OmiseGo.
* Immobiliare
  + Il settore immobiliare è un altro ambito di attività che sfrutta i maggiori vantaggi dello sviluppo di asset tokenizzati.
  + Il concetto di tokenizzazione immobiliare semplifica il processo di investimento. Elimina gli intermediari, rendendo più facile e conveniente l'interazione tra acquirenti e venditori. Inoltre, consente a chiunque di investire qualsiasi importo, il che si traduce in un mercato migliore per tutti.
  + Inoltre, questo concetto riduce il rischio di frode. Per questo motivo, 66 milioni di edifici saranno presto tokenizzati su Ethereum Blockchain in un accordo record.
  + Quando si parla di tokenizzazione immobiliare, ci sono diverse piattaforme che offrono servizi eccezionali in questa direzione. Alcune di esse sono Harbor, Slice e Meridio.
* Assistenza sanitaria
  + Anche il settore sanitario si sta orientando verso l'idea di adottare la tokenizzazione per risolvere alcune delle principali sfide prevalenti al giorno d'oggi.
  + La tokenizzazione sostituisce i dati sensibili dei pazienti, come informazioni sanitarie private e protette elettronicamente, con valori unici e non sensibili, riducendo così i casi di violazione dei dati. Inoltre, trasferisce il potere di creare, accedere e condividere i dati sensibili dagli intermediari come le compagnie assicurative ai pazienti e alle organizzazioni mediche. In questo modo, la tokenizzazione dei processi e delle informazioni sanitarie consente ai pazienti di convalidare l'accuratezza dei propri dati, risparmiando al contempo un'ingente somma offerta a queste terze parti.
  + Clincoin è un esempio perfetto per comprendere la portata della tokenizzazione e della token economy nel settore sanitario. Questa piattaforma basata su Blockchain, oltre a mettere in contatto utenti, fornitori e sviluppatori, premia gli utenti che si impegnano in attività salutari. Gli utenti possono poi utilizzare questi token per acquistare strumenti, prodotti e servizi digitali nel mercato decentralizzato.
* Sport
  + Un altro settore che sperimenta significativi cambiamenti positivi con l'emergere della tokenizzazione degli asset è quello dello sport.
  + La tokenizzazione degli asset sulle piattaforme blockchain decentralizza l'intero mercato, rendendo più facile per gli investitori e i fan investire nei loro giocatori e club sportivi preferiti scambiando i benefici ottenuti. Questo aiuta ulteriormente le società sportive e i giocatori a soddisfare le loro esigenze finanziarie e a operare in questo campo in modo più efficace e redditizio.
* Imprese
  + Le imprese sfruttano anche il potenziale della tokenizzazione degli asset del mondo reale nel suo monumentale passaggio alla Blockchain. Considerano questo concetto per estendere il loro approccio a nuovi mercati, valutare le prestazioni dei dipendenti, garantire una corretta allocazione delle risorse, introdurre migliori modelli di incentivazione e aggiungere trasparenza a tutti i processi interni.
  + Inoltre, le imprese introducono token diversi per i vari tipi di incarico, per offrire un'esperienza personalizzata. Stabiliscono un premio per una particolare quantità di gettoni, che aiuta a mostrare la loro importanza in termini concreti. Qualcosa che va oltre le etichette "ad alta priorità" e "urgente", che i product manager usavano in precedenza.
* Industria dell'arte
  + La blockchain può rendere l'arte più accessibile non solo agli amanti dell'arte, ma anche agli artisti, che possono ora tokenizzare le loro opere e venderle in tutto il mondo senza intermediari.
  + Uno degli esempi recenti che ha fatto notizia è stato il successo della tokenizzazione della prima opera d'arte multimilionaria. Si tratta di un dipinto di Andy Warhol chiamato "14 Small Electric Chairs (1980)". Lo scopo principale dell'asta beta era quello di testare il processo d'asta olandese e in generale verificare il funzionamento della tokenizzazione delle opere d'arte e della tecnologia blockchain.

### Come creare un token

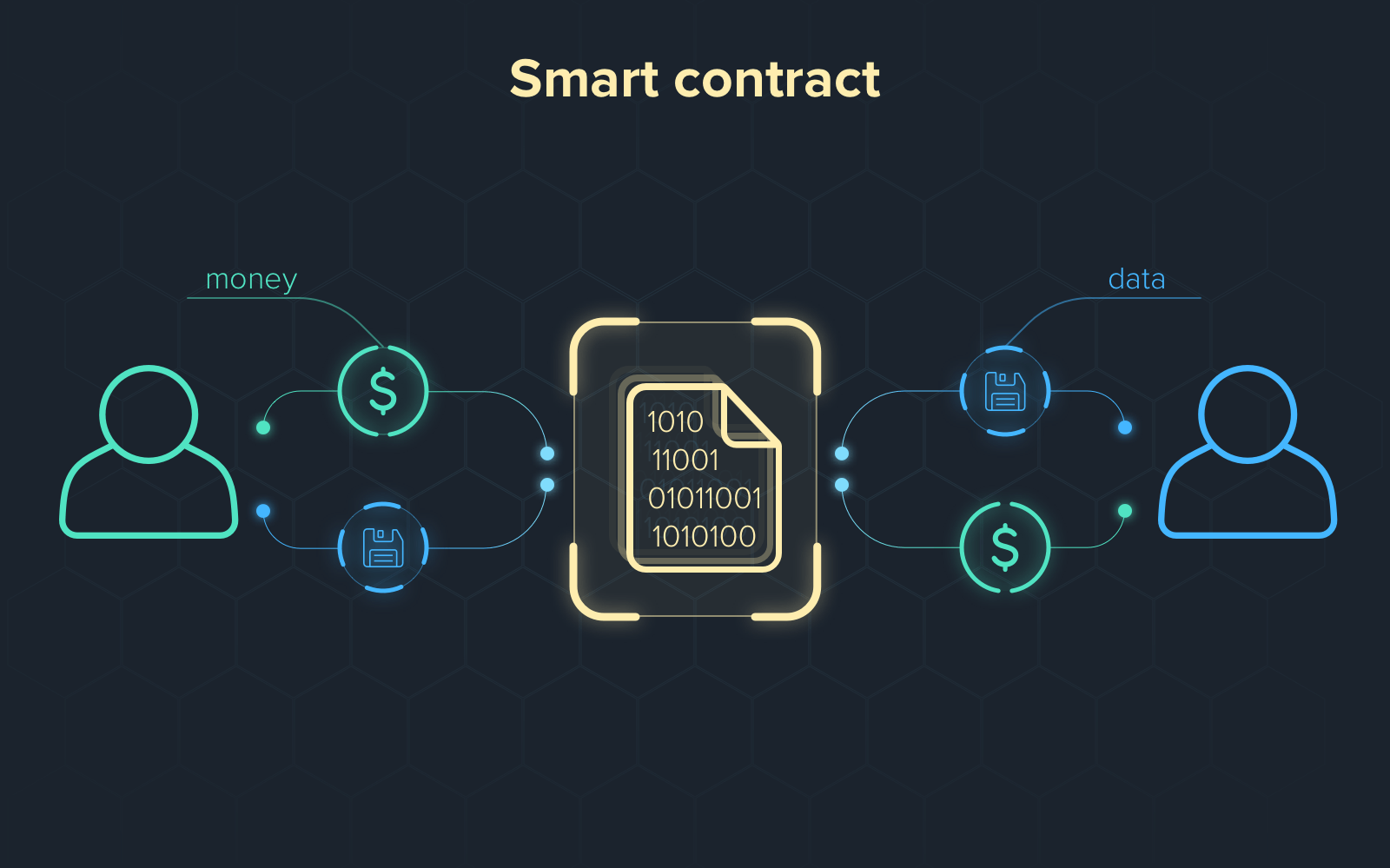
* Decidere quali problemi aziendali risolverà il token e a quale pubblico sarà rivolta
  + Molte persone che si avvicinano al settore delle criptovalute non capiscono cosa stia succedendo nella tecnologia blockchain e vogliono semplicemente guadagnare vendendo token.
  + Ad esempio, se parliamo di ICO, la maggior parte di esse è stata creata per fare soldi dal nulla. I loro creatori non sanno quanti utenti utilizzeranno i loro progetti o se ci sarà una domanda per i token emessi durante le ICO. Di conseguenza, molti team falliscono dopo il lancio a causa della mancanza di domanda per i loro servizi o prodotti.
  + Per questo motivo, il primo passo per la creazione di un token crittografico è decidere quali problemi aziendali risolverà il token, perché ne ha bisogno e se ne ha bisogno. Questo punto è importante per definire ulteriormente le caratteristiche del token e quindi per preparare un piano di progetto.
* Usare un wallet come MetaMask, CoinBase o similari, al fine di crearsi un account, associarsi una frase segreta e avere un posto dove depositare le valute per fare prove



* Usare i cosiddetti *faucet*, cioè dei siti che danno ricompense sulle cosiddette testnet (reti di test attivabili nelle impostazioni, come le seguenti:
* Creare uno smart contract di esempio, che definisca il token.
  + Un esempio può essere anche qui:
    - <https://levelup.gitconnected.com/how-to-create-a-cryptocurrency-token-15a898e2bb8d>
  + In questo definire il numero di token, decimali, il nome e l’abbreviazione

## Smart contract

Gli smart contract sono programmi che automatizzano le azioni richieste in un accordo o contratto, considerate tracciate e irreversibili. Essi sono programmi informatici auto-eseguibili che vengono eseguiti su una blockchain. In pratica, sono contratti digitali che automatizzano ed eseguono le clausole contrattuali in modo sicuro, trasparente e immutabile, senza la necessità di intermediari.



Riferimento: <https://www.arlawpractice.com/gli-smart-contract-cosa-sono-e-come-funzionano/>

* Possono essere considerati programmi event-driven che girano su un libro mastro decentralizzato, distribuito, condiviso e replicato che possono prendere in custodia e istruire il trasferimento di beni su quel libro mastro.
* Poiché i contratti intelligenti eseguono accordi, possono essere utilizzati per molti scopi diversi. Uno degli usi più semplici è quello di garantire che le transazioni tra due parti avvengano, come l'acquisto e la consegna di merci. Ad esempio, un produttore che ha bisogno di materie prime può impostare i pagamenti utilizzando gli smart contract e il fornitore può impostare le spedizioni. Poi, a seconda dell'accordo tra le due aziende, i fondi potrebbero essere trasferiti automaticamente al fornitore al momento della spedizione o della consegna.

L'esecuzione della transazione non prevede l'intervento di alcun intermediario. I partecipanti devono

firmare crittograficamente la loro partecipazione.

- I contratti intelligenti dispongono di interfacce di input che ricevono i dati in ingresso.

- I dati vengono interpretati e convalidati secondo le regole programmate.

- Le firme delle parti coinvolte nella transazione vengono verificate.

Nell'insieme degli smart contract di una blockchain, uno smart contract può agire in modo isolato dagli altri o interagire con essi per completare i compiti affidati. In questo modo, è possibile progettare suite di smart contract per scopi specifici.

Così, uno smart contract può concentrarsi sulla ricezione di dati dall'esterno, mentre un altro smart contract può concentrarsi sull'elaborazione di regole aziendali. Infine, un altro smart contract può occuparsi di dare persistenza al risultato (gli smart contract elaborano le loro istruzioni e alla fine producono un risultato da memorizzare nella blockchain).

* Gli scambi effettuati tramite smart contract sono definite DEX (decentralized exchanges) che consentono agli utenti di negoziare criptovalute e altri asset digitali senza la necessità di un intermediario centralizzato, come una borsa tradizionale o un broker. I DEX sono stati creati per eliminare la necessità di un'autorità che supervisionasse e autorizzasse le transazioni. A differenza delle borse centralizzate, i DEX non custodiscono i fondi degli utenti, che sono responsabili della loro perdita in caso di errore.
  + In un DEX, i contratti intelligenti vengono utilizzati per eseguire automaticamente le transazioni tra acquirenti e venditori. I contratti intelligenti fungono da "intermediari" e assicurano che le transazioni siano eseguite in modo sicuro e trasparente. Questi contratti sono auto-esecutivi e auto-applicativi, il che significa che eseguiranno automaticamente la compravendita non appena le condizioni stabilite nel contratto saranno soddisfatte.
  + I DEX possono essere progettati in modi diversi, ma in genere prevedono tutti l'uso di contratti intelligenti. Ad esempio, alcuni DEX utilizzano un modello di market maker automatizzato (AMM), in cui il prezzo di un'attività è determinato da un algoritmo che bilancia la domanda e l'offerta dell'attività. Altri utilizzano un modello tradizionale di portafoglio ordini, in cui acquirenti e venditori possono piazzare ordini limite o di mercato.
  + Ci sono tre tipi principali di DEX:
    - *Order books DEX*, che raccolgono le registrazioni di tutti gli ordini aperti di acquisto e vendita di asset per coppie di asset specifiche. Gli ordini di acquisto indicano che un trader è disposto ad acquistare o fare un'offerta per un asset a un prezzo specifico, mentre gli ordini di vendita indicano che un trader è pronto a vendere o chiedere un determinato prezzo per l'asset in questione. Lo spread tra questi prezzi determina la profondità del portafoglio ordini e il prezzo di mercato della borsa.
      * I DEX con order book sono di due tipi: order book on-chain e order book off-chain. I DEX che utilizzano gli order book spesso conservano le informazioni sugli ordini aperti sulla catena, mentre i fondi degli utenti rimangono nei loro portafogli. Queste borse possono consentire ai trader di fare leva sulle loro posizioni utilizzando fondi presi in prestito da prestatori sulla loro piattaforma. Il trading con leva aumenta il potenziale di guadagno di un'operazione, ma aumenta anche il rischio di liquidazione, poiché aumenta la dimensione della posizione con i fondi presi in prestito, che devono essere rimborsati anche se i trader perdono la loro scommessa.
      * Tuttavia, le piattaforme DEX che tengono i loro order book fuori dalla blockchain regolano le transazioni solo sulla blockchain per portare ai trader i vantaggi delle borse centralizzate. L'uso di libri degli ordini fuori dalla catena aiuta le borse a ridurre i costi e ad aumentare la velocità per garantire che le transazioni siano eseguite ai prezzi desiderati dagli utenti.
      * È importante sottolineare che i DEX con order book spesso soffrono di problemi di liquidità. Dato che sono essenzialmente in concorrenza con le borse centralizzate e che devono sostenere commissioni aggiuntive rispetto a quelle che si pagano per le transazioni on-chain, i trader di solito si attengono alle piattaforme centralizzate. Mentre i DEX con order book off-chain riducono questi costi, i rischi legati agli smart contract sorgono a causa della necessità di depositare fondi in essi.
    - *Automated Market Makers (AMM)*, che risolvono il problema della liquidità e si affidano a servizi basati sulla blockchain che forniscono informazioni dalle borse e da altre piattaforme per stabilire il prezzo degli asset scambiati, chiamati oracoli della blockchain. Invece di far coincidere gli ordini di acquisto con quelli di vendita, gli smart contract di queste borse decentralizzate utilizzano pool di asset prefinanziati noti come pool di liquidità.
      * I pool sono finanziati da altri utenti che hanno diritto alle commissioni di transazione che il protocollo addebita per l'esecuzione degli scambi su quella coppia. Questi fornitori di liquidità devono depositare un valore equivalente di ogni asset nella coppia di trading per guadagnare interessi sulle loro criptovalute, un processo noto come liquidity mining. Se tentano di depositare una quantità maggiore di un asset rispetto all'altro, lo smart contract alla base del pool invalida la transazione.
      * L'uso dei pool di liquidità consente ai trader di eseguire ordini o di guadagnare interessi in un modo privo di autorizzazioni e di fiducia. Queste borse sono spesso classificate in base alla quantità di fondi bloccati nei loro smart contract, chiamata valore totale bloccato (TVL), poiché il modello AMM ha un lato negativo quando non c'è abbastanza liquidità: lo slippage.
      * Lo slippage si verifica quando la mancanza di liquidità sulla piattaforma fa sì che l'acquirente paghi prezzi superiori a quelli di mercato per il suo ordine, con ordini più grandi che subiscono uno slippage maggiore. La mancanza di liquidità può dissuadere i trader più facoltosi dall'utilizzare queste piattaforme, in quanto gli ordini di grandi dimensioni rischiano di subire uno slippage in assenza di liquidità profonda.
      * I fornitori di liquidità devono inoltre affrontare vari rischi, tra cui la perdita impermanente, che è il risultato diretto del deposito di due asset per una specifica coppia di trading. Quando uno di questi asset è più volatile dell'altro, gli scambi in borsa possono ridurre la quantità di un asset nel pool di liquidità.
      * Se il prezzo dell'asset altamente volatile aumenta mentre la quantità di liquidità detenuta dai fornitori di liquidità diminuisce, questi ultimi subiscono una perdita impermanente. La perdita è impermanente perché il prezzo dell'attività può ancora risalire e gli scambi in borsa possono bilanciare il rapporto di coppia. Il rapporto di coppia descrive la proporzione di ciascun asset detenuto nel pool di liquidità. Inoltre, le commissioni incassate dal trading possono compensare la perdita nel tempo.
    - *DEX Aggregators*, che utilizzano diversi protocolli e meccanismi per risolvere i problemi associati alla liquidità. Queste piattaforme aggregano essenzialmente la liquidità di diversi DEX per minimizzare lo slippage sugli ordini di grandi dimensioni, ottimizzare le commissioni di swap e i prezzi dei token e offrire ai trader il miglior prezzo possibile nel minor tempo possibile.
      * Proteggere gli utenti dall'effetto prezzo e diminuire la probabilità di transazioni non andate a buon fine sono altri due obiettivi significativi degli aggregatori DEX. Alcuni aggregatori DEX utilizzano anche la liquidità delle piattaforme centralizzate per offrire agli utenti un'esperienza migliore, pur rimanendo non depositari grazie all'integrazione con specifiche borse centralizzate.

A livello di vantaggi:

* Decentralizzazione: I DEX operano su una rete decentralizzata e non si affidano a intermediari, come le autorità centrali, per eseguire le transazioni. Questo riduce il rischio di censura, manipolazione e furto.
* Privacy: Gli utenti non devono rivelare la propria identità o le proprie informazioni personali per utilizzare i DEX. Questo garantisce una maggiore privacy e anonimato.
* Sicurezza: I DEX sono più sicuri delle borse centralizzate poiché utilizzano contratti intelligenti e la tecnologia blockchain per eseguire le transazioni. Questo rende difficile per gli hacker manipolare il sistema e rubare i fondi.
* Accesso a una gamma più ampia di token: I DEX offrono l'accesso a una gamma più ampia di token, compresi quelli che non sono disponibili nelle borse centralizzate.
* Commissioni più basse: I DEX hanno spesso commissioni più basse rispetto alle borse centralizzate, in quanto non devono pagare per costose infrastrutture e conformità normativa.

A livello di svantaggi:

* Problemi di liquidità: I DEX soffrono spesso di problemi di liquidità, che possono causare slittamenti e costi di transazione più elevati per gli ordini di grandi dimensioni.
* Complessità: I DEX possono essere più complessi da usare rispetto alle borse centralizzate, il che può dissuadere alcuni utenti dall'utilizzarli.
* Mancanza di controllo normativo: Poiché i DEX operano su una rete decentralizzata, non hanno lo stesso livello di supervisione normativa delle borse centralizzate. Ciò può rendere difficile la risoluzione delle controversie e l'applicazione delle leggi a tutela dei consumatori.
* Vulnerabilità al front-running: I DEX sono vulnerabili agli attacchi di front-running, in cui i trader utilizzano algoritmi avanzati per vedere e agire sugli ordini di altri trader prima che vengano eseguiti.
* Perdita impermanente: I fornitori di liquidità sui DEX possono subire una perdita impermanente, che è il risultato diretto del deposito di due asset per una specifica coppia di trading. Quando uno di questi asset è più volatile dell'altro, gli scambi in borsa possono ridurre la quantità di un asset nel pool di liquidità, con conseguente perdita per il fornitore di liquidità.

### Come funzionano

Il funzionamento di uno smart contract è simile a quello di altri trasferimenti su blockchain. Questi sono i passi necessari:

1. Un utente avvia una transazione dal proprio portafoglio blockchain;
2. la transazione arriva al database distribuito, dove viene confermata l'identità del portafoglio dell'utente.
3. la transazione, che può essere un trasferimento di fondi, viene approvata; questa comprende il codice che definisce il tipo di transazione da eseguire;
4. le transazioni vengono aggiunte come blocco all'interno della blockchain;
5. qualsiasi modifica dello stato del contratto segue lo stesso processo per essere aggiornata.

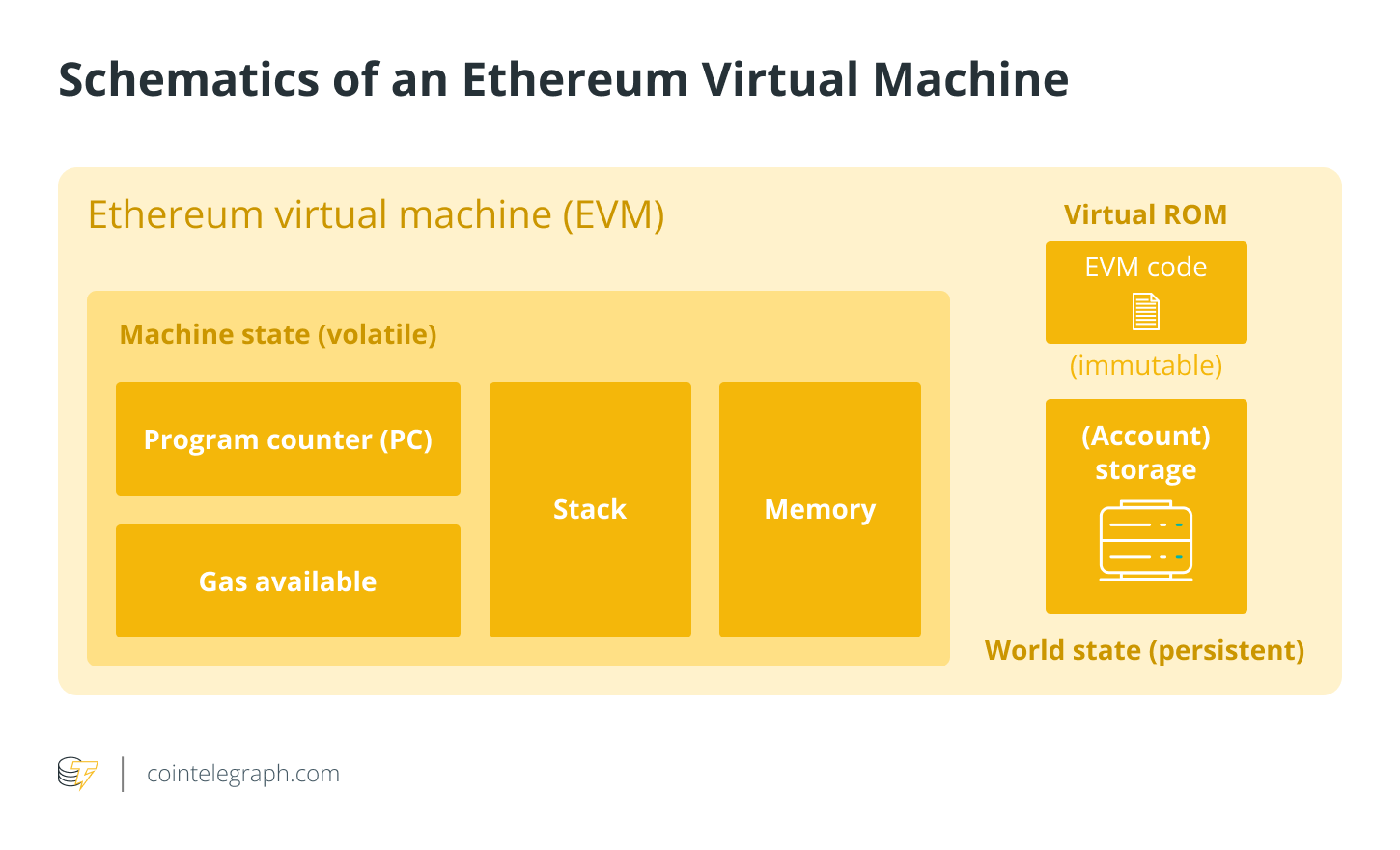
L’approvazione degli smart contract è definita trustless, dunque non si ha bisogno di una società intermediata e la gestione dei fondi è in mano ai validatori per le transazioni, agli staker per il Proof of Stake e i miner per il Proof of Work.

Gli smart contracts sono Turing-complete, pertanto possono potenzialmente essere eseguiti per sempre, bloccando ogni singolo nodo sulla blockchain.

### Dove vengono usati

Tra le piattaforme più vastamente utilizzate per sviluppare ed eseguire degli smart contract abbiamo:

* *Ethereum*: i contratti intelligenti sono scritti in un linguaggio di programmazione chiamato Solidity e vengono eseguiti dalla macchina virtuale di Ethereum (Ethereum Virtual Machine/EVM), Ethereum è considerata la piattaforma più popolare e la sua macchina virtuale di Ethereum è un motore di calcolo responsabile dell'implementazione, dell'esecuzione dei contratti smart e di calcolare lo stato di ogni nuovo blocco aggiunto alla blockchain di Ethereum.



Riferimento: <https://cointelegraph.com/news/what-is-an-ethereum-virtual-machine-evm-and-how-does-it-work>

* + La Ethereum Virtual Machine (EVM) è progettata come ambiente di runtime per i contratti intelligenti in Ethereum. È isolata dalle altre parti del sistema; ciò significa che qualsiasi operazione sulla EVM non dovrebbe influire in alcun modo sui dati o sui programmi, indipendentemente dal numero di volte in cui si chiama una particolare funzione.
  + La macchina è stata realizzata per essere in grado di eseguire qualsiasi tipo di contratto crittografico che possa essere costruito sulla blockchain di Ethereum. Per farlo utilizza un linguaggio di programmazione chiamato Solidity, che viene compilato nell'EVM per l'esecuzione. In questo modo, verranno creati degli smart contract e programmi che eseguono automaticamente solo quando certe condizioni si verificano.
  + Essa fornisce un ambiente Turing-complete per l'esecuzione di script e contratti intelligenti. Ciò significa che tutto ciò che può essere implementato con un computer può essere eseguito su EVM. Infatti, ogni nodo della rete di Ethereum deve accordarsi con quello successivo per eseguire le stesse istruzioni. Di fatto è un singleton e ognuno dei nodi esegue una copia locale della EVM. Tutte le operazioni sono eseguite su un’area dati virtuale denominata “stack”, formata da 1024 words di 256 bit.
  + Essa ha due parti principali:
    - EVM (la parte che esegue il codice sorgente di Solidity): La EVM è scritta in C++ e utilizza LLVM come compilatore. È una macchina virtuale completa con tutte le caratteristiche che si possono desiderare in una macchina virtuale per smart contract di uso generale, come il supporto di più linguaggi di programmazione, funzioni di sicurezza, ambienti di runtime e altro ancora. Consente inoltre di scrivere bytecode EVM personalizzati.
    - Zii (Uncles): Si tratta di piccoli pezzi di smart contract o di dati memorizzati sulla blockchain. È una funzione utile perché consente di memorizzare metadati sul programma.
  + La criptovaluta nativa della piattaforma Ethereum è ETH, definita anche come Ether. Questa valuta è l’oggetto di scambi degli smart contracts. Di fatto, ne vengono create 5 ogni 12 secondi. Il consenso si raggiunge tramite proof-of-stake. Possiamo inoltre distinguere ETC (Ethereum Classic), hard fork di Ethereum (ETH), che permette su di essa l’esecuzione degli smart contract. La politica viene modificata, riducendo la ricompensa del blocco del 20% al 5 milionesimo numero di blocco e del 20% ogni 5 milionesimi.
  + All’interno di questa rete si parla di *gas* come il costo aggiuntivo richiesto per eseguire uno smart contract o una transazione sulla rete blockchain. È ciò che permette anche ai minatori (prima) e ai validatori (ora) di trarre profitto e fornisce un incentivo per svolgere il lavoro necessario a proporre il prossimo blocco di transazioni per la blockchain. Due fattori determinano la quantità di gas richiesta da una transazioni:
    - La sua complessità
      * Questo si determina tramite la presenza degli *opcode*, presenti nel bytecode (intermedio tra linguaggio macchina e linguaggio di programmazione) di EVM, che determina la complessità di ciascuna operazione (es. STOP, ADD, MUL, SUB, etc.). Esiste una lista completa dei vari opcodes con un consumo di gas ad essi associato.
    - Il carico corrente sulla rete Ethereum
      * La formula per il calcolo del gas è:

.

Il prezzo della tariffa base è determinato dalla congestione della rete di Ethereum. La mancia è una tassa prioritaria e viene impostata automaticamente dalla maggior parte dei portafogli per accelerare il completamento della transazione, quindi non è molto importante per il calcolo del gas.

* + - * La commissione di base per il blocco corrente è calcolata in base ai blocchi precedenti. Quando un nuovo blocco viene aggiunto alla blockchain, c'è una quantità prestabilita di gas, chiamata gas incluso, che può contenere (somma di tutte le commissioni del gas dalle transazioni nel blocco).
  + La presenza della EVM permette l’esecuzione di codice non sicuro senza perdere dati, interagendo anche con contratti complicati e con un calcolo deterministico. Grazie al fatto che ogni nodo deve raggiungere consenso ad ogni momento, si ha che il sistema risulti essere più sicuro e robusto contro i fallimenti. Inoltre può costruire contratti stateful (cioè, validi e memorizzabili su un supporto)
  + La transazione dovrà sempre indicare:
    - GAS PRICE: prezzo che l’autore di una transazione può pagare per unità di gas ed incide sulla rapidità con cui la transazione verrà incluso in un blocco aggiunto alla blockchain. Se si ha un prezzo alto, si avranno commissioni maggiori per i nodi che aggiungeranno il blocco alla blockchain (miner); questi tenderanno a massimizzare il loro guadagno per blocco.
    - GAS LIMIT: quantità massima di unità di gas che si è disposti a spendere per l’esecuzione della transazione attuale. Se la transazione può consumare più risorse di quelle assegnate, l’esecuzione si blocca e la transazione fallisce non impegnando altre risorse, mentre se inferiore, la restante parte verrà rimborsata in Ether all’autore della transazione.
* *Hyperledger*: è un sistema open source sviluppato dalla Linux Foundation e non è una blockchain, ma una piattaforma flessibile su cui è possibile sviluppare contratti intelligenti.
* *Polkadot*: è un'alternativa alla blockchain ed è famosa per la sua capacità di ospitare parachain, cioè catene all'interno di catene, consentendo un numero maggiore di transazioni rispetto al normale.
* *Solana*: gli smart contract Solana possono essere creati utilizzando diversi linguaggi di programmazione. Mentre il linguaggio nativo per i contatti intelligenti di Solana è Rust, il protocollo supporta anche lo sviluppo di contratti intelligenti in C++ e Solidity, oltre che in C++ e Solidity, oltre al supporto di altri linguaggi attraverso client JSON RPC API SDK di terze parti.
* *Cosmos*: Cosmos è un ecosistema in espansione di blockchain indipendenti e interconnesse collegate attraverso il protocollo Inter-Blockchain Communication. Gli sviluppatori possono scegliere di costruire blockchain autonome e specifiche per le applicazioni che possono facilmente interconnettersi. Il protocollo standard per la comunicazione inter blockchain, l'IBC, consente alle blockchain dell'ecosistema di connettersi in modo da poter trasferire token e altri dati tra le blockchain. Cosmos attualmente ha tre diversi SDK che consentono agli sviluppatori di scrivere contratti intelligenti in Javascript, Rust o Solidity.
* *Stellar*: Stellar è stata fondata nel 2014, il che la rende una delle più antiche piattaforme di contratti intelligenti. È mantenuta dalla Stellar Development Foundation ed è stata ripetutamente proclamata come una delle più interessanti startup blockchain in circolazione.

### Tipologie

* Smart Legal Contracts (Contratti legali intelligenti)
  + Questi contratti sono legalmente applicabili e richiedono alle parti di soddisfare i loro obblighi contrattuali. Le parti possono incorrere in severe azioni legali in caso di inadempienza.
  + Per creare alcuni contratti legali intelligenti, le parti coinvolte lavorano sul codice del contratto intelligente - o lo fanno i loro sviluppatori di software - finché non si accordano sui termini e sulle condizioni dell'accordo. Altri contratti blockchain, come quelli dei siti di scommesse, sono creati da terzi. In entrambi i casi, le parti che stipulano un accordo non hanno bisogno di incontrarsi di persona o di conoscere l'identità dell'altro. Ciascuna di esse può apporre una firma digitale e lo status giuridico del contratto viene ufficializzato.
* Esempio reale:
  + Con l'avvento dei contratti intelligenti, il regno del diritto contrattuale si sta spostando in forma digitale. Home Depot utilizza contratti legali intelligenti con i fornitori. Il direttore delle operazioni finanziarie di Home Depot, Brian Quartel, spiega in un caso di studio IBM che stanno essenzialmente creando visibilità tra loro e i loro fornitori su ciò che è stato spedito e ricevuto.
  + L'attuale quadro giuridico per i contratti intelligenti varia nei sistemi giudiziari di varie regioni e nazioni. Esperti legali e associazioni stanno lavorando per definire uno smart contract legalmente applicabile e i processi si stanno evolvendo man mano che diventano più comuni. In California, ad esempio, è già possibile sposarsi con una licenza di matrimonio per smart contract.
* Organizzazioni autonome decentralizzate (DAO)
  + Per una DAO, la spina dorsale è il suo contratto intelligente. Il contratto è vincolato a regole specifiche che sono codificate nei contratti della blockchain e che si fondono con i meccanismi di governance. Ha diversi casi d'uso che vanno dal semplice al complesso, a seconda del numero di parti interessate.
  + Le DAO sono open-source e sono caratterizzate da trasparenza e, in teoria, sono incorruttibili. Inoltre, qualsiasi azione intrapresa dai membri della comunità viene sostituita da un codice auto-applicativo.
  + Possono essere autonome e autosufficienti utilizzando il codice dei contratti intelligenti concordato da tutti i partecipanti che hanno fondato la DAO, e il codice e la struttura aziendale possono evolvere utilizzando qualsiasi sistema di voto stabilito dalla DAO.
  + Essendo privo di fiducia, il codice persisterà con o senza gestione. Il cambio di leadership non influenza lo scopo, la visione o la funzione della DAO. E come tutti i contratti intelligenti, i DAO sono open source. Sono visibili a tutti i membri e non membri del DAO; i bug possono essere corretti e i suggerimenti possono essere proposti al DAO da chiunque.
* Esempio reale:
  + Nel febbraio 2022 è nata una collaborazione a tre tra Molecule, un'azienda di tecnologia decentralizzata, Apollo Health Ventures, un fondo di venture capital focalizzato sulla ricerca sulla longevità, e VitaDAO, un DAO focalizzato sul finanziamento della ricerca sulla longevità.
  + VitaDAO crea opportunità di finanziamento per gli scienziati biotecnologici senza richiedere sovvenzioni o registrare brevetti. A ogni progetto che VitaDAO prende in carico viene assegnato un IP-NFT, ovvero un gettone di proprietà intellettuale non fungibile. In questo modo, i membri del DAO gestiscono le proposte di ricerca e i loro progressi, assegnano i fondi ai ricercatori scelti e trovano persino i pazienti per la ricerca.
  + Oltre a eliminare la gerarchia aziendale dalle strutture di business, una DAO completamente autonoma gestirà transazioni come la revisione tra pari dei membri del team e l'assegnazione di token per i compiti portati a termine.
* Contratti logici applicativi (ALC)
  + Un altro tipo di smart contract in Blockchain è rappresentato dagli Application Logic Contracts (ALC), che consentono ai dispositivi di funzionare in modo sicuro e autonomo. Inoltre, gli ALC garantiscono una maggiore automazione, transazioni più economiche e scalabilità.
  + Questi contratti contengono un codice basato su un'applicazione, che di solito rimane sincronizzato con altri contratti blockchain. Consentono la comunicazione tra diversi dispositivi, come la fusione dell'Internet delle cose (IoT) con la tecnologia blockchain.
* Esempio reale
  + Beam Technologies, una startup con sede a Columbus, Ohio, ha lanciato uno spazzolino intelligente. Lo spazzolino è coordinato con la compagnia assicurativa dentale recentemente lanciata, che utilizza sensori nello spazzolino per raccogliere dati sull'igiene dentale dell'utente. Lo spazzolino "dialoga" con Beam Dental e influisce sulle tariffe del premio dentale dell'utente: più a lungo si usa lo spazzolino e più volte lo si usa ogni giorno, più il premio si riduce.
  + La chiave per gli ALC nell'ambito dell'Internet degli oggetti è l'esecuzione automatizzata del trasferimento dei dati, che attiva risposte diverse. Utilizzando il codice basato sulle applicazioni e la blockchain come spina dorsale per il trasferimento e l'archiviazione sicuri dei dati, gli sviluppatori possono trovare innumerevoli ruoli inventivi che gli ALC possono svolgere.

### Pro e contro

Alcuni dei vantaggi più degni di nota dei contratti intelligenti sono i seguenti:

* *Validità giuridica immediata*

In merito ai contrati intelligenti, questi pongono un meccanismo di vincolatività nella tecnologia blockchain e smart contract che impedisce dall’inizio l’inadempimento; infatti, per un nodo è tecnicamente impossibile violare volontariamente le condizioni prestabilite e ciò pone in secondo piano conseguenze legate all’inadempimento.

* *Velocità ed efficienza*

Grazie all'elevato grado di automazione e all'autoesecuzione dei termini concordati, i contratti intelligenti semplificano l'intero ciclo di vita dell'implementazione del contratto, contribuendo a un'esecuzione rapida, continua ed efficiente. Alcuni vantaggi facilmente raggiungibili che i contratti intelligenti consentono di ottenere: pagamenti automatizzati, richieste di risarcimento assicurativo, catene di fornitura trasparenti ed efficienti, efficienza aziendale, catene di fornitura trasparenti ed efficienti, una governance aziendale efficiente o lo snellimento della gestione dei dati per gli studi clinici.

* *Trasparenza*

I contratti intelligenti contribuiscono a garantire una maggiore trasparenza e allo stesso tempo a ridurre le possibilità di corruzione, dal momento che qualsiasi modifiche apportate al contratto richiedono il consenso di tutte le parti coinvolte o almeno possono essere rintracciate facilmente. La possibilità di manipolazione o di inadempimento da parte di un singolo individuo è altamente improbabile.

* *Sicurezza*

Grazie all'uso della crittografia dei dati, i contratti intelligenti sono a prova di manomissione e rappresentano un'alternativa altamente sicura ai contratti cartacei. I contratti intelligenti sono pezzi di codice che possono essere prontamente riutilizzati per operazioni simili con modifiche minime in base ai requisiti

* *Eliminazione degli intermediari e convenienza economica*

Eliminando la necessità di terze parti o intermediari per l'esecuzione, gli smart contract eliminano il rischio di manipolazione. I risparmi sui costi derivano dalla rimozione dei livelli di intermediazione nelle relazioni e nelle transazioni peer-to-peer. L'esecuzione automatica dei contratti intelligenti elimina potenzialmente la necessità di un'applicazione istituzionale e presenta un'alternativa più economica ed efficace per la garanzia ex-ante della performance. L'utilizzo degli smart contract comporta l'eliminazione degli errori dovuti alla compilazione manuale di numerosi moduli.

Possiamo tuttavia citare una serie di svantaggi:

* Il primo riguarda l'incapacità degli smart contract di valutare gli eventi del mondo reale. Questo non deve sorprendere, poiché le blockchain sono generalmente ambienti indipendenti e separati. La separazione è un meccanismo che garantisce la sicurezza della rete sulla base dell'algoritmo di consenso.
  + Di conseguenza, i dati che risiedono nei database, dalle informazioni sui dipendenti a dati qualsiasi relativi al mondo esterno, sono dati al di fuori della blockchain considerati potenzialmente pericolosi per la rete stessa.
* La seconda limitazione fa riferimento alla dimensioni massime dei contratti per gli smart contract. La limitazione è definita dal protocollo EIP-170 che previene gli attacchi DOS (Denial of Service) attraverso l'esecuzione di contratti infiniti per congestionare la rete. Gli sviluppatori sono tenuti a includere funzionalità che non superino il limite massimo di 24 KB.

### Considerazioni legali e casi d’uso

* Il Parlamento italiano ha approvato la legge n. 12/2019, che definisce la tecnologia distributed ledger e convalida gli smart contract con forma scritta. La legge indica che un documento informatico registrato tramite DLT ha gli effetti legali della validazione temporale elettronica. Inoltre, stabilisce che gli smart contract sono programmi informatici che operano su ledger distribuiti e la loro esecuzione vincola automaticamente due o più parti sulla base di effetti predefiniti, dando completa equivalenza tra contratti cartacei e documenti digitali
* La Commissione giuridica del Regno Unito raccomanda che il quadro giuridico in Inghilterra e Galles supporti l'uso dei contratti legali intelligenti. Definisce tre categorie di contratti legali intelligenti, ma è necessario affrontare questioni relative alla formazione del contratto, all'interpretazione, alla protezione dei consumatori, ai rimedi e alla giurisdizione. Secondo il diritto inglese, i requisiti per un contratto legalmente vincolante possono essere soddisfatti da uno smart contract. Tuttavia, l'aggiunta di codice nel mix interpretativo può dare origine a difficoltà interpretative, richiedendo a una persona ragionevole con conoscenza e comprensione del codice di capire il significato del contratto.

Alcuni casi d’uso includono:

* Le *DAO (Decentralized Autonomous Organizations)* sono un tipo di struttura di governance organizzativa che consiste in comunità native di Internet che lavorano con l'infrastruttura digitale e che si concentrano sulla condivisione di un'infrastruttura digitale e si concentrano su una missione condivisa e su un protocollo blockchain. Esse sono gestite e possedute dai contributori (membri) della comunità, e operano in modo fluido dal basso verso l'alto con una struttura piatta, ma comunque strutturata.
  + Nel contesto delle DAO, gli smart contract sono utilizzati per rappresentare beni e capitali sotto forma di token, che possono essere utilizzati per esercitare il potere di voto, governare un protocollo, allocare fondi e incentivare la partecipazione alla rete. Ad esempio, quando un membro della DAO vuole votare su una proposta, può utilizzare i propri token per esprimere il proprio voto attraverso uno smart contract. Se la proposta viene approvata con il numero di voti richiesto, lo smart contract eseguirà automaticamente le azioni specificate nella proposta, come l'assegnazione di fondi a un progetto specifico o l'aggiornamento del codice del protocollo.
* *DeFi* (*Decentralized Finance)* è un ecosistema in rapida crescita di applicazioni finanziarie costruite sulla base della tecnologia blockchain. Le applicazioni DeFi sono progettate per fornire servizi finanziari in modo decentralizzato, aperto e senza permessi, consentendo a chiunque abbia una connessione a Internet e un portafoglio di criptovalute di partecipare al sistema finanziario globale senza la necessità di intermediari tradizionali come banche o istituzioni finanziarie.
  + Gli smart contract sono utilizzati per creare protocolli finanziari accessibili e utilizzabili da chiunque abbia una connessione a Internet, senza bisogno di intermediari come banche o istituzioni finanziarie. Questi protocolli possono essere utilizzati per un'ampia gamma di attività finanziarie, come prestiti, mutui, scambi e assicurazioni.
    - Ad esempio, in un protocollo di prestito DeFi, gli smart contract sono utilizzati per automatizzare il processo di prestito e di erogazione. I mutuatari possono presentare garanzie sotto forma di criptovalute, che vengono tenute in deposito dallo smart contract. Il contratto emette quindi automaticamente un prestito nella valuta desiderata, con i termini del prestito e il tasso di interesse specificati nel contratto stesso. Il mutuatario deve rimborsare il prestito più gli interessi entro una certa data, altrimenti la garanzia viene incamerata dal mutuante.
    - Allo stesso modo, in un protocollo di trading DeFi, gli smart contract sono utilizzati per eseguire le transazioni tra le parti. Gli acquirenti e i venditori possono inviare i loro ordini allo smart contract, che li confronta in base ai criteri specificati. Lo smart contract esegue quindi la compravendita, trasferendo gli asset tra le parti e aggiornando i rispettivi saldi dei conti.

### Possibili problemi

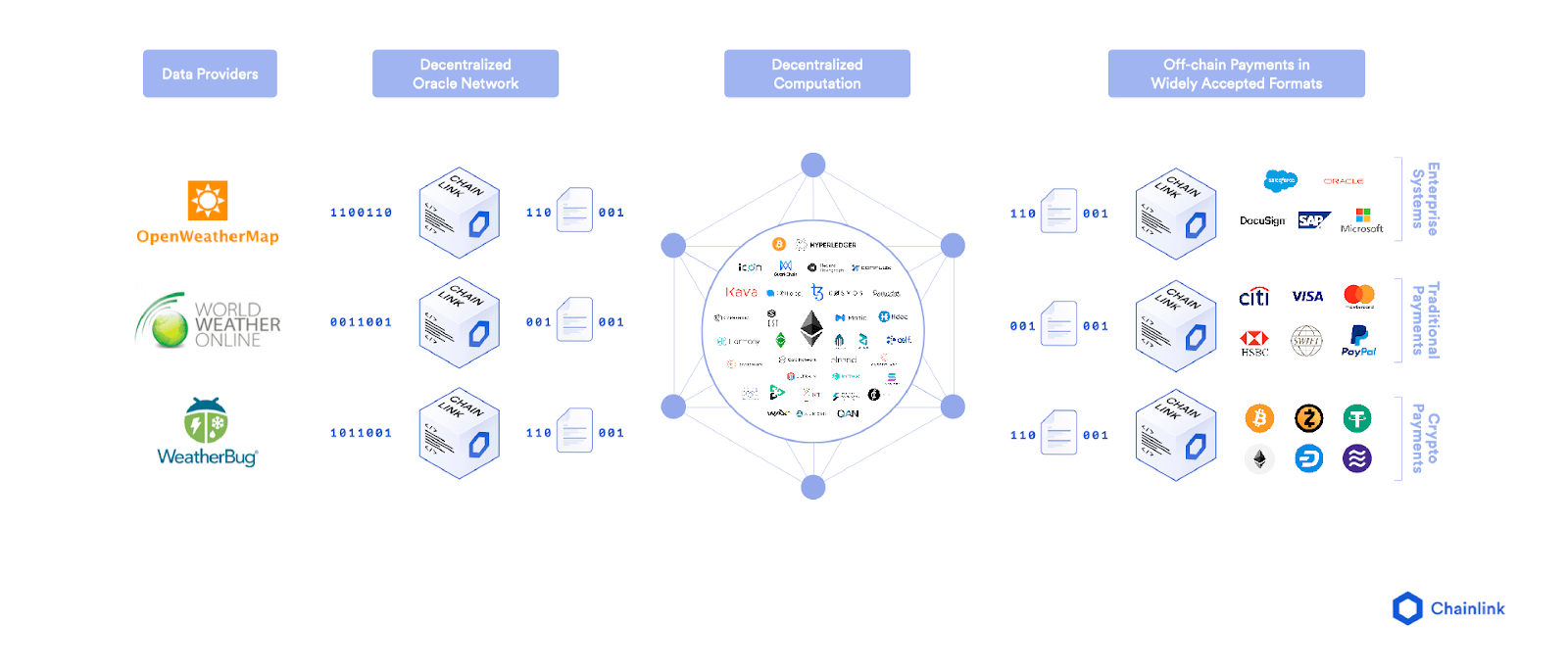
Per loro natura, gli smart contract sono deterministici e tutti i nodi all’interno della sua rete devono raggiungere tutti il consenso. I contratti smart minimizzano efficacemente il rischio di controparte e forniscono trasparenza, ma presentano ancora diversi limiti alla loro capacità.

* La crescente necessità di dati esterni che confluiscono nelle blockchain e, per estensione, negli smart contract ha portato a dibattiti e innovazioni intorno agli oracoli (oracles), che sono flussi di dati provenienti da sistemi esterni che alimentano le blockchain con informazioni vitali che gli smart contract possono avere bisogno di eseguire in condizioni specifiche. La crescente necessità di oracoli rappresenta la continua espansione dei sistemi blockchain in casi d'uso pratici e reali, dove l'accuratezza dei dati è fondamentale.
* Gli oracoli recuperano e verificano i dati esterni per le blockchain e gli smart contract attraverso metodi quali API web o feed di dati di mercato. Il tipo di dati richiesti dagli smart contract può includere informazioni sui prezzi, informazioni meteo o persino la generazione di numeri casuali per il gioco d'azzardo. L'utilizzo degli oracoli consiste nell'interrogare la fonte di dati per ottenere informazioni specifiche e successivamente connettersi a tale fonte per interfacciarsi tra la blockchain e il feed di dati. Di conseguenza, i contratti intelligenti possono essere eseguiti in base alle informazioni specifiche provenienti dal flusso di dati.
* Questi consentono la creazione degli hybrid smart contracts, combinando una infrastruttura on-chain con una off-chain e quindi fornendo calcolo a reti decentralizzate di oracoli (quindi, multipli per evitare un singolo punto di fallimento e anzi estendere la persistenza e la ridondanza della rete a tutti i nodi). Queste reti sono decentralized oracle networks (DONs).
* I DON possono essere utilizzati per collegare vari tipi di dati esterni a e da blockchain, consentendo di scrivere contratti intelligenti ibridi attorno a quei dati specifici. Alcuni dei tipi di dati iniziali resi accessibili includono:
  + Price Feeds - dati sui prezzi delle attività aggregati da centinaia di borse, ponderati per il volume e ripuliti da outlier e wash trading.
  + Proof of Reserve - dati aggiornati sugli attuali saldi di riserva a sostegno degli asset tokenizzati, come le riserve di BTC che garantiscono WBTC o il conto bancario in USD che garantisce TUSD.
  + Qualsiasi API - dati premium provenienti da API protette da password, che vanno dalle previsioni del tempo e dai risultati delle partite sportive alle informazioni provenienti da un backend aziendale e da una rete IoT.
  + Blockchain Middleware - livello di astrazione per un sistema fuori catena per leggere e scrivere dati da e verso smart contract su qualsiasi rete blockchain.
* Per comprendere meglio la differenza tra i componenti on-chain e off-chain, identifichiamo i ruoli distinti di ciascuno:
* On-Chain: Blockchain
  + Mantenere un libro mastro persistente che fornisce una custodia autorevole delle attività degli utenti e interagisce con le chiavi private.
  + Eseguire il regolamento finale elaborando transazioni irreversibili che trasferiscono valore tra gli utenti.
  + Fornire la risoluzione delle controversie e i guardrail per garantire il corretto funzionamento dei servizi off-chain eseguiti da un DON.
* Off-Chain: Rete decentralizzata di Oracle
  + Recuperare, convalidare, proteggere e consegnare dati da API esterne a smart contract in esecuzione su blockchain e soluzioni di livello 2
  + Eseguire vari tipi di calcoli per gli smart contract in esecuzione su blockchain e soluzioni layer-2.
  + Trasmettere le uscite del codice degli smart contract ad altre blockchain o a sistemi esterni.
* I DON possono eseguire una serie di calcoli fuori catena per conto dello smart contract, per aiutarlo a ottenere input specifici o a generare determinate caratteristiche non possibili sulla sua specifica blockchain, come la privacy, la scalabilità e l'equità degli ordini. Alcune delle computazioni off-chain attuali e future possibili attraverso i DON includono:
  + Keeper Network - bot di automazione che eseguono compiti di manutenzione regolare per lo smart contract, svegliandolo quando deve eseguire funzioni chiave sulla catena.
  + Off-Chain Reporting (OCR) - aggregazione scalabile delle risposte dei nodi oracolo in un DON che vengono poi consegnate sulla catena in un'unica transazione per ridurre i costi sulla catena.
  + Scalable Computation - esecuzione di contratti ad alta velocità e a basso costo per smart contract autonomi, che si sincronizza periodicamente sulla catena utilizzando la tecnologia layer-2 esistente.
  + Verifiable Randomness Function (VRF) - generazione di numeri casuali sicura e verificabile, supportata da prove crittografiche che dimostrano l'integrità del processo.
  + Data and Computation Privacy - calcolo oracolare che preserva la privacy e che rende i dati sensibili disponibili in modo confidenziale agli smart contract utilizzando prove a conoscenza zero (DECO), hardware fidato (Town Crier), calcolo multi-party sicuro e/o utilizzando comitati DON selezionati.
  + Fair Sequencing Services (FSS) - ordinamento decentralizzato delle transazioni basato su una nozione predefinita di equità, che impedisce il frontrunning e il miner extractable value (MEV).
  + On-Chain Contract Privacy - privacy delle transazioni per uno smart contract attraverso una decorrelazione tra la logica del contratto e l'output del regolamento, utilizzando il DON per ritrasmettere la comunicazione tra le due parti, come nel caso di Mixicles.

Un oracolo blockchain è un pezzo sicuro di middleware che facilita la comunicazione tra le blockchain e qualsiasi sistema esterno alla catena, compresi i fornitori di dati, le API web, i backend aziendali, i fornitori di cloud, i dispositivi IoT, le firme elettroniche, i sistemi di pagamento, altre blockchain e altro ancora. Questi svolgono diverse funzioni chiave:

* Ascoltare - monitorare la rete blockchain per verificare eventuali richieste di dati fuori catena da parte di utenti o smart contract.
* Estrarre - recuperare i dati da uno o più sistemi esterni, come le API off-chain ospitate su server web di terze parti.
* Formattare - formattare i dati recuperati da API esterne in un formato leggibile dalla blockchain (input) e/o rendere i dati della blockchain compatibili con un'API esterna (output).
* Validare - generare una prova crittografica che attesti le prestazioni di un servizio oracolo utilizzando qualsiasi combinazione di firma dei dati, firma delle transazioni blockchain, firme TLS, attestazioni di Trusted Execution Environment (TEE) o prove a conoscenza zero.
* Computare - eseguire qualche tipo di calcolo sicuro fuori dalla catena per lo smart contract, come ad esempio calcolare una mediana da più invii dell'oracolo o generare un numero casuale verificabile per un'applicazione di gioco.
* Broadcast - firma e trasmette una transazione sulla blockchain per inviare i dati e qualsiasi prova corrispondente sulla catena per il consumo da parte dello smart contract.
* Output (opzionale) - inviare dati a un sistema esterno in seguito all'esecuzione di uno smart contract, come ad esempio trasmettere istruzioni di pagamento a una rete di pagamento tradizionale o attivare azioni da un sistema cyber-fisico.

Per svolgere le funzioni sopra descritte è necessario che il sistema oracolo operi contemporaneamente sia sulla blockchain che al di fuori di essa. La componente on-chain serve a stabilire una connessione alla blockchain (per ascoltare le richieste), trasmettere dati, inviare prove, estrarre dati della blockchain e potenzialmente eseguire calcoli sulla blockchain. La componente off-chain serve per elaborare le richieste, recuperare e formattare i dati esterni, inviare i dati della blockchain a sistemi esterni ed eseguire calcoli off-chain per una maggiore scalabilità, privacy, sicurezza e vari altri miglioramenti degli smart contract.



Riferimento: <https://blog.chain.link/what-is-the-blockchain-oracle-problem/>

Esistono diverse forme di oracoli, tra cui:

* Gli *oracoli hardware* sono sensori integrati in oggetti fisici tangibili. Gli esempi principali sono la tracciabilità della catena di approvvigionamento con l'uso di tag RFID per fornire dati come le condizioni ambientali dei prodotti alla blockchain.
* Gli *oracoli software* sono la forma più comune che attinge i dati da fonti di terze parti, come le API web, e possono includere informazioni del mondo reale come lo stato dei voli e i dati meteorologici.
* Gli *oracoli di consenso* rappresentano un passo avanti verso gli oracoli decentralizzati e si basano sull'aggregazione di dati provenienti da diversi oracoli con metodi proprietari per determinarne l'autenticità e l'accuratezza.
* Gli *oracoli in entrata (inbound oracles)* riflettono gli scenari "se questo accade allora fai quello" associati agli oracoli software, come ad esempio "se questo prezzo è raggiunto da un asset, allora attiva una vendita".
* Gli *oracoli in uscita (outbound oracles)* consentono agli smart contract di inviare dati a fonti esterne alla rete blockchain su cui si trovano e sono anch'essi oracoli software.

Qui si parla di *Oracle Problem*, in quanto un oracolo fornisce agli smart contract le risposte alle domande sul mondo. Nella maggior parte dei casi, senza un oracolo che fornisca informazioni, non ci sarebbe modo per uno smart contract di conoscere le cose che deve sapere per svolgere il suo lavoro.

* Poiché un oracolo determina ciò che uno smart contract vede (cioè controlla gli input dello smart contract), ha anche il potere di controllare ciò che lo smart contract fa in risposta a questi input. Ciò significa che gli oracoli possiedono un'enorme quantità di potere quando si tratta di smart contract (specialmente in contesti di mercato predittivo). Se l'oracolo è compromesso, lo è anche l'intero contratto; le blockchain non possono ricevere dati da o inviare dati a sistemi esterni come funzionalità integrata.
* Le blockchain non sono adatte a rispondere a domande che si spingono nel regno della soggettività o che richiedono dati esterni non facilmente accessibili a tutti i nodi della rete. Ad esempio, una semplice domanda come "Qual è il prezzo di mercato del Bitcoin?" o "Che tempo fa a New York?" può suscitare un'ampia gamma di risposte diverse che possono variare a seconda della fonte di dati utilizzata e del momento in cui vengono richiesti. La domanda diventa quindi: qual è la risposta corretta e come si può verificare che sia vera?
* L'introduzione della soggettività nel livello di base della blockchain apre il vaso di Pandora a tutta una serie di problemi di sicurezza, affidabilità e governance, mettendo a rischio la stessa proposta di valore che le blockchain intendono fornire.
* È estremamente difficile gestire e far rispettare la qualità dei dati off-chain inviati dai nodi della blockchain, poiché chiunque può gestire un nodo in modo pseudo-anonimo e inviare risposte, anche se non è disposto ad acquistare un abbonamento a un'API off-chain di alta qualità. Se la qualità dei dati fosse imposta, la blockchain avrebbe un limite superiore inferiore alla decentralizzazione, poiché i costi di gestione dei nodi aumenterebbero per ogni nuovo lavoro di oracolo sulla rete, incidendo sulla sicurezza di tutte le altre applicazioni in esecuzione su quella particolare blockchain.
* Un'altra grande preoccupazione è la scalabilità. Ogni volta che una nuova fonte di dati deve essere aggiunta alla rete o un metodo di aggregazione dei dati esistente deve essere adattato, è necessario un massiccio coordinamento di governance sociale per far sì che ogni nodo della rete sia d'accordo e aggiorni il proprio software. L'aggiunta di costi generali di governance porta a un aumento dell'attrito, a un rallentamento dello sviluppo delle funzionalità fondamentali della blockchain (come PoS e sharding) e a forti limiti all'innovazione di Oracle. In definitiva, maggiore è la complessità del livello di base della blockchain, maggiore è la superficie di attacco e il rischio per tutte le applicazioni che vi girano sopra.

L'intero scopo di uno smart contract è quello di ottenere il determinismo attraverso l'esecuzione tecnologica dei termini del contratto, in contrapposizione all'esecuzione probabilistica effettuata dall'uomo. Per raggiungere questo obiettivo, la blockchain non può avere un singolo punto di guasto, caratteristica che deve essere estesa all'oracolo se si vuole mantenere il determinismo degli smart contract end-to-end. Perché far funzionare un contratto multimilionario come smart contract su una blockchain completamente decentralizzata se un singolo oracolo centralizzato può controllare gli input che determinano l'esito del contratto (diventano Single Point of Failure)?

Esistono sistemi come Chainlink in grado di portare determinismo all’oracle layer, che ha sviluppato una rete di reti di oracoli decentralizzati (DON), con ogni DON che comporta una combinazione di più tecniche di sicurezza necessarie per servire un particolare caso d'uso. Alcune caratteristiche utili:

* Open-source - il fatto di essere una tecnologia open-source consente alla più ampia comunità blockchain di verificare in modo indipendente la sicurezza e l'affidabilità del codice sorgente e delle funzioni di Chainlink, nonché di contribuire al suo miglioramento.
* Adattatori esterni - la possibilità per i nodi di memorizzare in modo sicuro le chiavi API e di gestire i login degli account consente agli smart contract di recuperare dati da qualsiasi sistema e API esterni, compresi quelli protetti da password/credenziali.
* Decentralizzazione - l'impiego della decentralizzazione a livello di nodi e di fonti di dati assicura che nessun nodo o fonte di dati sia un singolo punto di guasto, fornendo agli utenti forti garanzie sulla disponibilità dei dati, sulla loro puntualità e sulla loro resistenza alla manipolazione.
* Firma dei dati - il fatto che i nodi firmano crittograficamente i dati che forniscono agli smart contract consente agli utenti di identificare quali nodi hanno inviato i dati e di esaminare la loro storia passata per determinare la qualità delle loro prestazioni.
* Accordi di servizio - l'utilizzo di accordi vincolanti on-chain tra lo smart contract richiedente e il fornitore dell'oracolo, che delineano i termini del servizio dell'oracolo e le penalità/premi per le prestazioni, fornisce agli utenti garanzie applicabili sulla qualità delle loro richieste di dati off-chain.
* Sistemi di reputazione - l'alimentazione dei dati firmati sulla catena nei sistemi di reputazione consente agli utenti di prendere decisioni informate su quali nodi sono validi e quali no, sulla base di una serie di metriche come i lavori eseguiti con successo, l'elenco dei clienti serviti, il tempo medio di risposta, ecc.
* Servizi di certificazione - consentire ai nodi di aumentare la loro sicurezza e affidabilità ottenendo un numero qualsiasi di certificazioni può fornire agli utenti garanzie aggiuntive come KYC, posizione geografica del nodo, revisioni di sicurezza della sua infrastruttura e altro ancora.
* Crittografia e hardware avanzati - fornire flessibilità per la crittografia più avanzata (come le prove a conoscenza zero) e l'hardware (come gli ambienti di esecuzione fidati) consente agli oracoli di svolgere funzioni aggiuntive come dimostrare l'origine dei dati (ad esempio, dati specifici provengono da un server specifico), mantenere i dati riservati, eseguire calcoli fuori catena e altro ancora.

## Linguaggi per smart contract: Solidity

Solidity è un linguaggio ad alto livello usato per implementare gli smart contract designato per essere utilizzato nella Ethereum Virtual Machine (EVM), supportando l’ereditarietà, librerie, tipi definiti dall’utente ed altre caratteristiche, essendo linguaggio staticamente tipizzato. Si ispira a C++, Python, JavaScript e simili.

Si può scriverne direttamente codice al link: <https://remix.ethereum.org>

### Esempi di codice e passi base

La prima cosa da fare è dichiararsi la versione da utilizzare, con la direttiva *pragma* e si aggiunge la licenza di utilizzo.

//SPDX-License-Identifier: MIT

pragma solidity ^0.8.0;

Di seguito l’esempio semplice di un contratto, creando due funzioni di *get* e *set* e un costruttore nella sintassi specifica:

pragma solidity ^0.8.0;

contract MyContract{

    string value;

    constructor() public{

        value = "myValue";

    }

    function get() public view returns(string memory){

        return value;

    }

    function set(string memory \_value) public{

        value = \_value;

    }

}

Esistono diversi tipi di Solidity, ognuno dei quali ha una funzione specifica.

* booleano (*bool*): è utilizzato per rappresentare i valori booleani, ovvero vero o falso. In Solidity, il valore "true" è rappresentato dal numero 1 e il valore "false" dal numero 0.
* intero (*int*): viene utilizzato per rappresentare numeri interi positivi o negativi. In Solidity, esistono diversi tipi di dati interi con dimensioni variabili, a seconda del numero di bit utilizzati per rappresentare il valore. Per gli interi solamente positivi, avremo *uint* (*unsigned int*) e ce ne sono diversi: uint, uint8, uint16, uint24, uint32, uint64, uint128 e uint256
* *address*, che indica un indirizzo pubblico per inviare Ether
* stringa (*string*): il tipo di dati stringa viene utilizzato per rappresentare stringhe di caratteri. In Solidity, le stringhe vengono codificate in UTF-8 e la loro lunghezza massima è di 2^256 caratteri. Si basano su Byte, ma le stringhe si usano per il testo, mentre i Byte grezzi per memorizzarne ulteriori. È inoltre possibile utilizzare byte1, byte2, byte3 fino a byte32. Questi sono usati per lo stesso scopo, per la memorizzazione di dati di testo e di byte ma usano meno gas, il che è ottimo per gli utenti perché dovranno pagare meno ether per interagire con quelle variabili.
* *array*: il tipo di dati array viene utilizzato per rappresentare una serie di valori dello stesso tipo. In Solidity, esistono diversi tipi di array, tra cui gli array statici e gli array dinamici. Come per altri linguaggi, si dichiarano con le quadre
* *mapping*: il tipo di dati mapping viene utilizzato per rappresentare una corrispondenza tra una chiave e un valore. In Solidity, i mapping sono utilizzati spesso per creare strutture dati complesse.
* *struct*: il tipo di dati struct viene utilizzato per definire una struttura dati personalizzata che può contenere diversi tipi di dati.

Quando si esegue un contratto, si pagherà del gas (tassa della blockchain Ethereum) quando viene fatto il deploy del contatto specifico. Per esempio, una funzione di lettura come nel codice che segue, costerà solo quando chiamo *retrieve()*.

Immagine che contiene testo

Descrizione generata automaticamente

Un esempio di *struct* oppure di *array*:

    struct People{

        uint256 favoriteNumber;

        string name;

    }

    uint256 public favoriteNumbersList;

    People[] public people;

Distinguiamo:

* *memory*, che permette di salvare una variabile temporaneamente. Questa viene usata anche per indicare come si comporti una locazione “insicura” (ad esempio, strutture come array, struct e mappings. In questo modo, diciamo al compilatore esattamente come comportarsi quando non si sa esattamente quanta memoria allocare)
* *storage*, che la salva permanentemente e si vede anche al di fuori della funzione

Un esempio di seguito di codice che usa una struttura *mapping* per salvare e “mappare” un intero ad una stringa:

//SPDX-License-Identifier: MIT

pragma solidity ^0.8.0;

contract MyContract{

    uint256 public favoriteNumber;

    //People public person = People({favoriteNumber: 2, name : "Gabriel"});

    struct People{

        uint256 favoriteNumber;

        string name;

    }

    People[] public people; //static array

    mapping (string => uint256) public nameToFavoriteNumber; //dictionary to map name to a specific number

    //view, pure

    function store (uint256 \_favoriteNumber) public {

        favoriteNumber = \_favoriteNumber;

    }

    function retrieve() public view returns(uint256){

        return favoriteNumber;

    }

    //calldata, memory (temporary variable, like "name"), storage (permanent variable, exists even outside the function)

    function addPerson(string memory \_name, uint256 \_favoriteNumber) public{

        people.push(People(\_favoriteNumber, \_name));

        nameToFavoriteNumber[\_name]=\_favoriteNumber;

    }

}

SI può inoltre cambiare l’ambiente di esecuzione nella sezione di compilazione, ad esempio:

* Injected Provider setta (tramite un wallet come Metamask) l’utilizzo di un servizio esterno per interagire con lo smart contract che stiamo scrivendo,

È possibile che un file contenga più contratti, poi scegliendo su quale fare il deploy facilmente in fase di esecuzione. Come si vede, si può fare in questo modo, importando il codice precedente:

// SPDX-License-Identifier: MIT

pragma solidity 0.8.8;

import "./SimpleStorage.sol";

contract StorageFactory{

    SimpleStorage[] public simpleStorageArray;

    function createSimpleStorageContract() public{

        SimpleStorage simpleStorage = new SimpleStorage();

    }

}

Quando si importa una funzione, prendiamo subito la sua ABI (Application Binary Interface) nel contesto dell'informatica è un'interfaccia tra due moduli di programma, spesso tra sistemi operativi e programmi utente.

Gli smart contract scritti in linguaggi di alto livello come Solidity o Vyper devono essere compilati in bytecode eseguibile EVM; quando uno smart contract viene distribuito, questo bytecode viene memorizzato sulla blockchain ed è associato a un indirizzo. Per Ethereum ed EVM, uno smart contract è proprio questa sequenza di bytecode. Per accedere alle funzioni definite nei linguaggi di alto livello, gli utenti devono tradurre i nomi e gli argomenti in rappresentazioni di byte affinché il codice byte possa lavorare con esse.

Per interpretare i byte inviati in risposta, gli utenti devono riconvertire le tuple di valori di ritorno definiti nei linguaggi di alto livello. I linguaggi che compilano per l'EVM mantengono convenzioni rigorose su queste conversioni, ma per eseguirle è necessario conoscere i nomi e i tipi precisi associati alle operazioni. L'ABI documenta questi nomi e tipi in modo preciso, in un formato facilmente analizzabile, rendendo le traduzioni tra le chiamate di metodo volute dall'uomo e le operazioni degli smart-contract scopribili e affidabili.

È molto simile all'API (Application Program Interface), una rappresentazione leggibile dall'uomo dell'interfaccia di un codice. L'ABI definisce i metodi e le strutture utilizzate per interagire con il contratto binario, proprio come fa l'API, ma a un livello inferiore. L'ABI indica al chiamante della funzione di codificare le informazioni necessarie, come le firme delle funzioni e le dichiarazioni delle variabili, in un formato che l'EVM può comprendere per chiamare la funzione in bytecode; questa operazione è chiamata codifica ABI. La codifica ABI è per lo più automatizzata, gestita da compilatori come REMIX o da portafogli che interagiscono con la blockchain. L'ABI dei contratti è rappresentato in formato JSON. Esistono specifiche chiare su come codificare e decodificare un contratto ABI.

Il JSON di una ABI è come segue:

Immagine che contiene testo

Descrizione generata automaticamente

Riferimento: <https://www.quicknode.com/guides/ethereum-development/smart-contracts/what-is-an-abi/>

Un esempio di ereditarietà rispetto al codice precedente avviene con la keyword *is* per indicare il fatto di essere sottoclasse e ridefiniamo una funzione precedente (*overriding*) con la keyword *override*.

// SPDX-License-Identifier: MIT

pragma solidity ^0.8.0;

import "./SimpleStorage.sol";

contract ExtraStorage is SimpleStorage{

    function store(uint256 \_favoriteNumber) public override{

        favoriteNumber = \_favoriteNumber + 5;

    }

}

Possiamo settare una funzione per fare in modo ripaghi nelle valute Ethereum. Di seguito un recap:

Immagine che contiene tavolo

Descrizione generata automaticamente

Riferimento: <https://www.alchemy.com/gwei-calculator>

Similmente, possono essere introdotte due keywords:

* *payable* per fare in modo una transazione trasferisca delle valute
* *require* per fare in modo vengano correttamente eseguiti dei controlli e prevenire transazioni inutili
* *library,* simili ai contratti ma non si possono dichiarare variabili di stato oppure inviare degli ether

Possiamo vedere l’utilizzo di un oracolo con l’utilizzo di un metodo di conversione da Ether a dollari con un indirizzo di test e relativa conversione.

// SPDX-License-Identifier: MIT

pragma solidity ^0.8.8;

import "@chainlink/contracts/src/v0.8/interfaces/AggregatorV3Interface.sol";

contract FundMe {

    function fund() public payable {

        require(msg.value.getConversionRate() >= MINIMUM\_USD, "You need to spend more ETH!");

        // require(PriceConverter.getConversionRate(msg.value) >= MINIMUM\_USD, "You need to spend more ETH!");

        addressToAmountFunded[msg.sender] += msg.value;

        funders.push(msg.sender);

    }

    function getVersion() public view returns (uint256){

        // ETH/USD price feed address of Sepolia Network.

        AggregatorV3Interface priceFeed = AggregatorV3Interface(0x694AA1769357215DE4FAC081bf1f309aDC325306);

        return priceFeed.version();

    }

}

Un esempio di library è come segue, seguendo la logica di sopra:

// SPDX-License-Identifier: MIT

pragma solidity ^0.8.8;

import "@chainlink/contracts/src/v0.8/interfaces/AggregatorV3Interface.sol";

// Why is this a library and not abstract?

// Why not an interface?

library PriceConverter {

    // We could make this public, but then we'd have to deploy it

    function getPrice() internal view returns (uint256) {

        // Sepolia ETH / USD Address

        // https://docs.chain.link/data-feeds/price-feeds/addresses#Sepolia%20Testnet

        AggregatorV3Interface priceFeed = AggregatorV3Interface(0x694AA1769357215DE4FAC081bf1f309aDC325306);

        (, int256 answer, , , ) = priceFeed.latestRoundData();

        // ETH/USD rate in 18 digit

        return uint256(answer \* 10000000000);

        // or (Both will do the same thing)

        // return uint256(answer \* 1e10); // 1\* 10 \*\* 10 == 10000000000

    }

    // 1000000000

    function getConversionRate(uint256 ethAmount)

        internal

        view

        returns (uint256)

    {

        uint256 ethPrice = getPrice();

        uint256 ethAmountInUsd = (ethPrice \* ethAmount) / 1000000000000000000;

        // or (Both will do the same thing)

        // uint256 ethAmountInUsd = (ethPrice \* ethAmount) / 1e18; // 1 \* 10 \*\* 18 == 1000000000000000000

        // the actual ETH/USD conversion rate, after adjusting the extra 0s.

        return ethAmountInUsd;

    }

}

Similmente, possiamo rappresentare delle vulnerabilità matematiche poi risolte con questo esempio:

// SPDX-License-Identifier: MIT

pragma solidity ^0.8.0;

contract SafeMathTester{

    uint8 public bigNumber = 255; //unchecked --> check for the max number to not fail the single transaction

    function add() public{

        bigNumber = bigNumber + 1;

    }

}

/\*

In 0.8 above:

    function add() public{

        unchecked {bigNumber = bigNumber + 1;}

    }

The code might be more gas efficient

\*/

In Solidity, la parola chiave *modifier* viene utilizzata per definire un pezzo di codice riutilizzabile che può essere applicato alle funzioni, consentendo l'implementazione di funzionalità comuni di validazione, autorizzazione o altro che possono essere condivise tra più funzioni.

Inoltre:

In Solidity, le funzioni receive e fallback sono funzioni speciali utilizzate per gestire le transazioni in arrivo a un contratto quando nessun'altra funzione corrisponde alla firma della funzione.

La funzione receive ha la seguente firma:

*receive() external payable {*

*// codice per gestire le transazioni in entrata*

*}*

È una funzione speciale che viene chiamata automaticamente quando un contratto riceve etere senza una chiamata di funzione o quando un contratto riceve etere e la funzione chiamata non è contrassegnata come pagabile. La funzione receive deve essere contrassegnata come pagabile per poter ricevere etere e può essere definita solo una volta in un contratto.

La funzione di fallback, invece, ha la seguente firma:

*fallback() external payable {*

*// codice per gestire le transazioni in entrata*

*}*

È una funzione speciale che viene chiamata automaticamente quando un contratto riceve una transazione che non corrisponde a nessuna delle sue firme di funzione. A differenza della funzione receive, la funzione fallback non deve essere contrassegnata come pagabile e può essere definita più volte all'interno di un contratto.

Sia la funzione *receive* che la funzione *fallback* possono essere utilizzate per gestire le transazioni in arrivo a un contratto, ma la funzione receive è preferita per gestire l'etere in arrivo, mentre la funzione fallback è utilizzata per altri tipi di transazioni, come l'invio di dati a un contratto. È importante notare che nelle versioni di Solidity successive alla 0.6.0, si raccomanda di usare receive() invece di fallback() per gestire gli ether in entrata.

Nota di margine 🡪 \_; // this underscore indicates where the function code should be executed

Fino ad ora è stato usato Remix Ethereum per compilare; tuttavia:

* può solo gestire degli smart contract
* non si può integrare con altre parti di un progetto
* non può salvare file in locale senza un plugin
* tutte le funzionalità custom richiedono un plugin

Ora useremo *Hardhat*, basato su JavaScript.

È uno strumento open-source che fornisce una suite di funzioni e utilità per rendere il processo di sviluppo e di test più veloce, più facile e più snello.

Hardhat supporta una serie di funzioni per la creazione, il test e la distribuzione di contratti intelligenti. Alcune di queste funzioni includono:

* Compilazione di contratti intelligenti: Hardhat consente agli sviluppatori di compilare i loro contratti smart Solidity in bytecode che possono essere distribuiti sulla rete Ethereum.
* Test dei contratti: Hardhat fornisce una suite di strumenti di test che facilitano la scrittura e l'esecuzione di test per i contratti intelligenti.
* Debug: Hardhat include un debugger integrato che consente agli sviluppatori di analizzare il codice e di verificare i cambiamenti di stato durante l'esecuzione.
* Stima del gas: Hardhat fornisce uno strumento di stima del gas che può aiutare gli sviluppatori a ottimizzare i loro contratti per l'utilizzo del gas.
* Script di distribuzione: Hardhat consente agli sviluppatori di scrivere script che automatizzano la distribuzione dei contratti intelligenti sulla rete Ethereum.

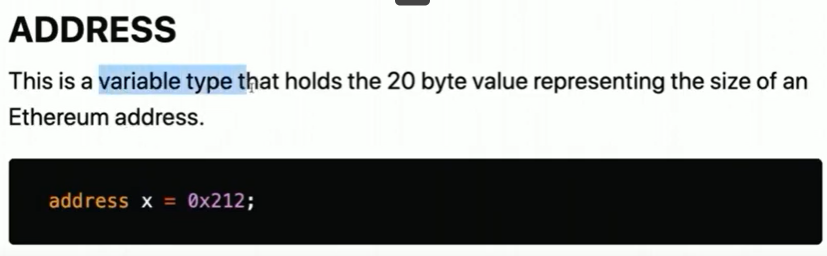
Oltre a queste caratteristiche, Hardhat si integra anche con altri strumenti e framework Ethereum popolari, come Truffle, Web3.js e Ethers.js. In questo modo è facile utilizzare Hardhat insieme a questi altri strumenti o migrare da uno strumento all'altro, se necessario.

Comandi base:

* Aggiunta al progetto 🡪 yarn add hardhat
* Compilazione 🡪 yard hardhat compile
* Deploy 🡪 yarn hardhat deploy
  + Su una testnet 🡪 yard hardhat deploy --network rinkeby
* Test 🡪 yarn hardhat test
* Coverage 🡪 yarn hardhat coverage

### Solidity in a nutshell

* All’inizio di un file:
  + Licenza 🡪 // SPDX-License-Identifier: MIT
  + Versione di Solidity 🡪 pragma solidity ^0.8.17;
* Gli indirizzi sono di tipo *address* ed hanno 20 byte



* *mapping*

Immagine che contiene testo

Descrizione generata automaticamente

* *memory* 🡪 Variabile temporanea che è in memoria ed esiste mentre la funzione viene chiamata
* *storage* 🡪 Variabile di stato (quindi, memorizzate permanentemente nella memoria del contratto) memorizzata sulle blockchain
  + Diversamente dalle funzioni, le variabili di stato non possono essere sovrascritte dichiarandole nuovamente nel contratto figlio.
* *calldata* 🡪 Locazione speciali di memoria che contengono argomenti di funzione
* *payable* 🡪 La funzione e gli indirizzi possono ricevere ether nel contratto
* *view* 🡪 Lo stato della funzione non sarà cambiato
* *pure* 🡪 La funzione dichiara che nessuna variabile di stato sarà cambiata o letta
* *modifier* 🡪 Codice eseguito prima o dopo la chiamata di una funzione. Vengono usati per
  + Restringere l’accesso (controllando ad esempio il chiamante sia il proprietario del contratto)
  + Validare gli input
  + Proteggersi dagli attacchi di reentrancy
* *event* 🡪 Permette il logging sulla blockchain Ethereum; spesso usati per aggiornare le interfacce ed ascoltare cosa succede sulla blockchain
* Modificatori
  + *public* - tutti possono accedervi
  + *external* - Non si può accedere internamente, solo esternamente
  + *internal* - solo questo contratto e i contratti da esso derivati possono accedervi
  + *private* - si può accedere solo da questo contratto
* Funzione e scrittura
  + *function nome (parametri eventuali) modificatori altre (virtual/pure/etc.) returns (tipo memory)*
* Ereditarietà
  + I contratti ereditano usando la keyword *is*
  + La funzione che verrà sovrascritta da un contratto figlio deve essere dichiarata come *virtual*
  + Le funzioni che sovrascrivono una funzione genitore devono utilizzare la keyword *override*.
  + I contratti padre possono essere richiamati direttamente o utilizzando la parola chiave *super*.
  + Utilizzando la parola chiave *super*, verranno richiamati tutti i contratti padre immediati.
* Invio Ether (tre funzioni: *transfer, end, call*)
  + Come inviare Ether?
    - È possibile inviare Ether ad altri contratti tramite
      * *transfer* (2300 gas, lancia errore)
      * *send* (2300 gas, restituisce bool)
      * *call* (inoltrare tutto il gas o impostare il gas, restituisce bool)
  + Come ricevere Ether?
    - Un contratto che riceve Ether deve avere almeno una delle funzioni seguenti
      * *receive() external payable*
      * *fallback() external payable*
      * *receive()* viene chiamata se msg.data è vuoto, altrimenti viene chiamata fallback().
  + Quale metodo si dovrebbe utilizzare?
    - *call* in combinazione con la protezione da rientranze è il metodo consigliato da utilizzare dopo dicembre 2019.
  + Per evitare la rientranza, occorre
    - effettuando tutte le modifiche allo stato prima di chiamare altri contratti
    - utilizzando il modificatore di guardia di rientranza
* fallback 🡪 funzione speciale che viene eseguita quando
  + viene chiamata una funzione che non esiste o
  + L'Etere viene inviato direttamente a un contratto, ma receive() non esiste o msg.data non è vuoto.
  + fallback ha un limite di 2300 gas quando viene chiamata da transfer o send.
* Quando si chiama una funzione, i primi 4 byte di calldata specificano quale funzione chiamare.
  + Questi 4 byte sono chiamati selettori di funzione.
  + Prendiamo ad esempio il codice qui sotto. Utilizza call per eseguire il trasferimento su un contratto all'indirizzo addr.
* addr.call(abi.encodeWithSignature("transfer(address,uint256)", 0xSomeAddress, 123))
  + I primi 4 byte restituiti da abi.encodeWithSignature(....) sono il selettore di funzione. Forse si può risparmiare una piccola quantità di gas se si precompila e si mette in linea il selettore di funzioni nel codice?
* Ecco come viene calcolato il selettore di funzioni.

// Identificatore di licenza SPDX: MIT

pragma solidity ^0.8.17;

contract FunctionSelector {

/\*

"transfer(indirizzo,uint256)"

0xa9059cbb

"transferFrom(indirizzo,indirizzo,uint256)"

0x23b872dd

\*/

function getSelector(string calldata \_func) external pure returns (bytes4) {

return bytes4(keccak256(bytes(\_func));

}

}

* La chiamata del contratto si ha:
  + Il modo più semplice è chiamarlo, come A.foo(x, y, z).
  + Un altro modo per chiamare altri contratti è quello di utilizzare la chiamata di basso livello con call. Questo metodo non è raccomandato.
* I contratti possono essere creati da altri contratti utilizzando la parola chiave new. Dalla 0.8.0, la nuova parola chiave supporta la funzione create2 specificando le opzioni di sale.
* *Library* 🡪 Le librerie sono simili ai contratti, ma non è possibile dichiarare alcuna variabile di stato e non è possibile inviare etere.
  + Una libreria è incorporata nel contratto se tutte le funzioni della libreria sono interne.
  + In caso contrario, la libreria deve essere distribuita e poi collegata prima che il contratto venga distribuito.
* *keccak256* 🡪 calcola l'hash Keccak-256 dell'input. Alcuni casi d'uso sono:
  + Creazione di un ID univoco deterministico da un input
  + Schema Commit-Reveal
  + Firma crittografica compatta (firmando l'hash invece di un input più grande)
* Alcune tecniche di risparmio di gas.
  + Sostituzione della memoria con i dati di chiamata
  + Caricamento della variabile di stato in memoria
  + Sostituzione del ciclo for i++ con ++i
  + Caching degli elementi dell'array
  + Cortocircuito
* L'overflow e l'underflow dei numeri in Solidity 0.8 generano un errore. Questo può essere disabilitato usando unchecked.
  + Disabilitando il controllo dell'overflow e dell'underflow, si risparmia gas.

## Framework JavaScript per interazione blockchain

### Ethers.js

Ethers.js è uno degli strumenti più necessari per gli sviluppatori che cercano di costruire una dApp o un'applicazione basata su qualsiasi Blockchain nella loro fase iniziale di sviluppo. Fornisce un'interfaccia semplice che rende il lavoro dello sviluppatore estremamente facile ed efficace. Con il suo aiuto possono interagire facilmente con la Blockchain sottostante e possono anche fare richieste alla Blockchain e generare risposte attraverso di essa.

Esempi di codice della sezione presenti nella repo: <https://github.com/dappuniversity/ethers_examples>

Alcune caratteristiche rilevanti:

* API facile da usare: Ethers.js fornisce un'API semplice ma potente che consente agli sviluppatori di interagire facilmente con la rete Ethereum e di eseguire operazioni comuni. È leggera e la sua API è abbastanza semplice da consentire a qualsiasi sviluppatore di iniziare il proprio percorso di sviluppo.
* Interfaccia semplice e facile da interagire: Ethers.js riduce la complessità della rete Ethereum, rendendo più facile per gli sviluppatori concentrarsi sulla creazione delle loro applicazioni. Anche altri framework come Web3.js sono validi, ma la loro interfaccia complessa non è facile da comprendere e utilizzare per i principianti.
* Ampiamente utilizzato dalla comunità di Ethereum: È popolare all'interno della comunità di Ethereum, il che rende facile per gli sviluppatori trovare e cercare la sua documentazione e le sue pratiche. Inoltre, aiuta a condividere le proprie conoscenze con gli altri.
* Supporto sia per Node.js che per i browser web: Ethers.js supporta sia Node.js che i browser web, rendendolo uno strumento versatile per lo sviluppo di Ethereum.

Ethers.js è caratterizzato da un bundle di dimensioni ridotte, da una documentazione ampia e semplice e da una struttura API di facile utilizzo. Inoltre, è intuitivo e semplice da usare. Inoltre, oltre a JavaScript, ethers.js supporta TypeScript (TS). Tutti questi vantaggi, uniti alle caratteristiche sopra elencate, rendono ethers.js una libreria molto interessante per molti sviluppatori Web3.

Vediamo una sintesi dei principali vantaggi di ethers.js:

* Dimensioni minime - ethers.js è minuscolo, solo 88 KB compressi e 284 KB non compressi.
* Include un'ampia documentazione.
* Viene fornito con un'ampia raccolta di casi di test mantenuti.
* Ethers.js include i file di definizione e i sorgenti completi di TS - è completamente pronto per TypeScript.
* Viene fornito con una licenza MIT open-source che include tutte le dipendenze.

Oltre a questi vantaggi specifici di ethers.js, imparare a usare ethers.js porta alcuni vantaggi universali. Dopo tutto, la blockchain è qui per restare ed è pronta a cambiare radicalmente il modo in cui il mondo opera. Pertanto, potete utilizzare questo tutorial su ethers.js sapendo che vi aiuterà a imparare a diventare parte di questa soluzione.

Inoltre, alcuni degli esempi più comuni di ethers.js riguardano la creazione di whale alert (dati in tempo reale sulle transazioni blockchain), la costruzione di bot automatizzati, il monitoraggio delle collezioni NFT, le notifiche di gioco Web3, ecc. In definitiva, quando si ascoltano gli eventi della catena con ethers.js o con altri strumenti affidabili, l'obiettivo principale è quello di eseguire automaticamente azioni in risposta a specifici eventi della catena. Quindi, il vostro obiettivo deve essere quello di scegliere uno strumento che vi permetta di farlo in modo efficace ed efficiente.

Un piccolo esempio molto semplice.

* Quindi, in prima battuta, bisogna assicurarsi che il file NodeJS utilizzi ethers.js. Poi, bisogna importare l'interfaccia binaria dell'applicazione (ABI), che è specifica per ogni smart contract. Detto questo, la funzione asincrona "getTransfer" merita la massima attenzione. Dopo tutto, quest'ultima si occupa di ascoltare la blockchain.
* Utilizzando il nostro script di esempio ethers.js, vi concentrerete sull'indirizzo del contratto USDC. Naturalmente, è possibile convertirlo in altri esempi ethers.js utilizzando altri indirizzi di contratto. Inoltre, la funzione "getTransfer" si basa sull'endpoint "WebSocketProvider" della libreria ethers.js. Utilizzando questo endpoint, è possibile definire il provider di nodi che si desidera utilizzare. Per far sì che ciò funzioni, è necessario ottenere anche la chiave di tale provider, che è un aspetto importante dell'utilizzo di ethers.js. L'esempio di codice che segue si concentra su Alchemy. Tuttavia, si dovrebbe utilizzare un provider di nodi che supporti la catena o le catene su cui ci si vuole concentrare.
* Oltre all'indirizzo del contratto e al provider, la funzione "getTransfer" accetta anche un ABI. Infine, è necessario impostare un ascoltatore di "transfer". Quest'ultimo registrerà anche i dettagli relativi alla catena.

const ethers = require("ethers");

const ABI = require("./abi.json");

require("dotenv").config();

async function getTransfer(){

const usdcAddress = "0xA0b86991c6218b36c1d19D4a2e9Eb0cE3606eB48"; ///USDC Contract

const provider = new ethers.providers.WebSocketProvider(

`wss://eth-mainnet.g.alchemy.com/v2/${process.env.ALCHEMY\_KEY}`

);

const contract = new ethers.Contract(usdcAddress, ABI, provider);

contract.on("Transfer", (from, to, value, event)=>{

let transferEvent ={

from: from,

to: to,

value: value,

eventData: event,

}

console.log(JSON.stringify(transferEvent, null, 4))

})

}

getTransfer()

La lettura del bilancio di un particolare account avviene ad esempio come segue:

const { ethers } = require("ethers");

require('dotenv').config()

//we take it from https://app.infura.io/dashboard/ethereum/9f904279fdbb4f5e881c41b51bad948d/settings/endpoints

const provider = new ethers.providers.JsonRpcProvider(`https://mainnet.infura.io/v3/${process.env.INFURA\_ID}`)

//rich contract from EtherScan

const address = '0x73BCEb1Cd57C711feaC4224D062b0F6ff338501e'

const main = async () => {

    const balance = await provider.getBalance(address)

    console.log('Balance: ', balance.toString())

}

main()

La lettura degli smart contract, basandosi su una ABI dello standard ERC-20 che permette la lettura delle funzioni stabilite dallo stesso standard appare come segue:

const { ethers } = require("ethers");

require('dotenv').config()

//we take it from https://app.infura.io/dashboard/ethereum/9f904279fdbb4f5e881c41b51bad948d/settings/endpoints

const provider = new ethers.providers.JsonRpcProvider(`https://mainnet.infura.io/v3/${process.env.INFURA\_ID}`)

//rich contract from EtherScan

const address = '0x6B175474E89094C44Da98b954EedeAC495271d0F' // DAI (which is a stablecoin) Contract

const ERC\_20\_ABI = [

    // Read-Only Functions

    "function name() view returns (string)",

    "function symbol() view returns (string)",

    "function decimals() view returns (uint)",

    "function totalSupply() view returns (uint)",

    "function balanceOf(address) view returns (uint)",

    "function allowance(address, address) view returns (uint)",

    // Write Functions

    "function transfer(address, uint) returns (bool)",

    "function approve(address, uint) returns (bool)",

    "function transferFrom(address, address, uint) returns (bool)",

]

const contract = new ethers.Contract(address, ERC\_20\_ABI, provider)

const main = async () => {

    const name = await contract.name()

    const symbol = await contract.symbol()

    const decimals = await contract.decimals()

    const totalSupply = await contract.totalSupply()

    console.log('Contract Address: ', address)

    console.log('Name: ', name)

    console.log('Symbol: ', symbol)

    console.log('Decimals: ', decimals.toString())

    console.log('Total Supply: ', totalSupply.toString())

    const balance = await contract.balanceOf('0x73BCEb1Cd57C711feaC4224D062b0F6ff338501e')

    console.log('Balance: ', balance.toString())

    console.log(`Balance Formatted: ${ethers.utils.formatEther(balance)}\n`)

}

main()

La creazione di una chiave privata, con annesso completamento base di una transazione è come segue:

const { ethers } = require("ethers");

require('dotenv').config()

//we take it from https://app.infura.io/dashboard/ethereum/9f904279fdbb4f5e881c41b51bad948d/settings/endpoints

const provider = new ethers.providers.JsonRpcProvider(`https://mainnet.infura.io/v3/${process.env.INFURA\_ID}`)

//const account = ethers.Wallet.createRandom() //create a rondom one

//need to have some ether in the account, using Sepolia since it's the only one that receives ethers on the testnet from Chainlink,

//the others are deprecated

const account\_sender = new ethers.Wallet(process.env.SENDER, provider)

const account\_receiver = new ethers.Wallet(process.env.RECEIVER, provider)

//from advanced option on MetaMask, we get the private key

const privateKey = process.env.PRIVATE\_KEY

//we create a wallet from the private key

const wallet = new ethers.Wallet(privateKey, provider)

const main = async () => {

    const balance\_sender = await wallet.getBalance(account\_sender)

    console.log("Balance before transaction of sender: ", ethers.utils.formatEther(balance\_sender))

    const balance\_receiver = await wallet.getBalance(account\_receiver)

    console.log("Balance before transaction of receiver: ", ethers.utils.formatEther(balance\_receiver))

    const tx = await wallet.sendTransaction({

        to: account\_receiver.address,

        value: ethers.utils.parseEther("0.01")

    })

    console.log("Transaction hash: ", tx.hash)

    const receipt = await tx.wait()

    console.log("Transaction receipt: ", receipt)

    const balance\_sender\_after = await wallet.getBalance(account\_sender)

    console.log("Balance after transaction of sender: ", ethers.utils.formatEther(balance\_sender\_after))

    const balance\_receiver\_after = await wallet.getBalance(account\_receiver)

    console.log("Balance after transaction of receiver: ", ethers.utils.formatEther(balance\_receiver\_after))

}

main()

In questo caso, invece, abbiamo i dettagli di una transazione completa di tutti i dettagli utili:

const { ethers } = require("ethers");

require('dotenv').config()

//we take it from https://app.infura.io/dashboard/ethereum/9f904279fdbb4f5e881c41b51bad948d/settings/endpoints

const provider = new ethers.providers.JsonRpcProvider(`https://mainnet.infura.io/v3/${process.env.INFURA\_ID}`)

//const account = ethers.Wallet.createRandom() //create a rondom one

//need to have some ether in the account, using Sepolia since it's the only one that receives ethers on the testnet from Chainlink,

//the others are deprecated

const account\_sender = new ethers.Wallet(process.env.SENDER, provider)

const account\_receiver = new ethers.Wallet(process.env.RECEIVER, provider)

//from advanced option on MetaMask, we get the private key

const privateKey = process.env.PRIVATE\_KEY

//we create a wallet from the private key

const wallet = new ethers.Wallet(privateKey, provider)

const ERC\_20\_ABI = [

    // Some details about the token

    "function name() view returns (string)",

    "function symbol() view returns (string)",

    "function decimals() view returns (uint8)",

    "function totalSupply() view returns (uint256)",

    // Get the account balance

    "function balanceOf(address) view returns (uint256)",

    // Send some of your tokens to someone else

    "function transfer(address to, uint amount)",

    // An event triggered whenever anyone transfers to someone else

    "event Transfer(address indexed from, address indexed to, uint amount)"

]

const contract = new ethers.Contract(process.env.CONTRACT\_ADDRESS, ERC\_20\_ABI, provider)

const main = async () => {

    const name = await contract.name()

    const symbol = await contract.symbol()

    const decimals = await contract.decimals()

    const totalSupply = await contract.totalSupply()

    const balance = await wallet.getBalance(account\_sender)

    console.log("Name: ", name)

    console.log("Symbol: ", symbol)

    console.log("Decimals: ", decimals)

    console.log("Total supply: ", totalSupply.toString())

    console.log("Balance of sender: ", balance.toString())

    const tx = await contractWithWallet.transfer(account\_receiver.address, ethers.utils.parseEther("0.01"))

    console.log("Transaction hash: ", tx.hash)

    const receipt = await tx.wait()

    console.log("Transaction receipt: ", receipt)

    const balance\_sender\_after = await wallet.getBalance(account\_sender)

    console.log("Balance after transaction of sender: ", balance\_sender\_after.toString())

    const balance\_receiver\_after = await wallet.getBalance(account\_receiver)

    console.log("Balance after transaction of receiver: ", balance\_receiver\_after.toString())

}

main()

Ogni dettaglio della transazioni eseguite rimane presente su EtherScan, accessibile anche tramite MetaMask. Il flusso completo di trasferimento di un contratto, completo di filtraggio blocchi e successiva approvazione segue:

const { ethers } = require("ethers");

require('dotenv').config()

//we take it from https://app.infura.io/dashboard/ethereum/9f904279fdbb4f5e881c41b51bad948d/settings/endpoints

const provider = new ethers.providers.JsonRpcProvider(`https://mainnet.infura.io/v3/${process.env.INFURA\_ID}`)

const ERC\_20\_ABI=[

    // Read-Only Functions

    "function name() view returns (string)",

    "function symbol() view returns (string)",

    "function decimals() view returns (uint)",

    "function totalSupply() view returns (uint)",

    "function balanceOf(address) view returns (uint)",

    "function allowance(address, address) view returns (uint)",

    // Write Functions

    "function transfer(address, uint) returns (bool)",

    "function approve(address, uint) returns (bool)",

    "function transferFrom(address, address, uint) returns (bool)",

    // Events - These are emitted when state changes

    "event Transfer(address indexed from, address indexed to, uint value)",

    "event Approval(address indexed owner, address indexed spender, uint value)"

]

const address = '0x6B175474E89094C44Da98b954EedeAC495271d0F' // DAI Contract

const contract = new ethers.Contract(address, ERC20\_ABI, provider)

const main = async () => {

    const block = await provider.getBlockNumber()

    // Filter for Transfer events to and from an address

    const transferFilter = contract.filters.Transfer(null, '0x73BCEb1Cd57C711feaC4224D062b0F6ff338501e') // here we can put the address we want to filter

    const transferEvents = await contract.queryFilter(transferFilter, block - 1000, block) // we can also specify the block range we want to filter

    console.log('Transfer Events: ', transferEvents)

    // Filter for Approval events to and from an address

    const approvalFilter = contract.filters.Approval('0x73BCEb1Cd57C711feaC4224D062b0F6ff338501e', null)

    const approvalEvents = await contract.queryFilter(approvalFilter)

    console.log('Approval Events: ', approvalEvents)

}

main()

Qui di seguito, i dettagli dei singoli blocchi:

const { ethers } = require("ethers");

require('dotenv').config()

//we take it from https://app.infura.io/dashboard/ethereum/9f904279fdbb4f5e881c41b51bad948d/settings/endpoints

const provider = new ethers.providers.JsonRpcProvider(`https://mainnet.infura.io/v3/${process.env.INFURA\_ID}`)

const main = async () => {

    const block = await provider.getBlock(1000000)

    console.log('Block: ', block)

    const blockNumber = await provider.getBlockNumber()

    console.log('Block number: ', blockNumber)

    const blockWithTransactions = await provider.getBlockWithTransactions(1000000)

    console.log('Block with transactions: ', blockWithTransactions)

}

main()

### web3.js

Tutta la sezione ha come riferimento di massima: <https://www.dappuniversity.com/articles/web3-js-intro>

Lo sviluppo di applicazioni blockchain con Ethereum presenta diversi aspetti:

* Sviluppo di contratti intelligenti - scrittura di codice che viene distribuito sulla blockchain con il linguaggio di programmazione Solidity.
* Sviluppo di siti web o client che interagiscono con la blockchain - scrittura di codice che legge e scrive dati dalla blockchain con gli smart contract.

Web3.js consente di adempiere alla seconda responsabilità: sviluppare client che interagiscono con la blockchain Ethereum. Si tratta di una raccolta di librerie che consentono di eseguire azioni come l'invio di Ether da un conto a un altro, la lettura e la scrittura di dati da contratti intelligenti, la creazione di contratti intelligenti e molto altro ancora.

* Web3.js dialoga con la Blockchain di Ethereum con il protocollo JSON RPC, che sta per "Remote Procedure Call". Ethereum è una rete peer-to-peer di nodi che memorizza una copia di tutti i dati e del codice sulla blockchain. Web3.js ci permette di fare richieste a un singolo nodo Ethereum con JSON RPC per leggere e scrivere dati sulla rete. È un po' come usare jQuery con un'API JSON per leggere e scrivere dati su un server web.
* Dipendenze

Ci sono alcune dipendenze che vi aiuteranno a iniziare a sviluppare con Web3.js.

* Gestore di pacchetti Node (NPM)

La prima dipendenza di cui abbiamo bisogno è Node Package Manager, o NPM, che viene fornito con Node.js. Per verificare se Node è già installato, basta andare sul proprio terminale e digitare

$ node -v

* Libreria Web3.js

È possibile installare la libreria Web3.js con NPM nel terminale in questo modo:

$ npm install web3

* URL RPC di Infura

Per connettersi a un nodo Ethereum con JSON RPC sulla rete principale, è necessario accedere a un nodo Ethereum. Ci sono alcuni modi per farlo. Ad esempio, si potrebbe gestire il proprio nodo Ethereum con Geth o Parity. Ma questo richiede di scaricare molti dati dalla blockchain e di mantenerli sincronizzati. Questo è un enorme mal di testa se avete mai provato a farlo prima.

Per comodità, è possibile utilizzare Infura per accedere a un nodo Ethereum senza doverne gestire uno. Infura è un servizio che fornisce gratuitamente un nodo Ethereum remoto. Tutto quello che dovete fare è iscrivervi e ottenere una chiave API e l'URL RPC per la rete a cui volete connettervi.

Una volta effettuata l'iscrizione, l'URL RPC di Infura dovrebbe essere simile a questo:

<https://mainnet.infura.io/YOUR_INFURA_API_KEY>

* Controllare i saldi dei conti

Ora che tutte le dipendenze sono installate, si può iniziare a sviluppare con Web3.js! Per prima cosa, si dovrebbe avviare la console di Node nel terminale, come segue:

$ node

All'interno della console di Node, è possibile richiedere Web3.js in questo modo:

const Web3 = require('web3')

Ora si ha accesso a una variabile in cui è possibile creare una nuova connessione Web3! Prima di generare una connessione Web3, dobbiamo prima assegnare l'URL di Infura a una variabile come questa:

const rpcURL = "https://mainnet.infura.io/YOUR\_INFURA\_API\_KEY"

Assicurarsi di sostituire YOUR\_INFURA\_API\_KEY con la chiave API Infura ottenuta in precedenza. Ora è possibile istanziare una connessione Web3 in questo modo:

const web3 = new Web3(rpcURL)

Ora avete una connessione Web3 attiva che vi permetterà di parlare con la rete principale di Ethereum! Utilizziamo questa connessione per controllare il saldo del conto: 0x90e63c3d53E0Ea496845b7a03ec7548B70014A91. Possiamo vedere quanto Ether detiene questo conto controllando il suo saldo con web3.eth.getBalance().

Per prima cosa, assegniamo l'indirizzo a una variabile:

const account = "0x90e63c3d53E0Ea496845b7a03ec7548B70014A91"

Ora controlliamo il saldo del conto in questo modo:

web3.eth.getBalance(address, (err, wei) => {

balance = web3.utils.fromWei(wei, 'ether')

})

Lasciate che vi spieghi questo codice. Per prima cosa, si controlla il saldo chiamando web3.eth.getBalance(), che accetta una funzione di callback con due argomenti, un errore e il saldo stesso. Per ora ignoreremo l'argomento dell'errore e faremo riferimento al saldo con l'argomento wei. Ethereum esprime i saldi in Wei, che è la più piccola suddivisione di Ether, un po' come un piccolo penny. Possiamo convertire questo saldo in Ether con web3.utils.fromWei(wei, 'ether').

Riassumendo, avremo il seguente codice:

const Web3 = require('web3')

const rpcURL = '' // Il vostro URL RPC va qui

const web3 = new Web3(rpcURL)

const address = '' // L'indirizzo del vostro account va qui

web3.eth.getBalance(address, (err, wei) => {

balance = web3.utils.fromWei(wei, 'ether')

})

* Leggere I dati degli smart contract

Per leggere i dati dagli smart contract con Web3.js, abbiamo bisogno di due cose:

* una rappresentazione in JavaScript dello smart contract con cui vogliamo interagire
* un modo per chiamare le funzioni dello smart contract durante la lettura dei dati.

Possiamo ottenere una rappresentazione JavaScript di uno smart contract Ethereum con la funzione web3.eth.Contract(). Questa funzione si aspetta due argomenti: uno per l'ABI dello smart contract e uno per l'indirizzo dello smart contract.

L'ABI di uno smart contract sta per "Abstract Binary Interface" (Interfaccia binaria astratta) ed è un array JSON che descrive il funzionamento di uno specifico smart contract. Ecco un esempio di ABI:

const abi = [{"constant":true, "input":[], "name": "mintingFinished", "outputs":[{"name":", "type": "bool"}], "payable":false, "type": "function"},{"constant":true, "input":[], "name": "name", "outputs":[{"name":", "type": "stringa"}], "pagabile":false, "tipo": "funzione"},{"constant":false, "ingressi":[{"nome":"\_spender", "tipo": "indirizzo"},{"nome":"\_valore", "tipo": "uint256"}], "nome": "approvare", "uscite":[], "pagabile":false, "tipo": "funzione"},{"constant": true,"inputs":[],"name":"totalSupply","outputs":[{"name":"","type":"uint256"}],"payable":false,"type":"function"},{"constant":false,"inputs":[{"name":"\_from","type":"address"},{"name":"\_to","type":"address"},{"name": "\_valore", "tipo": "uint256"}], "nome": "transferFrom", "uscite":[], "pagabile":false, "tipo": "function"},{"constant":true, "ingressi":[], "nome": "decimali", "uscite":[{"nome":", "tipo": "uint256"}], "pagabile":false, "tipo": "function"},{"constant":false, "input":[], "name": "unpause", "output":[{"name":"", "type": "bool"}], "payable":false, "type": "function"},{"constant":false, "input":[{"name":"\_to", "type": "address"},{"name":"\_amount", "type": "uint256"}],"name":"mint","outputs":[{"name":"","type":"bool"}],"payable":false,"type":"function"},{"constant":true,"inputs":[],"name":"paused","outputs":[{"name":"","type":"bool"}],"payable":false,"type": "function"},{"constant":true, "input":[{"name":"\_owner", "type": "address"}], "name": "balanceOf", "output":[{"name": "balance", "type": "uint256"}], "payable":false, "type": "function"},{"constant":false, "input":[], "name": "finishMinting", "output":[{"name":"", "type": "bool"}], "payable":false, "type": "function"},{"constant":false, "input":[], "name": "pause", "output":[{"name":"", "type": "bool"}], "payable":false, "type": "function"},{"constant":true, "input": [], "nome": "proprietario", "uscite":[{"nome":"", "tipo": "indirizzo"}], "pagabile":false, "tipo": "funzione"},{"costante":true, "ingressi":[], "nome": "simbolo", "uscite":[{"nome":"", "tipo": "stringa"}], "pagabile":false, "tipo": "funzione"},{"costante": false,"inputs":[{"name":"\_to","type":"address"},{"name":"\_value","type":"uint256"}],"name":"transfer","outputs":[],"payable":false,"type":"function"},{"constant":false,"inputs":[{"name":"\_to","type":"address"},{"name": "\_amount","type":"uint256"},{"name":"\_releaseTime","type":"uint256"}],"name":"mintTimelocked","outputs":[{"name":"","type":"address"}],"payable":false,"type":"function"},{"constant":true,"inputs":[{"name":"\_owner","type": "indirizzo"},{"nome":"\_spendente", "tipo": "indirizzo"}], "nome": "indennità", "uscite":[{"nome": "rimanente", "tipo": "uint256"}], "pagabile":false, "tipo": "funzione"},{"costante":false, "ingressi":[{"nome": "nuovoProprietario", "tipo": "indirizzo"}], "nome": "trasferimentoProprietà", "uscite":[], "pagabile":false, "tipo": "funzione"},{"anonymous":false, "input":[{"indexed":true, "name": "to", "type": "address"},{"indexed":false, "name": "value", "type": "uint256"}], "name": "Menta", "tipo": "evento"},{"anonymous":false, "input":[], "name": "MentaFinita", "type": "event"},{"anonymous":false, "input":[], "name": "Pausa", "type": "event"},{"anonymous":false, "input":[], "name": "Unpause", "type": "evento"},{"anonymous":false, "input":[{"indexed":true, "name": "owner", "type": "address"},{"indexed":true, "name": "spender", "type": "address"},{"indexed":false, "name": "value", "type": "uint256"}], "name": "Approval", "type": "evento"},{"anonymous":false, "input":[{"indexed":true, "name": "from", "type": "address"},{"indexed":true, "name": "to", "type": "address"},{"indexed":false, "name": "value", "type": "uint256"}], "name": "Transfer", "type": "event"}].

È un array ininterrotto molto lungo. Questo esempio è l'ABI per il token OmiseGo (OMG), che implementa lo standard ERC-20 (se non conoscete questo standard, potete guardare questo video). Potete trovare maggiori dettagli su questo token, compresi il suo abi e il suo indirizzo su Etherscan. Utilizzeremo questo ABI per il resto dell'esempio.

Già che ci siamo, procederò a memorizzare l'indirizzo del token OMG dalla rete principale di Ethereum:

const address = "0xd26114cd6EE289AccF82350c8d8487fedB8A0C07"

Ora che abbiamo assegnato entrambi i valori, possiamo creare una rappresentazione JavaScript completa dello smart contract OMG token in questo modo:

const contract = new web3.eth.Contract(abi, address)

Ora che la prima parte di questa lezione è stata completata, è necessario adempiere alla seconda parte: leggere i dati dallo smart contract chiamando le sue funzioni. Tutte le funzioni dello smart contract sono elencate nello spazio dei nomi contract.methods all'interno del contratto Web3 assegnato. Per esempio, possiamo chiamare contract.methods.myFunction() se il contratto implementa myFunction().

Ottimo! Quindi, in teoria, possiamo chiamare qualsiasi funzione implementata dal contratto smart. Ma come facciamo a sapere quali funzioni implementa? Ad esempio, potremmo registrare contract.methods nella console e vedere cosa viene restituito (potete vedere come lo faccio nel video qui sopra). Tuttavia, poiché questo smart contract implementa lo standard ERC-20, sappiamo che implementa diverse funzioni come totalSupply(), name(), symbol() e balanceOf(). Possiamo leggere ciascuno di questi valori singolarmente, in questo modo che segue.

In primo luogo, l'offerta totale di tutti i token OMG esistenti:

contract.methods.totalSupply().call((err, result) => { console.log(result) })

// > 140245398

In secondo luogo, il nome del token OMG:

contract.methods.name().call((err, result) => { console.log(result) })

// > Token OMG

Terzo, il simbolo del token OMG:

contract.methods.symbol().call((err, result) => { console.log(result) })

// > OMG

Infine, possiamo controllare il saldo di un determinato conto. Ho cercato un titolare di OMG ricco su Etherscan e ho trovato questo indirizzo 0xd26114cd6EE289AccF82350c8d8487fedB8A0C07. Possiamo controllare il saldo di questo conto in questo modo:

contract.methods.balanceOf('0xd26114cd6EE289AccF82350c8d8487fedB8A0C07').call((err, result) => { console.log(result) })

// > Un numero molto grande...

E questo è tutto! Ecco quanto è facile leggere i dati dagli smart contract con Web3.js.

* Nelle transazioni Ethereum

Oltre all'apprendimento di Web3.js, lo scopo di questa lezione è quello di aiutarvi a comprendere le basi del funzionamento delle transazioni sulla blockchain di Ethereum. Ogni volta che si crea una transazione, si scrivono dati sulla blockchain e si aggiorna il suo stato. Ci sono diversi modi per farlo, come l'invio di Ether da un conto a un altro, la chiamata di una funzione dello smart contract che scrive dati e il deploy di uno smart contract sulla blockchain. Possiamo comprendere meglio questi concetti eseguendo queste azioni con la libreria Web3.js e osservando come funziona ogni fase.

Per trasmettere le transazioni alla rete, dobbiamo prima firmarle. Per farlo, utilizzerò una libreria JavaScript aggiuntiva, chiamata ethereumjs-tx. È possibile installare questa dipendenza dalla riga di comando in questo modo:

$ npm install ethereumjs-tx

Il motivo per cui utilizzeremo questa libreria è che vogliamo firmare tutte le transazioni localmente. Se gestissimo il nostro nodo Ethereum in locale, potremmo sbloccare un account memorizzato in loco e firmare tutte le nostre transazioni in loco. Se così fosse, non avremmo necessariamente bisogno di usare questa libreria. Tuttavia, in questo tutorial stiamo utilizzando un nodo remoto ospitato da Infura. Sebbene Infura sia un servizio affidabile, vogliamo comunque firmare le transazioni localmente piuttosto che dare al nodo remoto la gestione delle nostre chiavi private.

Questo è esattamente ciò che faremo in questa lezione. Vi mostrerò come creare la transazione grezza, firmarla, quindi inviarla e trasmetterla alla rete! Per fare questo, creerò un semplice file app.js per eseguire il codice di questa lezione, invece di fare tutto nella console.

All'interno del file app.js, per prima cosa richiederemo la libreria appena installata in questo modo:

var Tx = require('ethereumjs-tx')

Quindi, configureremo una connessione Web3 come abbiamo fatto nelle lezioni precedenti:

const Web3 = require('web3')

const web3 = new Web3('https://ropsten.infura.io/YOUR\_INFURA\_API\_KEY')

Notate che stiamo usando la rete di test di Ropsten, che è diversa dalla rete principale di Ethereum che abbiamo usato nella lezione precedente. Vogliamo usare una rete di prova perché tutte le transazioni costano gas sotto forma di Ether. Possiamo utilizzare Ether falsi sulla rete di prova di Ropsten senza preoccuparci di spendere denaro. È possibile ottenere Ether falso da un rubinetto sulla rete di test Ropsten con un rubinetto. Ecco due faucet che si possono utilizzare:

* <http://faucet.ropsten.be:3001/>
* <https://faucet.metamask.io/>

In questa lezione creeremo una transazione che invia falsi Ether da un conto a un altro. Per farlo, avremo bisogno di due conti e delle loro chiavi private. È possibile creare nuovi account con Web3.js in questo modo:

web3.eth.accounts.create()

// > {

// indirizzo: "0xb8CE9ab6943e0eCED004cDe8e3bBed6568B2Fa01",

// privateKey: "0x348ce564d427a3311b6536bbcff9390d69395b06ed6c486954e971d960fe8709",

// signTransaction: function(tx){...},

// sign: function(data){...},

// encrypt: function(password){...}

// }

Una volta creati entrambi i conti, assicuratevi di caricarli con Ether falso da un rubinetto. Ora, li assegneremo alle variabili del nostro script in questo modo:

const account1 = '0xb8CE9ab6943e0eCED004cDe8e3bed6568B2Fa01'

const account2 = '0xb8CE9ab6943e0eCED004cDe8e3bBed6568B2Fa02'

Assicurarsi di utilizzare gli account generati, poiché questi account non funzioneranno per questa lezione. Ora, salviamo le chiavi private nell'ambiente in questo modo:

export PRIVATE\_KEY\_1='la vostra chiave privata 1'

export PRIVATE\_KEY\_1='la tua chiave privata 2 qui'

Vogliamo salvare queste chiavi private nell'ambiente, in modo da evitare di inserirle nel nostro file. È una cattiva pratica esporre le chiavi private in questo modo. Che cosa succederebbe se per sbaglio le caricassimo nei sorgenti di un progetto reale? Qualcuno potrebbe rubare il nostro Ether! Ora vogliamo leggere queste chiavi private dal nostro ambiente e memorizzarle in variabili. Possiamo farlo con l'oggetto globale process in NodeJS, in questo modo:

const privateKey1 = process.env.PRIVATE\_KEY\_1

const privateKey2 = process.env.PRIVATE\_KEY\_2

Per firmare le transazioni con le chiavi private, dobbiamo convertirle in una stringa di dati binari con un Buffer, un modulo disponibile a livello globale in NodeJS. Possiamo farlo in questo modo:

const privateKey1 = Buffer.from(process.env.PRIVATE\_KEY\_1)

const privateKey1 = Buffer.from(process.env.PRIVATE\_KEY\_2)

Bene, ora abbiamo impostato tutte le nostre variabili! So che alcune di queste cose potrebbero essere un po' confuse a questo punto.

A questo punto, vogliamo fare alcune cose:

* Creare un oggetto transazione
* Firmare la transazione
* Trasmettere la transazione alla rete

Possiamo costruire l'oggetto transazione in questo modo:

const txObject = {

nonce: web3.utils.toHex(txCount),

to: account2,

valore: web3.utils.toHex(web3.utils.toWei('0.1', 'ether')),

gasLimit: web3.utils.toHex(21000),

gasPrice: web3.utils.toHex(web3.utils.toWei('10', 'gwei'))

}

Lasciate che vi spieghi questo codice. Stiamo costruendo un oggetto che contiene tutti i valori necessari per generare una transazione, come nonce, to, value, gasLimit e gasPrice. Analizziamo ciascuno di questi valori:

* nonce - è il conteggio delle transazioni precedenti per il conto in questione. Assegneremo il valore di questa variabile tra poco. Dobbiamo anche convertire questo valore in esadecimale. Possiamo farlo con l'utility Web3.js web3.utils.toHex()
* to - il conto a cui stiamo inviando Ether.
* value - la quantità di Ether che vogliamo inviare. Questo valore deve essere espresso in Wei e convertito in esadecimale. Possiamo convertire il valore in Wei con l'utilità web3.js web3.utils.toWei().
* gasLimit - è la quantità massima di gas consumata dalla transazione. Una transazione di base come questa costa sempre 21000 unità di gas, quindi useremo questo valore.
* gasPrice - è l'importo che vogliamo pagare per ogni unità di gas. Qui utilizzeremo 10 Gwei.

Si noti che non c'è un campo from in questo oggetto transazione. Questo verrà dedotto quando firmeremo la transazione con la chiave privata di account1.

Ora assegniamo il valore della variabile nonce. Possiamo ottenere il nonce della transazione con la funzione web3.eth.getTransactionCount(). Avvolgeremo tutto il nostro codice all'interno di una funzione di callback come questa:

web3.eth.getTransactionCount(account1, (err, txCount) => {

const txObject = {

nonce: web3.utils.toHex(txCount),

to: account2,

valore: web3.utils.toHex(web3.utils.toWei('0.1', ‘ether’)),

gasLimit: web3.utils.toHex(21000),

gasPrice: web3.utils.toHex(web3.utils.toWei('10', 'gwei'))

}

})

Ed ecco l'oggetto transazione completato! Abbiamo completato il passo 1. :) Ora dobbiamo passare al passo 2, in cui firmiamo la transazione. Possiamo farlo in questo modo:

const tx = new Tx(txObject)

tx.sign(privateKey1)

const serializedTx = tx.serialize()

const raw = '0x' + serializedTx.toString('hex')

Qui usiamo la libreria etheremjs-tx per creare un nuovo oggetto Tx. Usiamo questa libreria anche per firmare la transazione con privateKey1. Quindi, serializziamo la transazione e la convertiamo in una stringa esadecimale, in modo da poterla passare a Web3.

Infine, inviamo la transazione firmata e serializzata alla rete di test con la funzione web3.eth.sendSignedTransaction() in questo modo:

web3.eth.sendSignedTransaction(raw, (err, txHash) => {

console.log('txHash:', txHash)

})

Ed ecco fatto! Questo è il passo finale di questa lezione, che invia la transazione e la trasmette alla rete. A questo punto, il file app.js completato dovrebbe avere questo aspetto:

var Tx = require('ethereumjs-tx')

const Web3 = require('web3')

const web3 = new Web3('https://ropsten.infura.io/YOUR\_INFURA\_API\_KEY')

const account1 = '' // L'indirizzo del vostro conto 1

const account2 = '' // L'indirizzo del vostro conto 2

const privateKey1 = Buffer.from('YOUR\_PRIVATE\_KEY\_1', 'hex')

const privateKey2 = Buffer.from('YOUR\_PRIVATE\_KEY\_2', 'hex')

web3.eth.getTransactionCount(account1, (err, txCount) => {

// Crea la transazione

const txObject = {

nonce: web3.utils.toHex(txCount),

to: account2,

valore: web3.utils.toHex(web3.utils.toWei('0.1', 'ether')),

gasLimit: web3.utils.toHex(21000),

gasPrice: web3.utils.toHex(web3.utils.toWei('10', 'gwei'))

}

// Firmare la transazione

const tx = new Tx(txObject)

tx.sign(privateKey1)

const serializedTx = tx.serialize()

const raw = '0x' + serializedTx.toString('hex')

// Trasmettere la transazione

web3.eth.sendSignedTransaction(raw, (err, txHash) => {

console.log('txHash:', txHash)

// Ora controllate etherscan per vedere la transazione!

})

})

* Deploy dei contratti a Web3

Esistono diversi modi per distribuire i contratti intelligenti sulla Blockchain di Ethereum. Ci sono anche diversi modi per distribuirli all'interno di Web3.js stesso. Come nella lezione precedente di questa serie, vi mostrerò un metodo che vi aiuterà a capire meglio cosa succede quando uno smart contract viene distribuito su Ethereum Blockchain. Questo esempio è stato progettato per suddividere il deployment in ogni fase del processo.

Questa lezione utilizzerà lo stesso file app.js che abbiamo usato nella lezione precedente. Lo imposteremo in questo modo:

Per seguire l'esercitazione, si consiglia di leggere questo codice:

var Tx = require('ethereumjs-tx')

const Web3 = require('web3')

const web3 = new Web3('https://ropsten.infura.io/YOUR\_INFURA\_API\_KEY')

const account1 = '' // L'indirizzo del vostro conto 1

const privateKey1 = Buffer.from('YOUR\_PRIVATE\_KEY\_1', 'hex')

Questo esempio di lezione consisterà negli stessi tre passi fondamentali della lezione precedente:

* Creare un oggetto transazione
* Firmare la transazione
* Inviare la transazione

Questi passaggi sono gli stessi perché ogni volta che scriviamo dati sulla blockchain, si tratta sempre degli stessi passaggi di base. Sto cercando di mostrarvi che l'implementazione di un contatto intelligente è in realtà molto simile all'invio di Ether da un conto a un altro, o alla chiamata di una funzione di uno smart contract. Stiamo ancora creando una transazione e la stiamo inviando alla rete. L'unica differenza sono i parametri della transazione.

Procediamo a costruire l'oggetto transazione in questo modo:

const txObject = {

nonce: web3.utils.toHex(txCount),

gasLimit: web3.utils.toHex(1000000), // Aumenta il limite di gas a un importo molto più alto

gasPrice: web3.utils.toHex(web3.utils.toWei('10', 'gwei')),

dati: dati

}

Stiamo costruendo questo oggetto transazione che ha molti degli stessi campi dell'oggetto della lezione precedente, come nonce, gasLimit e gasPrice. Ci sono anche alcune differenze fondamentali. Analizziamo ciascuno di essi:

* nonce - è il conteggio delle transazioni precedenti per il conto in questione. È lo stesso della lezione precedente.
* gasLimit - è la quantità massima di gas consumata dalla transazione. Aumenteremo questo limite perché la distribuzione di contratti intelligenti richiede molto più gas dell'invio di Ether.
* gasPrice - è l'importo che vogliamo pagare per ogni unità di gas. È lo stesso della lezione precedente.
* value - questo parametro è assente in questo esempio perché non stiamo inviando alcun Ether in questa transazione.
* to - questo parametro è assente perché non stiamo inviando questa transazione a un conto particolare. La stiamo invece inviando all'intera rete, perché stiamo distribuendo uno smart contract!
* data - questo sarà il bytecode dello smart contract che vogliamo distribuire. Assegneremo a questa variabile un valore, che spiegherò meglio tra poco.

Parliamo del parametro data. Si tratta della rappresentazione bytecode compilata dello smart contract in esadecimale. Per ottenere questo valore, abbiamo prima bisogno di uno smart contract e poi di compilarlo! Potete usare qualsiasi contratto smart che vi piace, soprattutto perché lo stiamo distribuendo su una rete di prova. Tuttavia, io utilizzerò uno smart contract con token ERC-20 che ho costruito in questo video. Potete seguirmi nel video tutorial Web3.js qui sopra per vedere come compilo questo particolare smart contract ERC-20 con Remix per ottenere questa stringa di dati. Una volta compilato il contratto, si può assegnare il valore dei dati a una variabile come questa:

const data = '0x60806040526040805190810160405280600a81526020017f44…

const txObject = {

nonce: web3.utils.toHex(txCount),

gasLimit: web3.utils.toHex(1000000), // Raise the gas limit to a much higher amount

gasPrice: web3.utils.toHex(web3.utils.toWei('10', 'gwei')),

data: data

}

const tx = new Tx(txObject)

tx.sign(privateKey1)

const serializedTx = tx.serialize()

const raw = '0x' + serializedTx.toString('hex')

web3.eth.sendSignedTransaction(raw, (err, txHash) => {

console.log('err:', err, 'txHash:', txHash)

// Use this txHash to find the contract on Etherscan!

})

})

* Chiamare funzioni degli smart contract

Questa lezione utilizzerà molti degli stessi passaggi di base delle lezioni precedenti perché, come le lezioni precedenti, è stata pensata per mostrare tutti i passaggi di base necessari per creare transazioni sulla Blockchain di Ethereum. Utilizzeremo la stessa configurazione di base con un file app.js che avrà questo aspetto:

const Web3 = require('web3')

const web3 = new Web3('https://ropsten.infura.io/YOUR\_INFURA\_API\_KEY')

const account1 = '' // L'indirizzo del vostro account 1

const account2 = '' // L'indirizzo del vostro conto 2

const privateKey1 = Buffer.from('YOUR\_PRIVATE\_KEY\_1', 'hex')

const privateKey2 = Buffer.from('YOUR\_PRIVATE\_KEY\_2', 'hex')

Creeremo anche un oggetto transazione, proprio come questo:

const txObject = {

nonce: web3.utils.toHex(txCount),

gasLimit: web3.utils.toHex(800000),

gasPrice: web3.utils.toHex(web3.utils.toWei('10', 'gwei')),

a: contractAddress,

dati: dati

}

Se avete seguito le lezioni precedenti, molti di questi valori dovrebbero esservi familiari. Prendiamo nota di alcuni cambiamenti.

* to - Questo parametro sarà l'indirizzo del contratto distribuito. Otterremo questo valore e lo assegneremo a breve.
* data - sarà la rappresentazione esadecimale della funzione che vogliamo chiamare sullo smart contract. Anche questo valore sarà assegnato momentaneamente.

Per compilare questi valori, dovremo ottenere l'ABI del contratto smart per questo token ERC-20. Potete seguirmi nel video qui sopra mentre ottengo l'ABI da Remix. Dovrò anche ottenere l'indirizzo dello smart contract da Etherscan (questo era disponibile quando abbiamo distribuito lo smart contract nell'ultima lezione). Ora che abbiamo entrambe le cose, possiamo creare una rappresentazione in JavaScript dello smart contract con Web3.js in questo modo:

const contractAddress = '0xd03696B53924972b9903eB17Ac5033928Be7D3Bc'

const contractABI = [{"constant":true, "inputs":[], "name": "name", "outputs":[{"name":", "type": "string"}], "payable":false, "stateMutability": "view","type":"function"},{"constant":false,"inputs":[{"name":"\_spender","type":"address"},{"name":"\_value","type":"uint256"}],"name": "approva", "uscite":[{"nome": "successo", "tipo": "bool"}], "pagabile":false, "statoMutabilità": "non pagabile", "tipo": "funzione"},{"costante":true, "ingressi": [], "nome": "totalSupply", "output":[{"nome":", "tipo": "uint256"}], "payable":false, "stateMutability": "view", "type": "function"},{"constant": false,"inputs":[{"name":"\_from","type":"address"},{"name":"\_to","type":"address"},{"name":"\_value","type":"uint256"}],"name":"transferFrom","outputs": [{"name": "success", "type": "bool"}], "payable":false, "stateMutability": "nonpayable", "type": "function"},{"constant":true, "input": [], "nome": "standard", "uscite":[{"nome":"", "tipo": "stringa"}], "pagabile":false, "mutabilità di stato": "vista", "tipo": "funzione"},{"costante": true, "input":[{"name":"", "type": "address"}], "name": "balanceOf", "output":[{"name":"", "type": "uint256"}], "payable":false, "stateMutability": "view", "type": "function"},{"constant":true, "input":[], "name": "symbol", "output":[{"name":"", "type": "string"}], "payable":false, "stateMutability": "view","type":"function"},{"constant":false,"inputs":[{"name":"\_to","type":"address"},{"name":"\_value","type":"uint256"}],"name": "trasferimento", "uscite":[{"nome": "successo", "tipo": "bool"}], "pagabile":false, "statoMutabilità": "non pagabile", "tipo": "funzione"},{"costante": true,"inputs":[{"name":"","type":"address"},{"name":"","type":"address"}],"name":"allowance","outputs":[{"name":"","type":"uint256"}],"payable": false, "stateMutability": "view", "type": "function"},{"input":[], "payable":false, "stateMutability": "nonpayable", "type": "constructor"},{"anonymous": false, "input":[{"indexed":true, "name":"\_from", "type": "address"},{"indexed":true, "name":"\_to", "type": "address"},{"indexed":false, "name": "\_valore", "tipo": "uint256"}], "nome": "Trasferimento", "tipo": "evento"},{"anonymous":false, "input":[{"indexed":true, "name":"\_owner", "type": "address"},{"indexed":true,"name":"\_spender","type":"address"},{"indexed":false,"name":"\_value","type":"uint256"}],"name":"Approval","type":"event"}]

const contract = new web3.eth.Contract(abi, contractAddress)

Ottimo! Ora abbiamo una rappresentazione JavaScript del contratto distribuito. Ora possiamo compilare il campo dati della transazione convertendo la funzione transfer() del contratto in bytecode (è la funzione che chiameremo su questo smart contract). Possiamo farlo con la funzione Web3.js encodeABI(), disponibile sull'oggetto contratto. L'aspetto è questo:

const data = contract.methods.transfer(account2, 1000).encodeABI()

Ecco fatto! Ecco quanto è facile codificare questa chiamata di funzione per la transazione! Si noti che stiamo trasferendo 1.000 gettoni al conto2. Questo metodo si occupa di codificare anche i parametri della funzione!

Questo è tutto ciò di cui abbiamo bisogno per costruire l'oggetto transazione. Come nelle lezioni precedenti, ora possiamo firmare la transazione e inviarla. Una volta fatto, possiamo registrare i valori dei saldi dei conti per vedere che la funzione dello smart contract è stata chiamata e che il trasferimento dei token è stato completato.

* Eventi degli smart contract con Web3.js

Gli smart contract di Ethereum hanno la capacità di emettere eventi che indicano che è successo qualcosa nell'esecuzione del codice dello smart contract. I consumatori hanno la possibilità di sottoscrivere questi eventi e Web3.js ci fornirà questa funzionalità. Questo è esattamente ciò che tratteremo in questa lezione.

Continueremo a usare uno smart contract ERC-20 come punto di riferimento per questa esercitazione, perché questo standard specifica che lo smart contract deve emettere un evento Transfer ogni volta che un token ERC-20 viene trasferito. Ci collegheremo alla rete principale di Ethereum per sottoscrivere l'evento Transfer per il token ERC-20 di OmiseGo

Procediamo con la configurazione del file app.js, come abbiamo fatto nelle lezioni precedenti. Questa volta, ci collegheremo alla rete principale di Ethereum. Incolliamo l'ABI e l'indirizzo dello smart contract OmiseGo, che possono essere ottenuti da Etherscan (guardate il video qui sopra per le istruzioni). Una volta ottenute queste due informazioni, possiamo creare una rappresentazione JavaScript dello smart contract con Web3.js e assegnarla a una variabile

* Ispezione dei blocchi con Web3

L'ispezione dei blocchi è spesso utile per analizzare la storia della Blockchain di Ethereum. Web3.js ha molte funzionalità che ci aiutano a fare proprio questo. Ad esempio, potremmo costruire qualcosa che assomigli a questa funzione di cronologia dei blocchi su Etherscan:

Prepariamo un file app.js per iniziare a utilizzare alcune delle funzionalità fornite da Web3.js. Questa configurazione sarà molto più semplice rispetto alle lezioni precedenti. Ci collegheremo alla rete principale per ispezionare i blocchi:

const Web3 = require('web3')

const web3 = new Web3('https://mainnet.infura.io/YOUR\_INFURA\_API\_KEY')

Per prima cosa, possiamo ottenere il numero dell'ultimo blocco in questo modo:

web3.eth.getBlockNumber().then(console.log)

Possiamo anche ottenere tutti i dati dell'ultimo blocco in questo modo:

web3.eth.getBlock('latest').then(console.log)

Se volessimo creare una funzione di cronologia dei blocchi come quella di Etherscan illustrata sopra, dovremmo ottenere un elenco dei blocchi più recenti nella catena. Possiamo farlo recuperando il blocco più recente e contando all'indietro fino agli ultimi 10 blocchi della catena. Possiamo farlo con un ciclo for come questo:

web3.eth.getBlockNumber().then((latest) => {

for (let i = 0; i < 10; i++) {

web3.eth.getBlock(latest - i).then(console.log)

}

})

Web3.js ha un'altra bella funzione che consente di ispezionare le transazioni contenute in un blocco specifico. Possiamo farlo in questo modo:

const hash = '0x66b3fd79a49dafe44507763e9b6739aa0810de2c15590ac22b5e2f0a3f502073'

web3.eth.getTransactionFromBlock(hash, 2).then(console.log)

* Web3 Utilities

Colleghiamoci alla rete principale di Ethereum in questo modo:

const Web3 = require('web3')

const web3 = new Web3('https://mainnet.infura.io/YOUR\_INFURA\_API\_KEY')

Per prima cosa, è possibile ottenere il prezzo medio del gas per la rete in questo modo:

web3.eth.getGasPrice().then((result) => {

console.log(web3.utils.fromWei(result, 'ether'))

})

Se avete già sviluppato sulla blockchain, probabilmente avete avuto a che fare con le funzioni di hashing. Web3.js ha molti aiuti integrati per l'uso delle funzioni di hashing. È possibile accedere direttamente alla funzione sha3 in questo modo:

console.log(web3.utils.sha3('Dapp University'))

Oppure come keccack256:

console.log(web3.utils.keccak256('Dapp University'))

Si può anche gestire la (pseudo) casualità generando un esagono casuale di 32 byte come questo:

console.log(web3.utils.randomHex(32))

Vi siete mai trovati a cercare di eseguire un'azione su un array o un oggetto JavaScript e avete avuto bisogno dell'aiuto di una libreria esterna? Fortunatamente, Web3.js viene fornito con la libreria underscoreJS:

const \_ = web3.utils.\_

\_.each({ key1: 'value1', key2: 'value2' }, (value, key) => {

console.log(key)

})

E questo è tutto! Questi sono alcuni suggerimenti e trucchi di fantasia che potete utilizzare con Web3.js. Ecco il codice completo di questa lezione:

const Web3 = require('web3')

const web3 = new Web3('https://mainnet.infura.io/YOUR\_INFURA\_API\_KEY')

// Ottenere il prezzo medio del gas in wei dagli ultimi blocchi di prezzo mediano del gas

web3.eth.getGasPrice().then((result) => {

console.log(web3.utils.fromWei(result, 'ether'))

})

// Utilizzare la funzione di hashing sha256

console.log(web3.utils.sha3('Dapp University'))

// Utilizzare la funzione di hashing keccak256 (alias)

console.log(web3.utils.keccak256('Dapp University'))

// Ottenere un esadecimale casuale

console.log(web3.utils.randomHex(32))

// Ottenere l'accesso alla libreria JS underscore

const \_ = web3.utils.\_

\_.each({ key1: 'value1', key2: 'value2' }, (value, key) => {

console.log(key)

})

## Vulnerabilità di Solidity e degli smart contract

Ci sono molte vulnerabilità che emergono nella creazione degli smart contract; l’intento di questa sezione è di considerarne quante più possibili e immaginare le possibili soluzioni, corredando il tutto con esempi pratici.

Prendiamo ad esempio un codice semplice, al quale aggiungeremo una funzione di impostazione del prezzo:

Immagine che contiene testo

Descrizione generata automaticamenteImmagine che contiene testo

Descrizione generata automaticamente Riferimenti di questi codici ed immagini: <https://www.gofyeo.com/post/vulnerabilities-in-solidity-smart-contracts>

Ora possiamo aggiornare il prezzo in qualsiasi momento, ma questa funzione ha introdotto alcuni problemi:

* Manca la convalida del mittente: chiunque può modificare il prezzo.
* Manca la convalida degli input: se il prezzo è impostato a 0, i token possono essere acquistati gratuitamente.

Per convalidare il mittente, dobbiamo memorizzare l'indirizzo che è autorizzato a eseguire operazioni di amministrazione.

Immagine che contiene testo

Descrizione generata automaticamente

Ora il proprietario sarà impostato sull'indirizzo che ha distribuito il contratto e solo questo indirizzo sarà in grado di modificare il prezzo. Inoltre, abbiamo aggiunto il controllo che il prezzo sia maggiore di 0 sia nel costruttore che nel setter. Si consiglia di eseguire un controllo di correttezza dei valori importanti anche nel costruttore.

La logica successiva che possiamo aggiungere a questo contratto è quella delle commissioni: ogni volta che l'utente acquista un token, il 5% delle entrate deve andare a un indirizzo dedicato.

Immagine che contiene testo

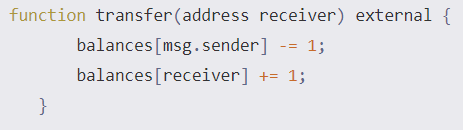
Descrizione generata automaticamente

La matematica potrebbe sembrare corretta: calcoliamo quanto è l'1% e poi lo moltiplichiamo per ottenere il 5%. Purtroppo, la divisione di numeri interi restituisce solo un quoziente e un promemoria andrebbe perso.

Se nel nostro esempio l'importo è 40, allora (40/100)\*5 restituirà 0, anche se il risultato corretto è 2. *Si dovrebbe sempre eseguire la moltiplicazione prima della divisione.*

Inoltre, abbiamo aggiunto una funzione per impostare l'indirizzo di feeCollector, ma il costruttore rimane invariato. *Le* *variabili non inizializzate sono impostate per default a 0*. Ciò significa che per default feeCollector sarebbe address(0) e, a causa della mancata validazione, possiamo impostare manualmente questo valore. *I PF inviati all'indirizzo(0) andrebbero persi.*

Ora gli utenti possono acquistare i token ma non possono utilizzarli. Diamo loro la possibilità di trasferire un token a qualcuno.



La funzione è molto semplice: basta diminuire il saldo del mittente e aggiungere il token all'indirizzo del destinatario. Si può notare che non ci sono controlli per verificare se il mittente ha token nel suo saldo. In questo esempio la versione del compilatore è impostata a >= 0.8.0, dove solidity controlla automaticamente i casi di underflow/overflow. Se si tenta di sottrarre da uno zero, viene generato un errore. *Prima di solidity 0.8.0 era necessario controllare manualmente la presenza di underflow/overflow o utilizzare una libreria speciale come SafeMath.*

*Si ricordi inoltre che la visibilità di default delle funzioni è pubblica e queste dovrebbero essere valutate quando devono essere accessibili all’esterno.*

Aggiungiamo una funzione che trasferisce token a due destinatari:

Immagine che contiene testo

Descrizione generata automaticamente

Il contratto sembra essere completato, ma manca ancora una parte importante. Quando il token viene acquistato, il 95% del prezzo rimarrà nel saldo del contratto, ma non c'è modo di ritirarlo. È importante *assicurarsi che il contratto abbia una funzionalità di prelievo di ETH se ha funzioni di pagamento.*

Immagine che contiene testo

Descrizione generata automaticamente

Come per altri linguaggi, occorre specificare i contratti nell’ordine giusto quando si lavora con l’ereditarietà multipla, sapendo che verrà considerato l’ultimo contratto ereditato.

Quando invece si hanno più variabili con lo stesso nome, si ha il *variable shadowing* come nel codice che segue:

Immagine che contiene testo

Descrizione generata automaticamente

Quando si verifica lo shadowing delle variabili, *entrambe le variabili vengono create in memoria e sono accessibili dal contratto corrispondente*. Nel contratto di esempio, Derived::getValueBase() restituirà 1.

Alcuni problemi comuni qui esaminati:

* Controlla se un valore deve essere modificato dopo la distribuzione;
* Controllare la convalida del mittente;
* Controllare la convalida degli input;
* Eseguire la moltiplicazione prima della divisione;
* Ricordare che le variabili non inizializzate sono 0 per impostazione predefinita;
* Controllare che l'ETH non venga inviato all'indirizzo(0);
* Controllare che vengano gestiti underflow/overflow;
* Verificare che ogni funzione specifichi il livello di visibilità;
* Verificare che le funzioni pubbliche siano accessibili dall'esterno;
* Assicurarsi che il contratto abbia una funzionalità di ritiro dell'ETH se ha funzioni a pagamento;
* Controllare l'ordine di ereditarietà;
* Verificare lo shadowing delle variabili;
* Verificare come vengono utilizzate le funzioni sovrascritte nel contratto di base.

Ulteriori vulnerabilità che si possono considerare:

* Salvare dati riservati non criptati sulla blockchain
  + La blockchain è accessibile a chiunque, il che significa che non c'è assolutamente nulla di riservato; se per errore vi si salvano password (o informazioni simili), ci si trova in grossi guai.
* Soluzione: Non memorizzare mai informazioni riservate su una blockchain, a meno che non vengano crittografate o sottoposte a hash.
* Overflow e underflow di interi (risolto da solidity 0.8)
  + Questa è probabilmente una delle vulnerabilità più facili da capire e da dimenticare allo stesso tempo. Solidity offre diversi tipi di interi in base al valore massimo disponibile: int8 (8 bit, valore massimo 2⁸-1), int16 (16 bit, valore massimo 2¹⁶-1), ...
  + Se si esegue un calcolo che restituisce un risultato più grande del valore massimo o più piccolo di 0, Solidity non lancerà un'eccezione, ma salterà all'altro capo dell'intervallo, ad esempio, per un uint8, 2⁸-1 + 4 = 3...
* Soluzione: Fortunatamente questa vulnerabilità è stata risolta a partire da solidity 0.8. Se si utilizza ancora una versione inferiore di solidity, è possibile utilizzare la libreria SafeMath di OpenZeppelin o verificare da soli la presenza di integer overflow/underflow.
* Valori di ritorno delle chiamate non controllati
  + A un certo punto del vostro codice, potreste invocare una funzione da uno smart contract di terze parti. L'esecuzione potrebbe fallire o restituire un risultato inaspettato.
* Soluzione: Occorre sempre controllare ogni valore ritornato, specie quando si ha a che fare con valori nulli. Dipende sempre dal tipo di funzione e da cosa ritorna.
* Re-entrancy attacks (attacchi di rientranza)
  + Questa è probabilmente la vulnerabilità più nota dell'elenco. Ogni volta che si invoca una funzione di un contratto esterno, bisogna sempre tenere presente che questo contratto potrebbe invocare a sua volta la funzione del proprio contratto (creando una sorta di ciclo di invocazione).
  + Questo può essere potenzialmente pericoloso se, ad esempio, si inviano fondi a un conto esterno (che si dà il caso sia un contratto esterno), ma non si è aggiornato il suo saldo prima di inviarli. Ogni volta che il contratto esterno "rientrerà" nella funzione, riceverà la stessa quantità di fondi. Al termine dell'esecuzione, l'importo inviato sarà probabilmente più volte superiore a quello previsto. Vediamo un esempio di seguito.

Prendiamo questa funzione:

Immagine che contiene testo

Descrizione generata automaticamente

Riferimento di questo codice e dei pezzi successivi: <https://medium.com/coinmonks/protect-your-solidity-smart-contracts-from-reentrancy-attacks-9972c3af7c21>

Tutto ciò di cui un aggressore ha bisogno per sfruttare questa funzione è ottenere una certa quantità di saldo mappata all'indirizzo del proprio smart contract e creare una funzione di fallback che chiami *withdraw*.

Dopo che *msg.sender.call.value(amount)()* trasferisce l'importo corretto, la funzione di fallback dell'attaccante chiama nuovamente withdraw, trasferendo altri fondi prima che *balances[msg.sender] = 0* possa fermare ulteriori trasferimenti. Questa procedura continua fino a quando non rimane più etere o l'esecuzione raggiunge la dimensione massima dello stack.

In genere una funzione vulnerabile effettua una chiamata esterna utilizzando *transfer*, *send* o *call*. Le differenze tra queste funzioni saranno trattate nella sezione dedicata alla prevenzione degli attacchi di rientranza.

Esistono diversi tipi di attacchi di rientranza:

* Attacco di rientranza a funzione singola (Single function reentrancy attack)
  + Questo tipo di attacco è il più semplice e facile da prevenire. Si verifica quando la funzione vulnerabile è la stessa funzione che l'attaccante sta cercando di chiamare ricorsivamente. L'esempio di codice precedente è un attacco di rientranza a funzione singola.
* Attacco di rientranza a funzioni incrociate (Cross-function reentrancy attack)
  + Questi attacchi sono più difficili da rilevare. Un attacco di tipo cross-function reentrancy è possibile quando una funzione vulnerabile condivide lo stato con un'altra funzione che ha un effetto desiderato dall'attaccante. Vediamo il seguente esempio.

Immagine che contiene testo

Descrizione generata automaticamente

In questo esempio, *withdraw* chiama la funzione di fallback dell'attaccante, come nel caso dell'attacco di rientranza a funzione singola.

La differenza è che la funzione di fallback effettua una chiamata a transfer invece di chiamare ricorsivamente *withdraw*. Poiché il saldo non è stato impostato a 0 prima di questa chiamata, la funzione *transfer* può trasferire un saldo che è già stato speso.

* Soluzioni possibili 🡪 Ci sono alcune best practices da seguire per proteggere gli smart contracts
  + *Send, Transfer* e *Call*
    - Poiché la maggior parte degli attacchi di rientranza coinvolge funzioni di *send*, *transfer* o *call*, è importante capire la differenza tra di esse.
    - Le funzioni *send* e *transfer* sono considerate più sicure perché sono limitate a 2.300 gas. Il limite di gas impedisce le costose chiamate di funzioni esterne al contratto di destinazione. L'unica insidia è quando un contratto imposta una quantità di gas personalizzata per un invio o un trasferimento utilizzando *msg.sender.call(ethAmount).gas(gasAmount*).
    - La funzione *call* è purtroppo molto più vulnerabile.
    - Quando si prevede che una chiamata di funzione esterna esegua operazioni complesse, in genere si vuole usare la funzione *call* perché inoltra tutto il gas rimanente. Questo apre la porta a un attaccante che può richiamare la funzione originale in un attacco di rientranza a funzione singola o una funzione diversa dal contratto originale in un attacco di rientranza a funzione incrociata.
    - Ove possibile, utilizzare *send* o *transfer* al posto di call per limitare il rischio di sicurezza.
  + Contrassegnare le funzioni non attendibili
    - Per proteggersi dagli attacchi di reentrancy, è importante identificare quando una funzione non è attendibile. Le best practices di Consensys (azienda leader nella formazione blockchain) consigliano di denominare le funzioni e le variabili in modo da indicare se non sono attendibili. Ad esempio:

Immagine che contiene testo

Descrizione generata automaticamente

* + - È importante ricordare che se una funzione chiama un'altra funzione non attendibile, è anch'essa non attendibile.
  + Schema controlli-effetti-interazioni (Checks-effects-interactions pattern)
    - Il metodo più affidabile per proteggersi dagli attacchi di rientranza è l'uso dello schema checks-effects-interactions. Questo schema definisce l'ordine in cui strutturare le funzioni.
    - Prima si eseguono i controlli, che di solito sono dichiarazioni assert e require, all'inizio della funzione. Se i controlli passano, la funzione deve poi risolvere tutti gli effetti allo stato del contratto.
    - Solo dopo che tutti i cambiamenti di stato sono stati risolti, la funzione deve interagire con altri contratti. Chiamando le funzioni esterne per ultime, anche se un attaccante effettua una chiamata ricorsiva alla funzione originale, non può abusare dello stato del contratto.
    - Riscriviamo la nostra funzione di prelievo vulnerabile utilizzando lo schema controlli-effetti-interazioni.

Immagine che contiene testo

Descrizione generata automaticamente

* + - Poiché azzeriamo il saldo - un effetto - prima di effettuare una chiamata esterna, una chiamata ricorsiva effettuata da un attaccante non sarà indotta a pensare che ci sia ancora un saldo residuo.
  + Mutex
    - In situazioni più complesse, come la protezione dagli attacchi di rientranza tra funzioni, può essere necessario utilizzare un mutex. Un mutex pone un blocco sullo stato del contratto. Solo il proprietario del blocco può modificare lo stato.

Immagine che contiene testo, lettera

Descrizione generata automaticamente

* + - Utilizzando questo blocco, un aggressore non può più sfruttare la funzione withdraw con una chiamata ricorsiva. Né un aggressore può sfruttare una chiamata al trasferimento per un attacco di rientranza tra funzioni. Tutte le modifiche allo stato avvengono quando il blocco è vero, impedendo a qualsiasi funzione che controlla il blocco di essere chiamata fuori ordine.
    - È necessario prestare attenzione all'implementazione di un mutex per assicurarsi che ci sia sempre un modo per rilasciare un blocco. Se un attaccante riesce a ottenere un blocco sul contratto e a impedirne il rilascio, il contratto può essere reso inerte.

* DoS (Denial of Service)
  + Supponiamo che a un certo punto della logica del codice si invochi una funzione di un contratto esterno e che, se l'invocazione ha successo, si prosegua con l'esecuzione del codice. Se per qualche motivo uno sviluppatore malintenzionato ha implementato la funzione del contratto esterno in modo che fallisca sempre, la parte finale della funzione non verrà mai eseguita, effettuando un attacco DoS in piena regola
* Soluzione: Presupporre sempre che le chiamate esterne possano fallire. Se possibile, implementare uno schema "pull" invece di uno "push".
  + In uno schema "push", i contratti invocano la funzione del proprio codice che invocherà un contratto di terze parti. In questo modo un contratto maligno potrebbe bloccare altri utenti.
  + In uno schema "pull", i contratti invocano la funzione del proprio codice, che in cambio invocherà solo la funzione del contratto chiamante. In questo modo, se uno sviluppatore malintenzionato tenta di fare un DoS alla vostra funzione, bloccherà solo sé stesso e non gli altri utenti.
* Replay signature attacks (Attacchi con firme di replay)
  + Le firme sono un modo per un conto di inviare le transazioni di un altro conto alla blockchain. Il conto originale firmerà un messaggio e il conto di consegna invierà il messaggio a uno smart contract, in modo che sia il conto di consegna a pagare le spese di transazione e non il conto originale.
  + Naturalmente lo smart contract deve avere funzioni in grado di convalidare le firme e di eseguire i compiti richiesti.
  + Tuttavia, se un messaggio firmato è valido e il vostro smart contract ne eseguirà il contenuto, chiunque avrà accesso ad esso (letteralmente chiunque monitori la blockchain) potrebbe inviarlo più volte, il che significa che la funzionalità dello smart contract potrebbe essere eseguita più volte per lo stesso messaggio, cosa che non era nelle intenzioni dell'account originale quando ha firmato il messaggio.
* Soluzione: Aggiungere un nonce per account ai messaggi firmati. I messaggi firmati dallo stesso account con un nonce già utilizzato non vengono accettati. Ogni volta che un account firma un nuovo messaggio, dovrà incrementare il nonce. Lo standard EIP-712 può aiutare in questo senso.
* Definire sempre la visibilità di default delle funzioni, specie per tenere private le funzioni che devono esserlo
* Definire sempre una specifica versione di pragma di Solidity
* Cercare sempre di usare i mapping quando si prevede un lungo elenco di valori
* Ereditarietà sbagliata
  + Solidity supporta eredità multiple che introducono l'ambiguità chiamata "problema del diamante". In pratica, se più contratti padre implementano la stessa funzione, occorre capire quale contratto sarà chiamato
  + Solidity utilizza la linearizzazione C3 per stabilire le priorità tra i contratti padre. Se non si presta attenzione a questo aspetto, si potrebbero avere comportamenti inaspettati. Esso garantisce che si consideri l’ordine di definizione delle classi e la relazione di ereditarietà tra di esse. In pratica, l'algoritmo crea una lista ordinata di tutte le classi coinvolte nell'ereditarietà, in modo che ogni classe appaia solo una volta nella lista e rispetti l'ordine di definizione delle classi.
* Soluzione: Come regola generale, si aggiungono i contratti padre partendo da quello più generico a quello più specifico.
* Saldo inatteso di ether
  + Si potrebbe pensare che l'unico modo per il proprio contratto di ricevere etere sia quello di implementare funzioni "payable" e/o una funzione di receive. Ma anche se il contratto non ha assolutamente funzioni " payable " e non ha funzioni di receive, potrebbe comunque riceverne nelle due situazioni seguenti:
    - Come risultato di una "autodistruzione" di un altro contratto. Ogni volta che un contratto si autodistrugge può decidere a quale conto inviare il suo saldo corrente.
    - Come risultato di un nuovo ether estratto. Ogni volta che un miner aggiunge un blocco alla blockchain riceverà una ricompensa in ether, il miner può decidere a quale conto assegnare l'ether.
    - Questi due tipi di trasferimento di ether non saranno comunque catturati dalla funzione receive
* Soluzione: Non pensare mai di poter memorizzare in una variabile di stato la quantità esatta di ether che il vostro contratto possiede semplicemente reagendo alle transazioni ricevute. Nel migliore dei casi, si sarà in grado di memorizzare la quantità minima che possiede.
* Accesso al di fuori dei limiti dell'array
  + Nelle prime versioni di Solidity, le lunghezze degli array dinamici di memorizzazione potevano essere modificate direttamente (senza aggiungere o rimuovere elementi). Il problema è che le variabili di stato condividono lo stesso spazio di memorizzazione, il che significa che se si lascia che gli utenti modifichino la lunghezza di un array, possono potenzialmente accedere a qualsiasi slot nello spazio di memorizzazione, e potrebbero sostituire qualsiasi variabile di stato nel contratto, anche quelle a cui non dovrebbero avere accesso.
* Soluzione: Utilizzare una versione di Solidity priva di questa vulnerabilità o semplicemente non consentire agli utenti di modificare le lunghezze degli array di storage.
* Delegare le chiamate a fonti non attendibili
  + Questa è probabilmente la più grande vulnerabilità di sicurezza che si possa avere. Le "chiamate delegate" invocano una funzione di un contratto esterno e la eseguono all'interno del contesto del contratto (a differenza delle normali chiamate che vengono eseguite nel contesto del chiamante). Il contratto esterno avrà pieno accesso alle variabili di stato e potrà fare ciò che vuole.
  + Soluzione: Usare con attenzione le chiamate ai delegati.
* Chiamate (regolari) a fonti non attendibili
  + Anche le chiamate a contratti non attendibili possono avere risultati inaspettati. Si tenga presente che richiamare due volte la funzione dello stesso contratto può restituire due risultati diversi, per esempio, si chiama una funzione di un contratto non attendibile "check\_condition\_X()" che inizialmente può restituire true, ma se la si richiama subito dopo (nulla nel contratto è cambiato nel frattempo e siamo ancora nella stessa transazione) potrebbe restituire false (lo stato del contratto non attendibile potrebbe essere cambiato a causa della prima invocazione).
* Soluzione: Se nel codice è necessario un valore fornito dall'esterno, invece di richiamare ogni volta il contratto esterno, salvare il valore in una variabile locale.
* Casualità insicura
  + Per definizione, i contratti intelligenti devono essere deterministici, devono restituire lo stesso risultato ogni volta che li eseguiamo, il che significa che, sempre per definizione, la casualità non è possibile.
* Soluzione: La soluzione più semplice è di non includere la casualità nel contratto smart, ma se si deve farlo, dipende da quanto deve essere sensibile il valore casuale. Potenzialmente si potrebbero usare variabili globali (timestamp del blocco, numero del blocco, ...) o usare un oracolo come Chainlink per ottenere una casualità sicura.
* Manipolazione del timestamp del blocco
  + I minatori scelgono il timestamp del blocco che estraggono, non possono impostare qualsiasi timestamp, ma hanno una certa flessibilità. Se state implementando uno smart contract che richiede valori di data e/o ora per funzionare correttamente, i minatori potrebbero avere la possibilità di violarlo.
* Soluzione: È possibile lavorare con i dati e l'ora a patto di non essere troppo restrittivi. Come regola generale, si dovrebbe essere in grado di tollerare una variazione di almeno 15 secondi.
* Contratti con codice zero
  + È possibile verificare se un conto corrisponde a uno smart contract o se è di proprietà esterna. Il modo per farlo è controllare la "dimensione del bytecode" relativa all'account (utilizzando l'assembly). Gli account hanno diverse proprietà in Ethereum (saldo, nonce, ecc.), e una di queste è il "bytecode", che è vuoto per un account di proprietà esterna e contiene alcuni dati per gli smart contract.
  + Tuttavia, durante la distribuzione di un nuovo contratto, la funzione "costruttore" del contratto verrà eseguita prima della creazione dello smart contract, il che significa che durante questo processo di esecuzione, l'indirizzo dello smart contract non avrà alcun "bytecode" collegato. Se uno smart contract invoca la vostra funzione mentre è stato distribuito, non verrà rilevato come smart contract ma come account di proprietà esterna.
* Soluzione: Se avete bisogno di limitare l'accesso al vostro smart contract, il modo migliore sarebbe quello di avere una whitelist, ma se davvero si ha bisogno di impedire agli smart contract di invocare il codice del proprio contratto, forse il modo migliore sarebbe quello di controllare se "msg.sender" corrisponde a "tx.origin", dato che l’account originale di una transazione deve sempre essere un conto di proprietà esterna.
* Puntatori di memoria non inizializzati
  + Questa è per lo più una buona pratica. Inizializzare sempre la variabile di stato (dal costruttore o dalla funzione di inizializzazione, se si utilizzano proxy trasparenti), altrimenti i valori vuoti predefiniti potrebbero portare a comportamenti inaspettati.
* Soluzione: Fortunatamente dalla versione 0.5.0 di Solidity i puntatori di stato devono essere inizializzati.
* Contratti intelligenti non aggiornabili
  + I contratti intelligenti sono per definizione immutabili (possono essere "autodistrutti" non modificati) e inarrestabili, il che significa che se per qualche motivo si distribuisce un contratto intelligente difettoso, non sarà possibile correggerlo e distribuire nuovamente una nuova versione...
* Soluzione: Utilizzate sempre modelli aggiornabili per i propri smart contract (usando il design pattern Proxy, con implementazioni quali proxy trasparenti, admin proxy, beacon proxy, ecc.). Si può anche considerare la possibilità di aggiungere lo schema "Pausable" ad alcune funzioni del contratto. Le funzioni "Pausable" possono essere messe in pausa (il che significa che nessuno può invocarle), il che darà più tempo per risolvere un bug e distribuire una nuova versione.
* Implementazioni logiche inizializzabili
  + Per avere contratti intelligenti aggiornabili, come detto il pattern "proxy" viene utilizzato molto spesso. L'idea è semplice, in poche parole: viene distribuito un contratto "logico" (con tutto il codice byte dell'EVM), quindi vengono distribuiti uno o più contratti "proxy", configurati per delegare tutte le loro chiamate al contratto logico. I proxy non contengono alcuna logica, tutto il codice è nel contratto logico, tuttavia i proxy mantengono la propria variabile di stato.
  + In molti di questi casi è necessario "inizializzare" i proxy. L'inizializzazione di un contratto viene normalmente effettuata tramite la funzione "costruttore", che viene eseguita una sola volta (durante la distribuzione), ma se viene implementato il "modello proxy", i costruttori non possono essere utilizzati, poiché inizializzerebbero la logica e non i proxy. Per questo motivo, per inizializzare un proxy, i contratti logici espongono un metodo *initialize* che può essere chiamato da chiunque e che imposta lo stato iniziale del contratto. I proxy chiamano questo metodo (utilizzando una chiamata delegata come per qualsiasi altro metodo logico) e inizializzano il proprio stato.
  + La vulnerabilità deriva dal fatto che il metodo initialize può essere chiamato da chiunque, il che significa che chiunque potrebbe potenzialmente inizializzare il contratto logico stesso. All'inizio questo non è un problema, poiché i proxy si affidano al contratto logico per il codice byte eseguibile, non per lo stato, poiché mantengono il proprio, e l'inizializzazione dello stato logico non avrebbe un impatto su nessuno dei proxy.
  + Tuttavia, in alcuni casi, potrebbe accadere che un aggressore possa impostare lo stato del contratto logico in modo da avere la possibilità di distruggerlo, ad esempio se il contratto logico effettua una chiamata di delega a un indirizzo incluso nel suo stato che può essere impostato solo da un amministratore.
  + L'account dell'amministratore sarà probabilmente impostato attraverso il metodo initialize; un aggressore potrebbe quindi rivendicare l'amministrazione del contratto logico e impostare l'indirizzo della chiamata delegata a un contratto che esegua semplicemente il comando "autodistruzione". In questo caso, il contratto logico verrebbe rimosso e i proxy resterebbero "puntati" a un contratto vuoto.
* Soluzione: La soluzione è in realtà piuttosto semplice. Quando si distribuisce il contratto logico, si dovrebbe includere un metodo costruttore, il cui scopo sarebbe semplicemente quello di impostare il flag di inizializzazione su false. L'idea è che il metodo initialize non sia eseguibile nel contesto del contratto logico stesso, ma solo attraverso chiamate di delegati.
* Attacco Flashloan
  + I Flashloans consentono agli utenti di prendere in prestito enormi quantità di criptovalute con una sola e rapida transazione. In altre parole, con un flashloan un utente può prendere in prestito una grande quantità di criptovaluta (come Ethereum o Bitcoin) per un breve periodo di tempo (generalmente poche ore), e poi restituire il prestito insieme agli interessi. Ciò può essere fatto senza la necessità di fornire alcuna garanzia, poiché il prestito viene concesso solo per un breve periodo di tempo e viene restituito in una sola transazione.
  + Grazie alla loro adattabilità e accessibilità, non richiedono garanzie e possono essere utilizzati per una moltitudine di scopi. Tuttavia, poiché non richiedono alcuna garanzia, i flashloan possono essere utilizzati per manipolare il mercato o sfruttare le debolezze degli smart contract.
  + Esempio di exploit Beanstalk DeFi (17 aprile 2022). Beanstalk è un protocollo decentralizzato di stablecoin che premia gli utenti che contribuiscono con fondi a un pool di finanziamento centrale chiamato "silo". Il protocollo mantiene il token "bean" a circa 1 dollaro. Inoltre, Beanstalk presenta una struttura di governance che consente ai titolari di token di votare per modificare il codice, con un potere di voto proporzionale al numero di token posseduti. Questa struttura di governance ha portato a una vulnerabilità del sistema che ha causato la caduta del progetto.
  + L’attaccante ha creato due proposte. Una era la richiesta di inviare tutti i fondi al proprio indirizzo. L'altra consisteva nel donare 250000 dollari in gettoni BEAN all'Ucraina. E qui arriva la parte difficile: la funzione emergencyCommit consente agli utenti con la super maggioranza dei fondi di approvare una proposta.
  + Il protocollo ha dimenticato di prendere le precauzioni per i flashloan consentendo l'hack appena un giorno dopo la proposta malevola. L'exploiter ha ottenuto un massiccio flashloan: 350 milioni di DAI, 500 milioni di USDC, 150 milioni di USDT, 32 milioni di Bean e 11,6 milioni di LUSD. Li ha convertiti in token BEAN3Crv-f e BEANLUSD-f della governance di Beanstalk DeFi. Raggiunta la maggioranza del 78% dei fondi, hanno chiamato emergencyCommit per eseguire entrambe le proposte che avevano creato.
* Soluzioni
  + Richiedere garanzie per i prestiti per rendere più costoso per gli hacker prendere in prestito le risorse.
  + Implementare limiti sulle dimensioni dei prestiti flash che possono essere sottoscritti, il che può aiutare a mitigare gli attacchi su larga scala.
  + Aggiungere blocchi temporali ai contratti intelligenti per impedire la possibilità di rimborsare i prestiti flash troppo velocemente, dando agli sviluppatori il tempo di rilevare e prevenire gli attacchi.
  + Inoltre, i progetti con sistemi di votazione della governance devono implementare un meccanismo anti-flashloan per impedire agli sfruttatori di utilizzare i flashloan per ottenere la maggioranza del potere di voto.
* Limite di dimensione dello stack
  + Fino a un hard fork, era possibile creare una lunga catena di chiamate (una in meno del limite) e poi chiamare un contratto vittima. Se la vittima chiamava qualche altra funzione, questa falliva con un'eccezione dovuta al limite di dimensione dello stack. Il contratto vittima potrebbe non aspettarsi un'eccezione.
* Front running
  + È interessante notare che i contratti intelligenti e le transazioni diventano completamente pubblici una volta inviati alla rete come transazione in sospeso. Queste transazioni sono visibili all'intera rete nelle mempool dei nodi Ethereum, consentendo ai minatori di blocchi di selezionare le transazioni con le tariffe di gas più elevate. Questa visibilità ha un effetto collaterale significativo. Permette ad attori malintenzionati di vedere il risultato previsto di uno smart contract prima che venga confermato sulla blockchain.
  + Il front running è una pratica dannosa che si verifica quando un utente sfrutta la conoscenza anticipata di una transazione in arrivo per eseguire una transazione correlata con lo stesso contratto, in modo da ottenere un vantaggio sui concorrenti. In altre parole, l'utente malintenzionato utilizza informazioni riservate sulle transazioni in arrivo per eseguire una transazione in anticipo e trarre un profitto.
* Soluzioni
  + Esistono diverse soluzioni per affrontare il problema, quali:
    - Zero-knowledge Proofs: Sono un meccanismo crittografico che consente a un utente di dimostrare di avere accesso a una determinata informazione senza rivelare la stessa informazione agli altri utenti. In questo modo, gli utenti possono dimostrare di avere accesso a una transazione senza rivelare la stessa transazione agli altri utenti.
    - Auction Mechanisms: l'utilizzo di meccanismi di asta può aiutare a mitigare il front running, in quanto gli utenti sono tenuti a fare offerte senza conoscere le offerte degli altri utenti. In questo modo, gli utenti non possono conoscere le offerte degli altri utenti e non possono quindi eseguire transazioni in anticipo.
    - Randomness: l'introduzione di elementi di casualità nelle transazioni può rendere più difficile prevedere quando avverranno le transazioni e ridurre il rischio di front running.
    - Off-Chain Matching: l'utilizzo di sistemi di matching off-chain può ridurre il rischio di front running, in quanto le transazioni sono eseguite al di fuori della blockchain, in modo che gli utenti non possano vedere le transazioni degli altri utenti.

# Riferimenti usati

* Corso Blockchain Theory 101 Udemy: <https://www.udemy.com/course/blockchain-theory-101>
* Investopedia: <https://www.investopedia.com/>
* BitPanda: <https://www.bitpanda.com/>
* Intellipaat: <https://intellipaat.com/>
* VSkills: <https://www.vskills.in/certification/blog/>
* Appinventiv: <https://appinventiv.com/blog/>
* HolaCripto: <https://www.hola-cripto.com/>
* Simplilearn: <https://www.simplilearn.com/>
* Blocktrade: <https://blocktrade.com/>
* CyberScrilla: <https://cyberscrilla.com>
* Academy Binance: <https://academy.binance.com/it/articles/>
* Blockchain4Innovation: <https://www.blockchain4innovation.it/>
* QuickNode: <https://www.quicknode.com/>
* Cointelegraph: <https://cointelegraph.com/>
* Affidaty: <https://affidaty.io/blog/it/>
* Bitstamp: <https://www.bitstamp.net/learn/crypto-101/>
* Gemini: <https://www.gemini.com/it-IT/cryptopedia/>
* 101Blockchain: <https://101blockchains.com/>
* Naukri: <https://www.naukri.com/learning/articles/>
* Yahoo Finance: <https://finance.yahoo.com/news/>
* Komodo Academy: <https://komodoplatform.com/en/academy/>
* Hacken.io: <https://hacken.io/insights/>
* Horizen.io: <https://www.horizen.io/academy/>
* AnalyticSteps: <https://www.analyticssteps.com/blogs/>
* Foley: <https://www.foley.com/en/insights/publications>
* Data-Flair: <https://data-flair.training/blogs/>
* Limechain: <https://limechain.tech/blog/>
* WeSecureApp: <https://wesecureapp.com/blog/>
* Vitalik.ca: <https://vitalik.ca/>
* Ethereum Improvement Proposals: <https://eips.ethereum.org/EIPS/>
* OriginStamp: <https://originstamp.com/blog/>
* GeeksforGeeks: <https://www.geeksforgeeks.org/>
* One37PM: <https://www.one37pm.com/nft/>
* LeewayHertz: <https://www.leewayhertz.com/>
* Hedera: <https://hedera.com/learning/>
* The Motley Fool: <https://www.fool.com/investing/stock-market/market-sectors/financials/cryptocurrency-stocks/>
* Blog Medium
  + <https://jimmysong.medium.com/>
  + <https://medium.com/coinmonks/>
  + <https://marcomanoppo.medium.com/>
  + <https://ancapalex.medium.com/>
  + <https://unibrightio.medium.com/>
  + <https://medium.com/digitalbitsorg/>
  + <https://medium.com/coinmonks/>
* Canali YouTube:
  + <https://www.youtube.com/@DappUniversity>
* Tesi di riferimento:
  + <https://amslaurea.unibo.it/7934/1/bertani_beatrice_tesi.pdf>
* Documenti di riferimento:
  + <https://www.eublockchainforum.eu/sites/default/files/reports/SmartContractsReport_Final.pdf>
  + <https://www.dmi.unict.it/tramonta/sdm/L16_Ethereum.pdf>
  + <https://associazioneblockchain.it/wp-content/uploads/2020/03/20.04.15-I-token.-Una-prospettiva-giuridica.pdf>
* Libri di riferimento:
  + <https://hacken-3.gitbook.io/l1-security/>
  + <https://tokens-economy.gitbook.io/>