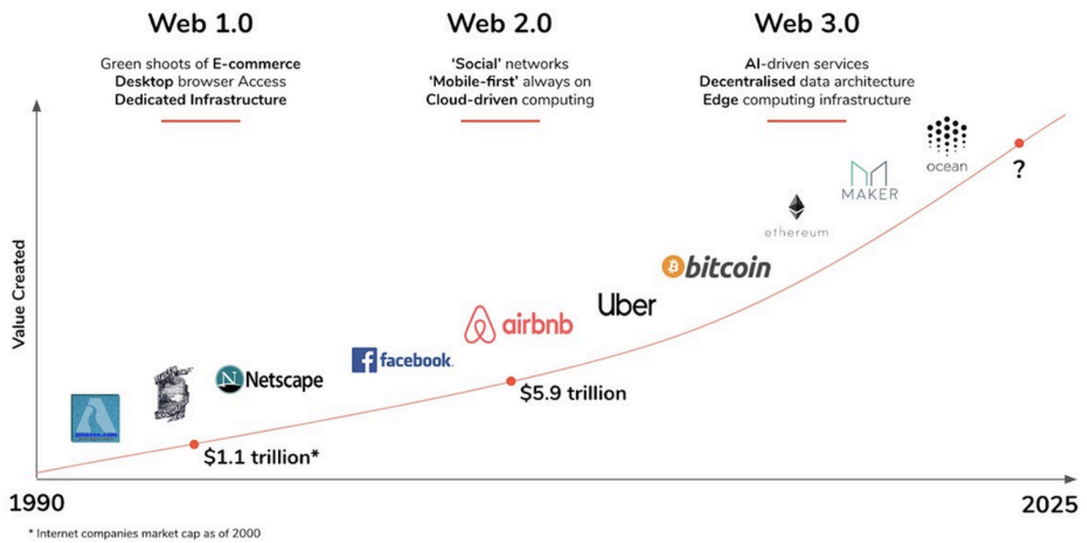


FIRME DIGITALI, FUNZIONI DI HASHING E ALGORITMI DI CONSENSO

Blockchain: Fondamenta e Crittografia

Integrità, Sicurezza e Decentralizzazione

The Evolution of the Web



Startups drive the transition to web 3.0

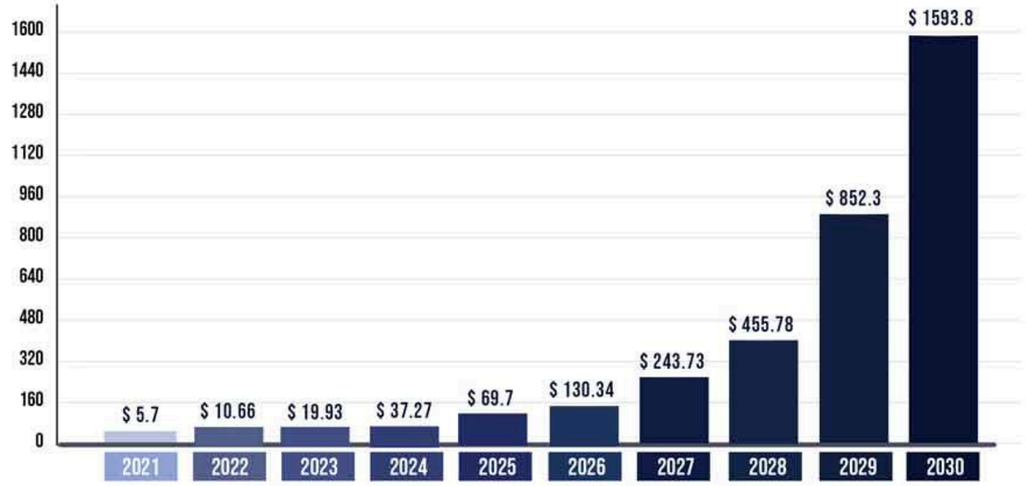


Source: GlobalData FutureTech Series Report

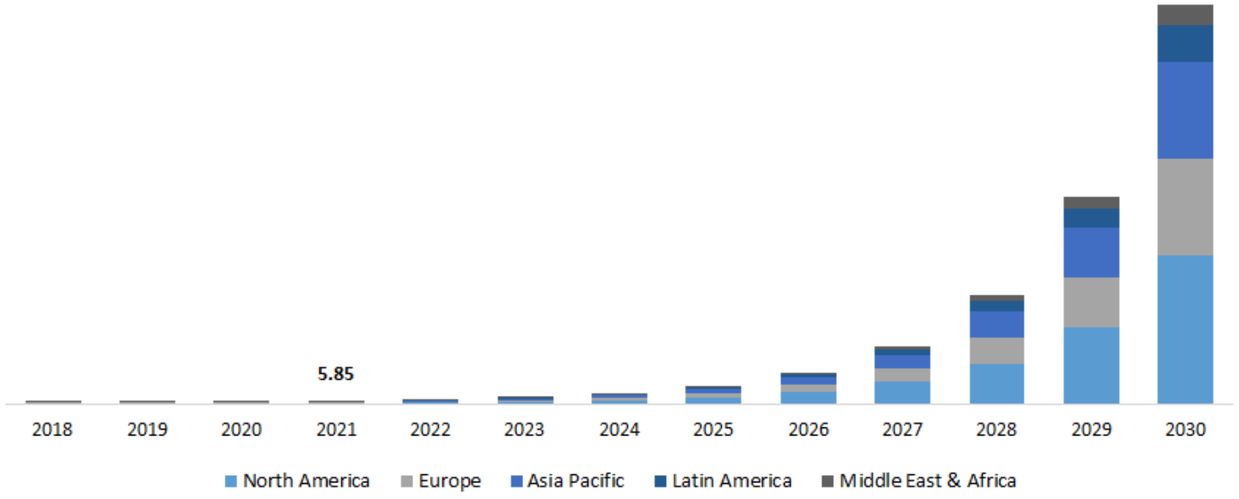
GlobalData.

PRECEDENCE RESEARCH

BLOCKCHAIN TECHNOLOGY MARKET SIZE, 2021 TO 2030 (USD BILLION)



Blockchain Technology Market Size, By Region, 2018 - 2030 (USD Billion)



Source: Polaris Market Research Analysis

EU digital finance strategy

- Promoting innovation in the financial sector
- Consumer protection and legal certainty
- Market integrity and financial stability

Digital Operational Resilience Act (DORA)

Aim: ensuring the resilience and performance of financial institutions' IT

Entry into force: 2023 at the earliest

Pilot regime for DLT market infrastructures (DLT pilot regime)

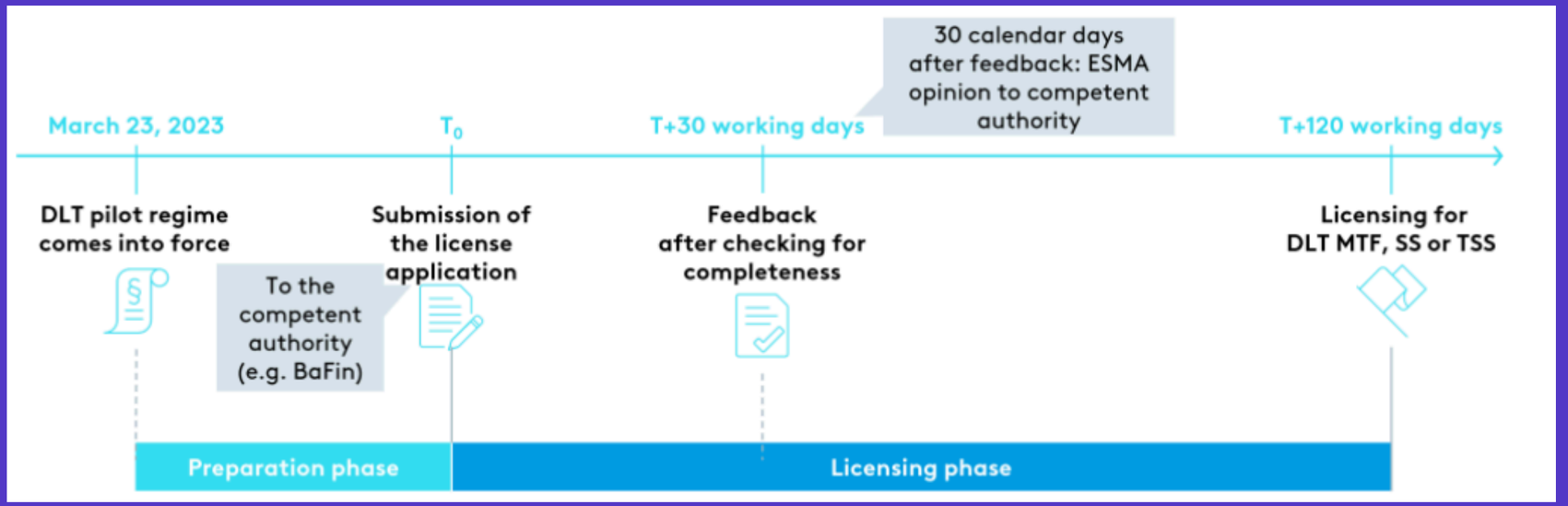
Aim: allowing exemptions from existing regulatory requirements for DLT market infrastructures

Entry into force: March 23, 2023

Markets in Crypto-Assets Regulation (MiCAR)

Aim: establishing a legal framework for crypto-assets to promote transformative technologies

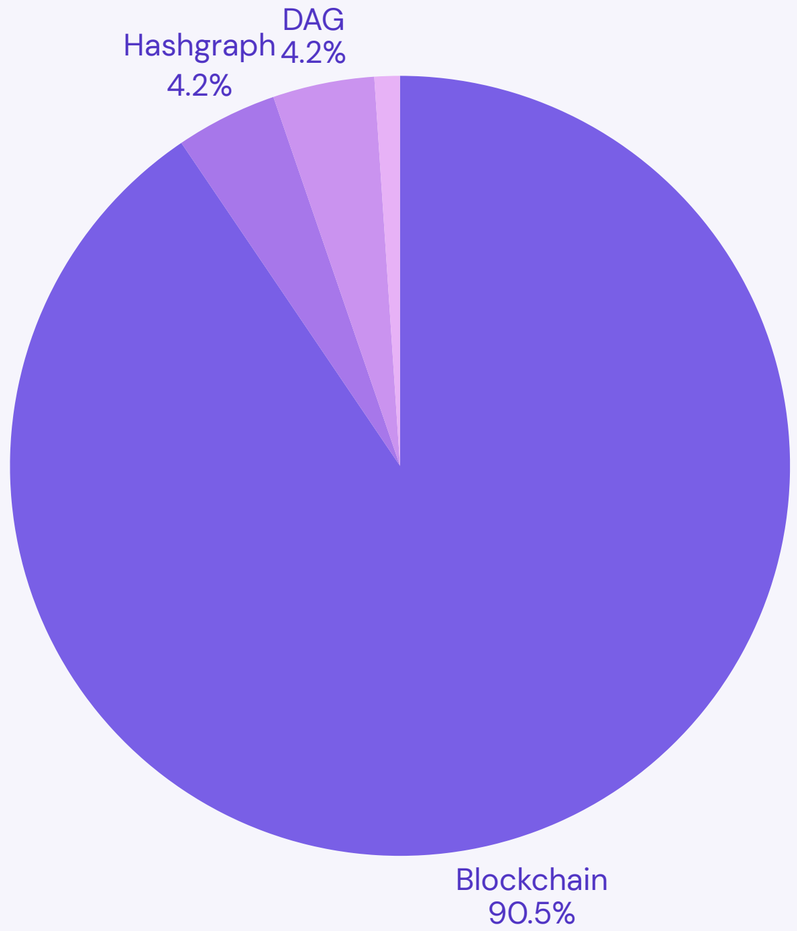
Entry into force: 2024 at the earliest



Distributed Ledger Technologies

DLT è l'acronimo di Distributed Ledger Technologies e si riferisce a registri transazionali, digitali e distribuiti che memorizzano dati su una rete di dispositivi.

La Blockchain è un tipo di DLT.



La storia in breve:

- David Chaum 1982 "Computer Systems Established, Maintained, and Trusted by Mutually Suspicious Groups.".
- 1992, Haber, Stornetta e Dave Bayer incorporano gli alberi di Merkle in un sistema di timestamping per documenti digitali da loro sviluppato in precedenza.
- La prima blockchain decentralizzata è stata concettualizzata da una persona o gruppo di persone nota come Satoshi Nakamoto.

Nel 2008, Nakamoto ha concettualmente espanso il sistema di timestamping con alberi di Merkle integrando un metodo simile a Hash-cash per contrassegnare i blocchi senza richiedere che fossero firmati da una parte fidata e introducendo un parametro di difficoltà per stabilizzare la velocità con cui i blocchi vengono aggiunti alla chain.

Il progetto, che funge da libro mastro pubblico per tutte le transazioni su una rete, è stato implementato l'anno successivo da Nakamoto come componente principale della criptovaluta Bitcoin.



BLOCKCHAIN

Una catena di blocchi che garantisce **l'integrità e l'immutabilità** dei dati all'interno di ogni blocco è nota come blockchain.

Ogni blocco è collegato agli altri attraverso un **hash crittografico** del blocco precedente, un timestamp e dei dati di transazione.

Le blockchain sono attualmente utilizzate in diverse applicazioni principalmente per creare registri distribuiti che **promuovono l'immutabilità dei dati, la loro ridondanza e la trasparenza dei record agli utenti della blockchain.**
I record, spesso indicati come transazioni, vengono archiviati in un **registro append-only** chiamato libro mastro o distributed ledger. **Il libro mastro viene replicato attraverso una rete peer-to-peer in cui ogni nodo mantiene una copia del ledger** (o una parte) formando così un database distribuito di tutte le transazioni. Viene implementato un **protocollo di consenso** per decidere come aggiungere nuove transazioni alla blockchain e come convalidare le transazioni esistenti.

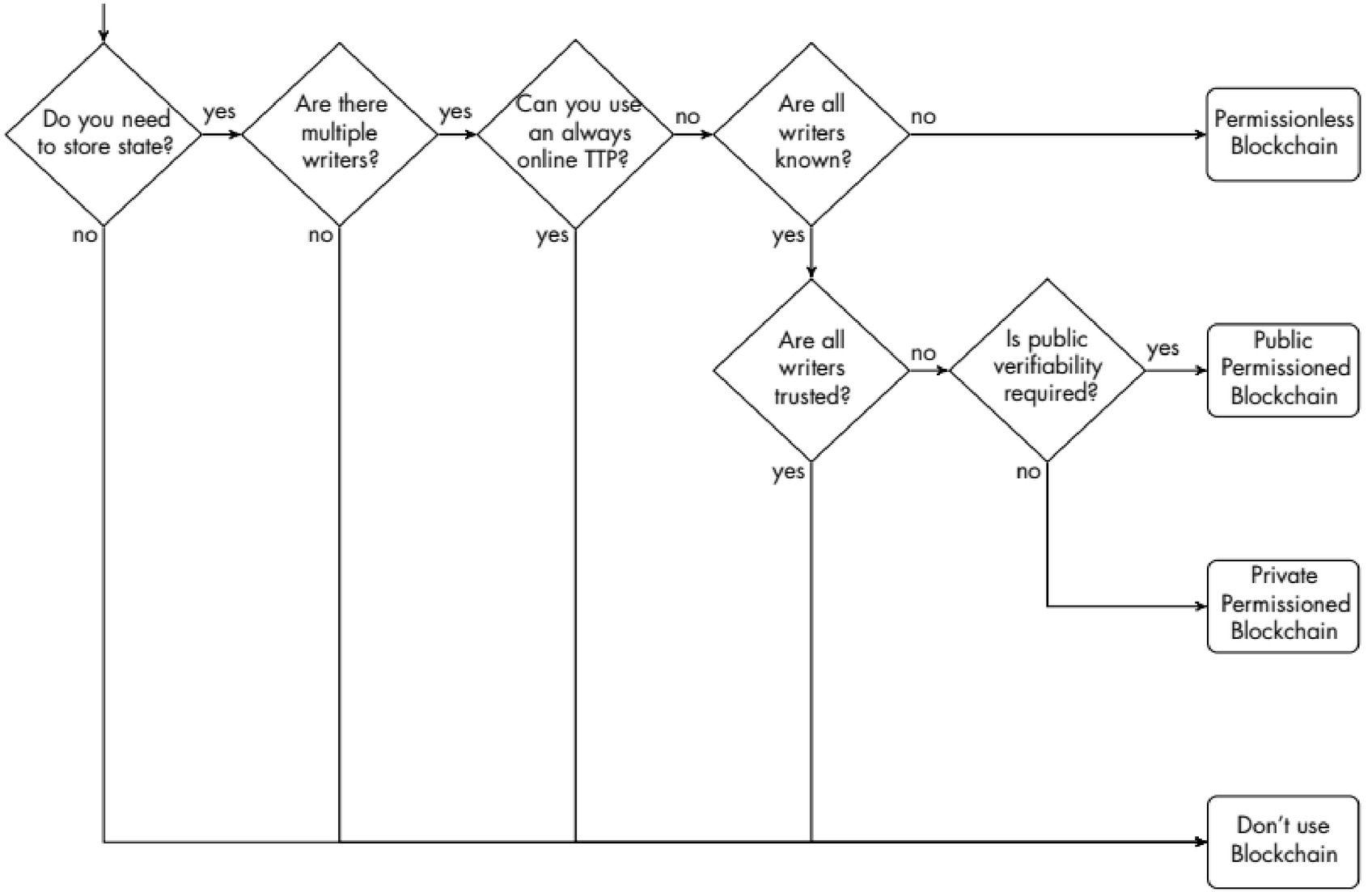
La rete peer-to-peer è composta da diverse entità, i nodi, che possono essere di diversi tipi, ad esempio un full-node conserva una copia di tutte le transazioni della blockchain, un half-node solo le più recenti. Per l'interconnessione dei nodi viene utilizzata una rete overlay, in cui ogni nodo di rete è connesso a un insieme modificabile di vicini in modalità peer-to-peer. Per la distribuzione dei dati tra peer vicini viene comunemente utilizzato un **algoritmo di flooding chiamato protocollo di gossip.**

Accessibilità, Consistenza e Resistenza alle Partizioni. ***-> Teoria dei Sistemi distribuiti e parallelismo***

CATEGORIE DI BLOCKCHAIN

Various kinds:

- Public/Private
- Permissioned/Permissionless
- Decentralized/Centralized



Bitcoin:

La prima generazione



LA PRIMA GENERAZIONE, IL BITCOIN



01 Blocchi con un proprio hash, generato utilizzando double SHA-256 sull'intestazione del blocco; hash che viene memorizzato nell'intestazione del blocco successivo formando una catena.

02 Transazioni che usano il modello UTXO, **firmate digitalmente con ECDSA.**

03 Consenso **Proof of Work** che determina come ogni transazione debba essere elaborata, convalidata, registrata e che delinea inoltre come si debba comportare la rete in caso di adulterazione dei registri.

04 **Chiavi pubbliche,** che identificano i partecipanti in modo pseudo anonimo, l'hashing di una chiave pubblica è l'address di un wallet.

What are the types and functions of CRYPTO WALLETS

1 ONLINE WALLETS

Online wallets can be accessed through multiple devices and suit our convenience. They are also vulnerable to cyber-attacks and need an extra layer of security.

2 MOBILE WALLETS

You need to open an account to access the mobile wallet through your phone and they are much safer than cold wallets

3 HARWARE WALLETS

Hardware wallets are offline wallets. They are available based on the compatibility with various web interfaces and allows the users to store their private keys.

4 DESKTOP WALLET

Desktop wallets are safer than online wallets and are easily accessible. They are downloaded and installed on desktop computers.

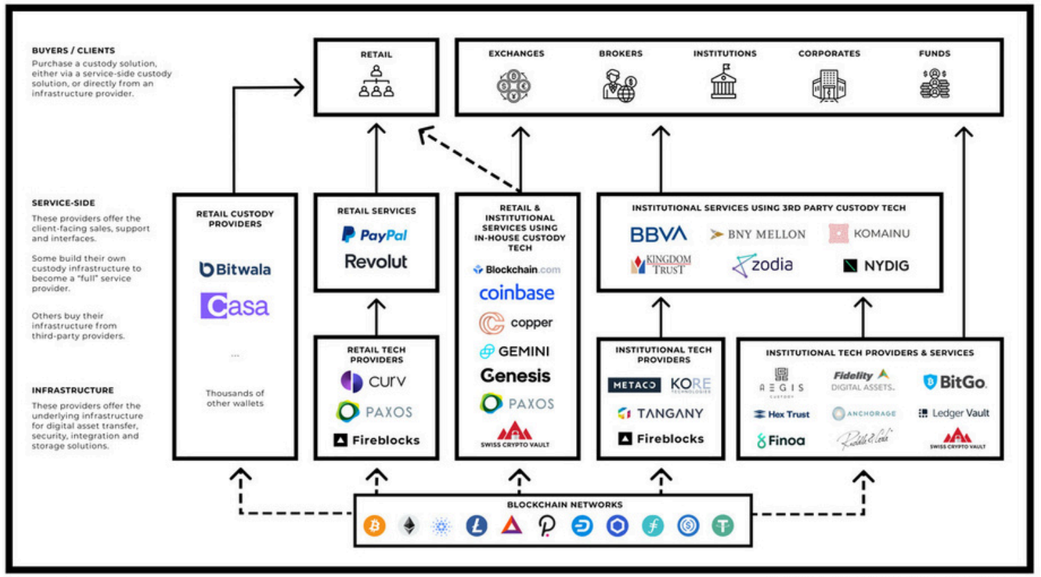
5 PAPER WALLETS

Paper wallets are the safest option available to store your digital assets. It's a hard copy of your generated public and private key.

Learn

BLOCKDATA

WHO SERVICES WHO IN THE CUSTODY BUSINESS?

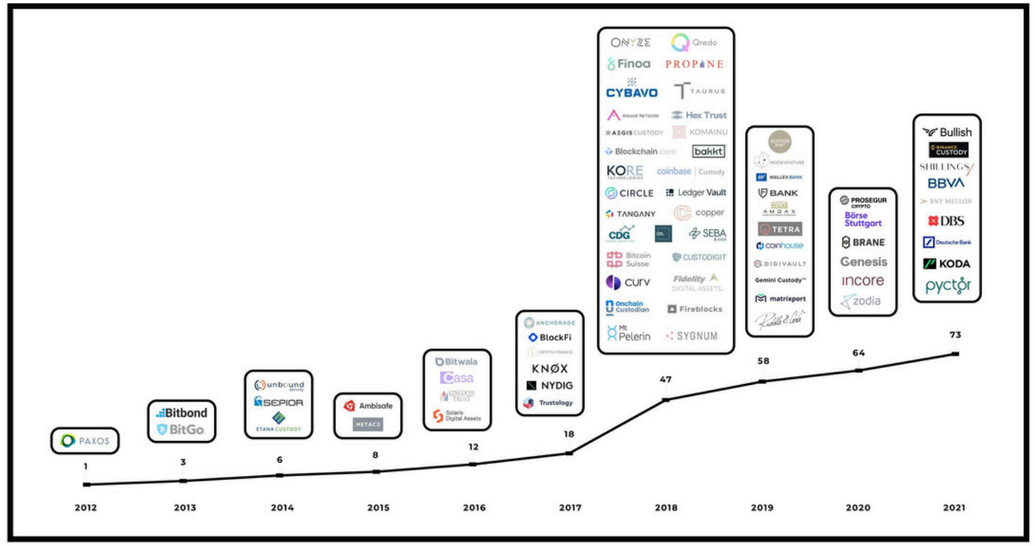


BLOCKDATA IS A CB INSIGHTS COMPANY

WWW.BLOCKDATA.TECH | INFO@BLOCKDATA.TECH

BLOCKDATA

CRYPTO CUSTODY SERVICES BY LAUNCHING YEAR

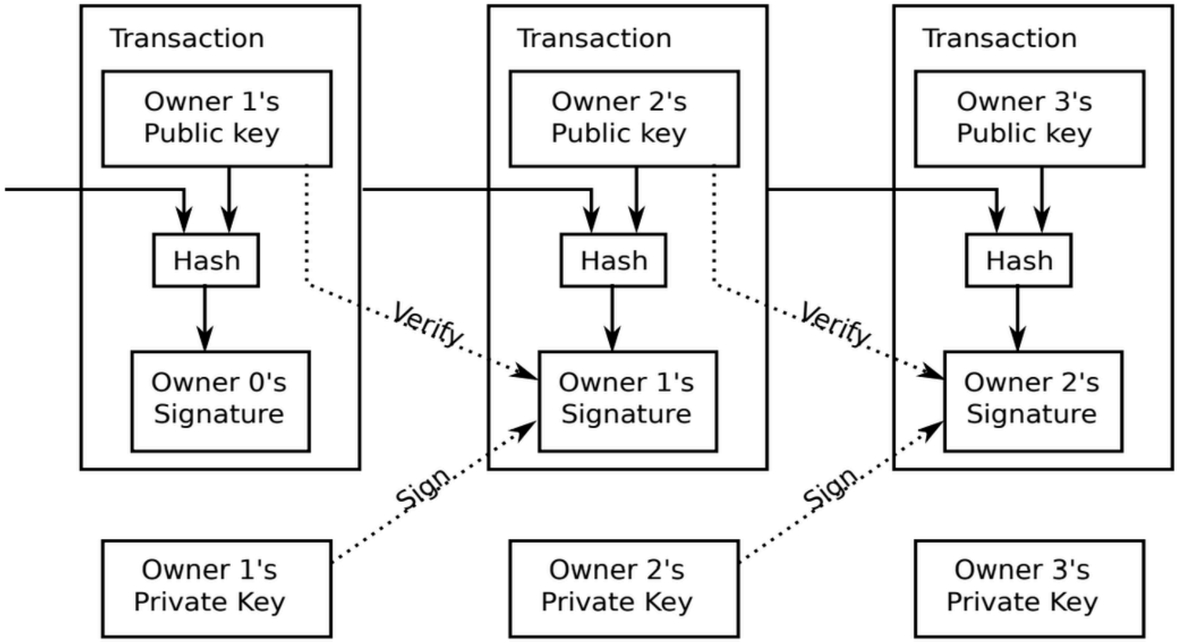


BLOCKDATA IS A CB INSIGHTS COMPANY

WWW.BLOCKDATA.TECH | INFO@BLOCKDATA.TECH

Transazioni bitcoin

- People in bitcoin are identified by their public keys.
- Suppose Alice to have keys (pk_A, sk_A) and Bob to have keys (pk_B, sk_B).
- A payment form Alice to Bob is a message m containing as payee pk_B signed with sk_A and indicating a certain number of bitcoins to be transferred.
- The database is updated by subtracting the number of bitcoir paid by Alice from the account pk_A and crediting the same amount on the account pk_B.
- Digital signatures guarantee authentication.
- Transactions are batched in *blocks*.



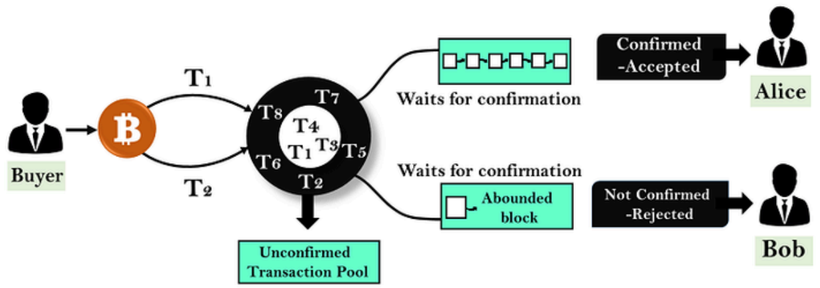
Double Spend e Generali Bizantini

La doppia spesa è un problema che si verifica quando si effettuano transazioni di valuta digitale che implicano che la stessa "moneta" venga spesa più volte. Transazioni multiple che condividono lo stesso input trasmesso sulla rete possono essere problematiche ed è un difetto unico delle valute digitali.

Un protocollo di accordo bizantino è un protocollo distribuito tra gli agenti A1, . . . , Ak tale che:

- Tutti gli agenti non difettosi/onesti terminano il protocollo in un numero finito di passaggi;
- Tutti gli agenti non difettosi/onesti concordano sulla risoluzione sulla stessa decisione;
- Sono possibili diverse decisioni.

La decisione presa è irrilevante fintanto che tutti gli agenti non difettosi sono d'accordo.



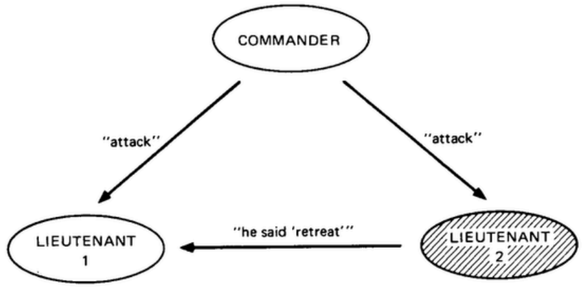
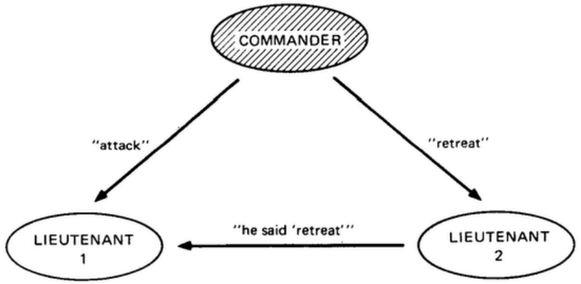


Fig. 1. Lieutenant 2 a traitor.



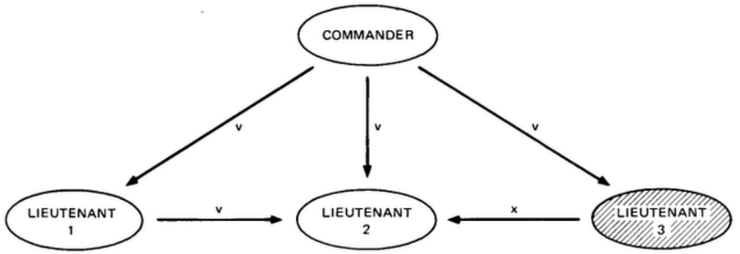


Fig. 3. Algorithm OM(1); Lieutenant 3 a traitor.

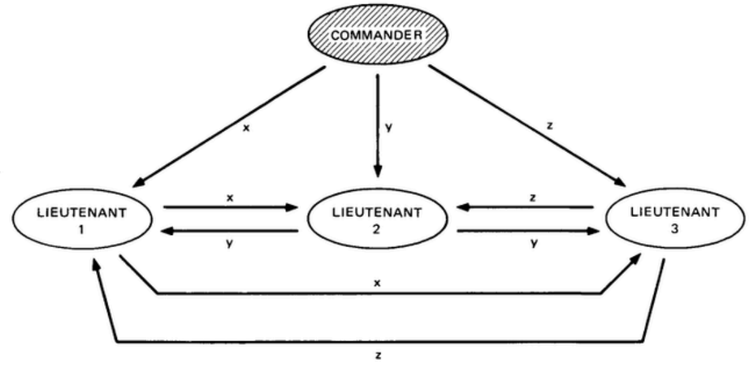


Fig. 4. Algorithm OM(1); the commander a traitor.

Authenticated Byzantine

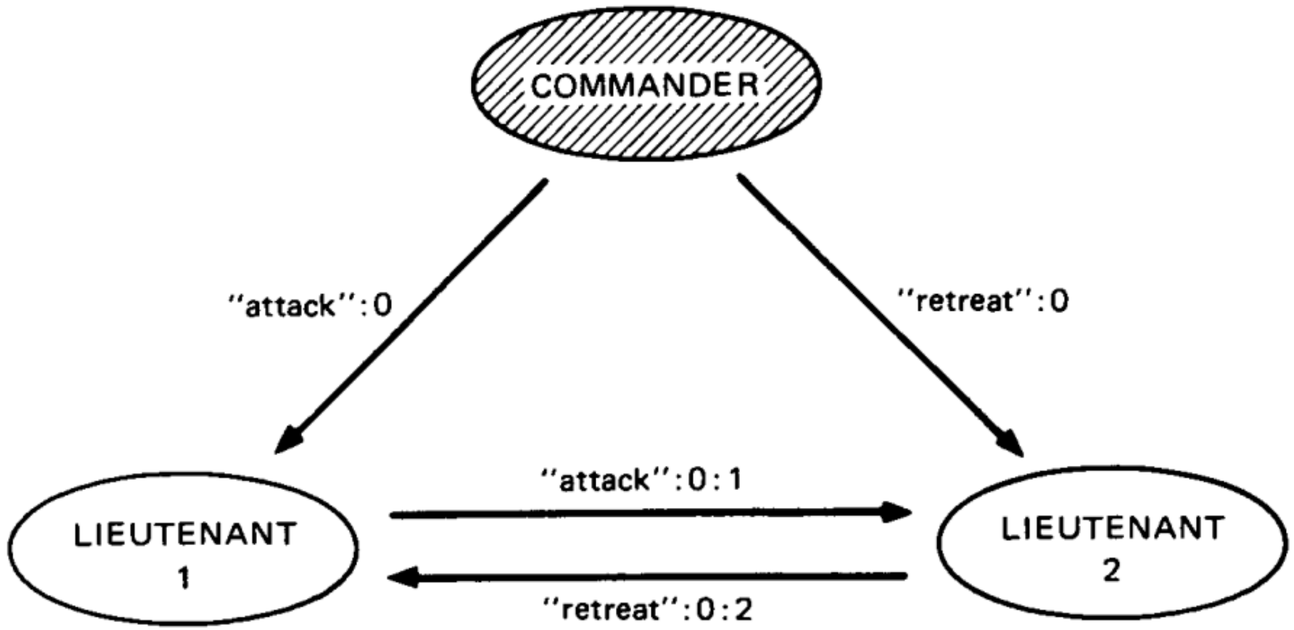
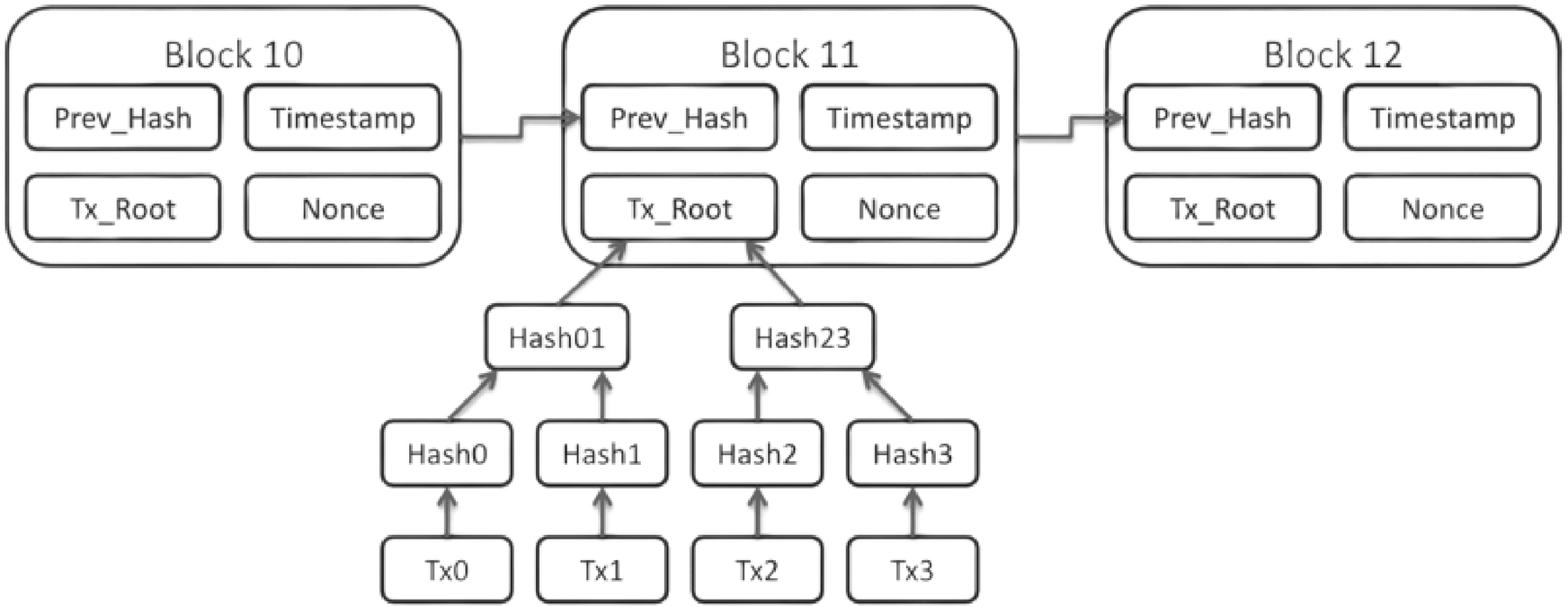


TABLE I. SMALLEST NUMBER OF PROCESSORS N_{min} FOR WHICH A t-RESILIENT CONSENSUS PROTOCOL EXISTS

Failure type	Synchronous	Asynchronous	Partially synchronous communication and synchronous processors	Partially synchronous communication and processors	Partially synchronous processors and synchronous communication
Fail-stop	t	∞	$2t + 1$	$2t + 1$	t
Omission	t	∞	$2t + 1$	$2t + 1$	$[2t, 2t + 1]$
Authenticated Byzantine	t	∞	$3t + 1$	$3t + 1$	$2t + 1$
Byzantine	$3t + 1$	∞	$3t + 1$	$3t + 1$	$3t + 1$

Struttura di un Blocco





- Proof of Work
- Proof of Burn
- Proof of Stake
- Proof of Authority (endorsement)

Proof of Work

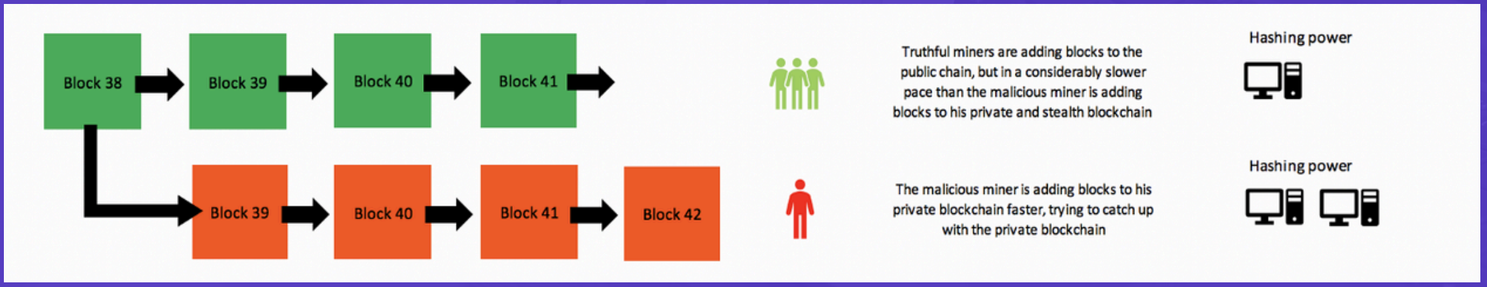
Cos'é e come funziona

PROOF OF WORK:

Al centro dell'architettura di Bitcoin ci sono le funzioni di hash; dato un determinato insieme di dati in ingresso, attraverso una funzione di hash (bitcoin utilizza **SHA-256, Ethereum SHA3-Keccac**) si genera una stringa "unica per quell'insieme" di lunghezza fissata.

Generare un qualsiasi hash per un insieme di transazioni Bitcoin sarebbe banale per un computer moderno, quindi per trasformare il processo in "lavoro", la rete di Bitcoin imposta un certo livello di "**difficulty**".
Questa impostazione viene regolata in modo che un nuovo blocco venga "estratto", quindi aggiunto alla blockchain conseguentemente alla generazione di un hash valido, circa ogni 10 minuti.
Dal momento che un determinato insieme di dati in ingresso "può generare solo un hash", **i miner alterano l'input aggiungendo un numero intero, chiamato Nonce. T***rovato un hash valido, viene trasmesso alla rete e il blocco associato viene aggiunto alla blockchain.*

I miner si uniscono per aumentare le loro chance di estrarre blocchi, per generare commissioni di transazione e una ricompensa in bitcoin. PoW rende estremamente difficile alterare qualsiasi aspetto della blockchain poiché tale alterazione richiederebbe il re-mining di tutti i blocchi successivi al blocco alterato, PoW rende inoltre difficile la monopolizzazione della potenza di calcolo nella rete. Se parte della rete inizia ad accettare una PoW alternativa (un hash valido), si parla di **fork (Soft / Hard). Nel caso si verifichi un fork, viene mantenuta la catena più lunga.**



A Byzantine attacker in bitcoin aims to keep a fork alive; ultimately this costs more than the expected gain.

Bitcoin non è perfetto

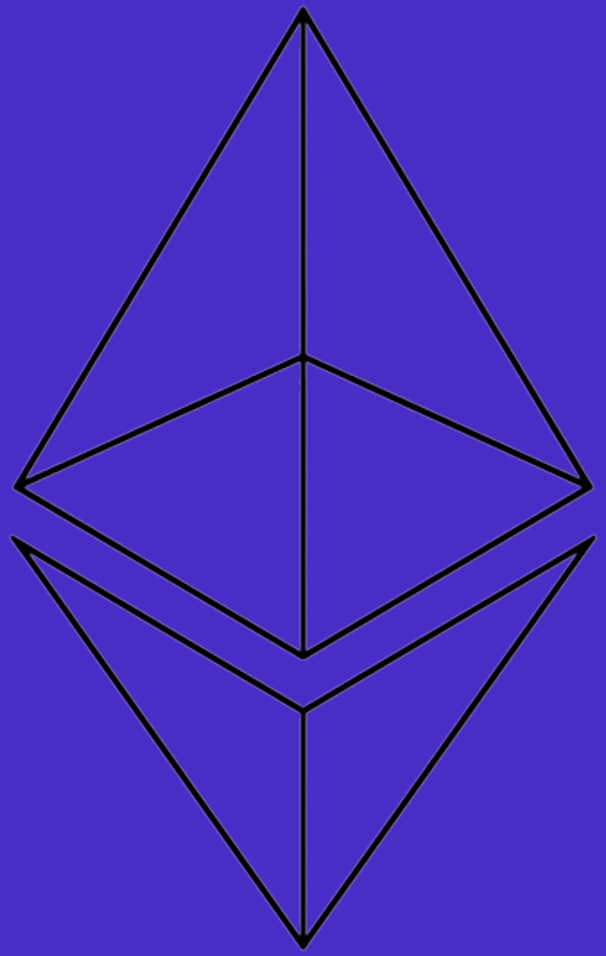
01 L'assenza di Smart-Contracts implica un' alta complessità di gestione e sviluppo di dApps (dove è possibile)

02 La velocità di elaborazione delle transazioni è molto ridotta (< 10 TPS)

03 È incompatibile con altre chain

04 PoW è costoso da sostenere sia in termini di hardware che in termini energetici

La Seconda Generazione: Ethereum



BITCOIN HA GETTATO LE BASI DI UNA TECNOLOGIA CHE HA FATTO PROGRESSI NOTEVOLI

Vitalik Buterin



Ethereum è stato concepito nel 2013 da Vitalik Buterin in collaborazione con Gavin Wood, Charles Hoskinson, Anthony Di Iorio, Joseph Lubin.
Nel 2014 sono iniziati i lavori di sviluppo che sono stati finanziati tramite crowdfunding e la rete è stata attivata il 30 luglio 2015.

SMART CONTRACTS

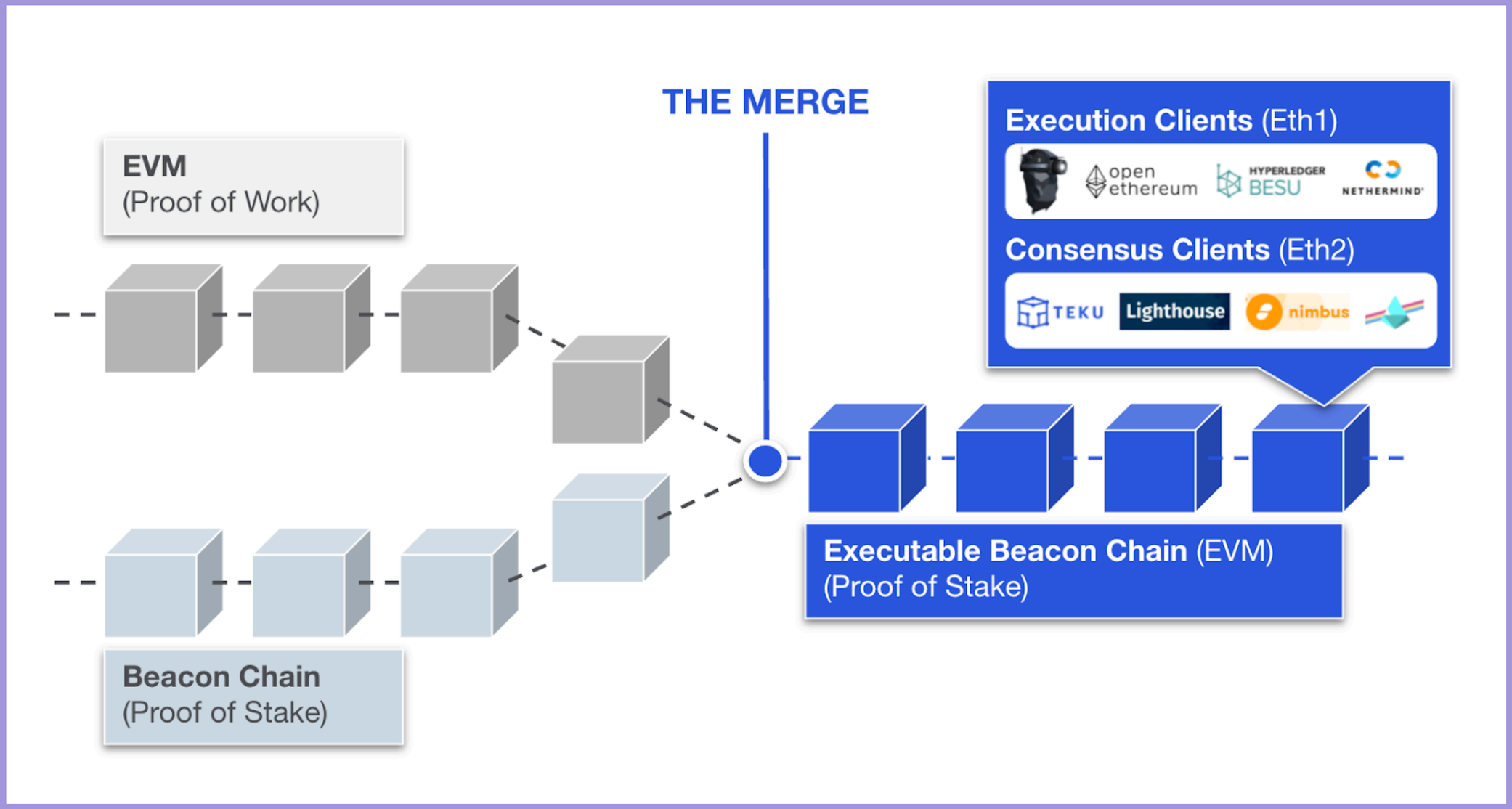
Nick Szabo ha coniato per la prima volta il termine "Smart Contract" nel 1997; in una Blockchain gli smart contracts consentono lo sviluppo di logica di business sotto forma di codice immutabile, pubblico con logica IFTTT.

TRANSAZIONI

Le transazioni sulla rete Ethereum possono contenere codice eseguibile, una transazione ETH è confermata in secondi.

CONSENSO

A Settembre 2022, la Beacon Chain (chain introdotta nel sistema Ethereum nel 2020 per testare il sistema PoS) è stata configurata per gestire le transazioni della catena Ethereum originale PoW; nello stesso momento, i client originali di Ethereum hanno disattivato il mining, la propagazione dei blocchi e la logica di consenso, delegando tutti questi aspetti alla Beacon Chain. Questo evento è noto come "Merge". Una volta verificatosi, ne è conseguita un'unica chain Proof of Stake che ha incrementato la scalabilità e ridotto (notevolmente) il consumo energetico del sistema.


THE MERGE
EVM
(Proof of Work)
Execution Clients (Eth1)
open ethereum
HYPERLEDGER BESU
NETHERMIND
Consensus Clients (Eth2)
TEKU
Lighthouse
nimbus
Executable Beacon Chain (EVM)
(Proof of Stake)
Beacon Chain
(Proof of Stake)

ETHEREUM'S YEARLY ENERGY CONSUMPTION

Proof-of-Work (current)

82.1 TWh

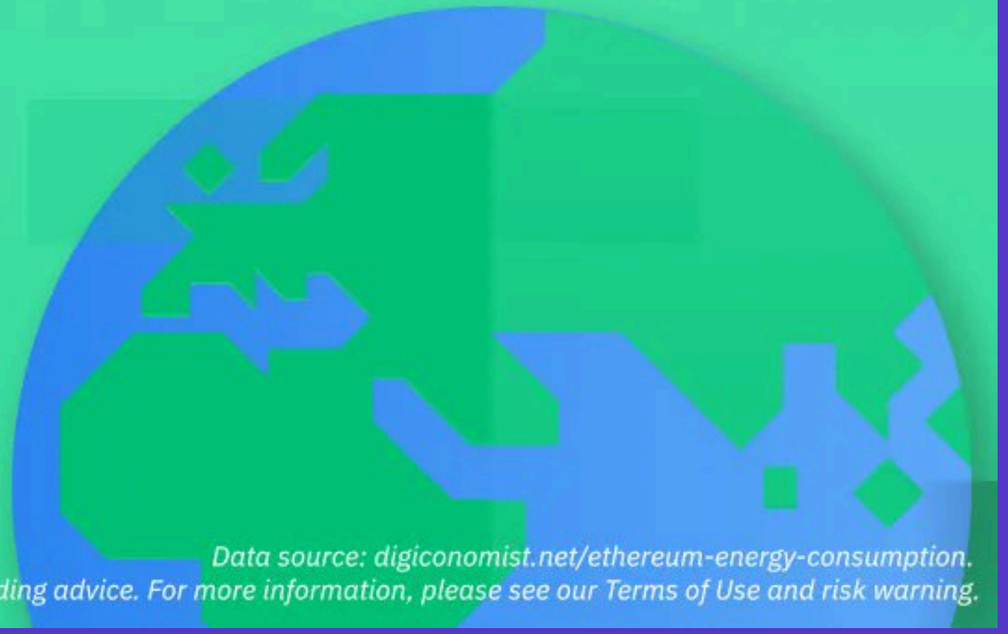
Equivalent to a country with 19M+ inhabitants



Proof-of-Stake (after upgrades)

~99.95%

Targeted energy reduction



Data source: digiconomist.net/ethereum-energy-consumption.

Not trading advice. For more information, please see our Terms of Use and risk warning.

Smart Contracts

Uno **"smart contract" è un tipo di account** (in questo caso Ethereum), **una raccolta di funzioni e dati** (stato) che risiede in **un indirizzo specifico sulla blockchain** di Ethereum; gli smart contracts hanno un saldo e **possono essere oggetto di transazioni**.

Un account Ethereum è un'entità con un saldo in Ether (ETH) **che può inviare transazioni su Ethereum**. Gli account possono essere controllati dall'utente o essere "smart contract".

Gli ***account utente possono interagire con uno smart contract inviando transazioni che eseguono una funzione ivi definita, gli smart contracts possono definire regole e applicarle automaticamente.*** **Gli smart contract sono immutabili e le interazioni con essi sono irreversibili.**

L'implementazione di uno smart contract è tecnicamente una transazione, quindi bisogna pagare "gas" (commissioni). Il "gas" per l'implementazione di uno smart contract è molto più elevato di quello per una normale transazione..

```
pragma solidity 0.8.7;

contract VendingMachine {

    // Declare state variables of the contract
    address public owner;
    mapping (address => uint) public cupcakeBalances;

    // When 'VendingMachine' contract is deployed:
    // 1. set the deploying address as the owner of the contract
    // 2. set the deployed smart contract's cupcake balance to 100
    constructor() {
        owner = msg.sender;
        cupcakeBalances[address(this)] = 100;
    }

    // Allow the owner to increase the smart contract's cupcake balance
    function refill(uint amount) public {
        require(msg.sender == owner, "Only the owner can refill.");
        cupcakeBalances[address(this)] += amount;
    }

    // Allow anyone to purchase cupcakes
    function purchase(uint amount) public payable {
        require(msg.value >= amount * 1 ether, "You must pay at least 1 ETH
        require(cupcakeBalances[address(this)] >= amount, "Not enough cupcakes
        cupcakeBalances[address(this)] -= amount;
        cupcakeBalances[msg.sender] += amount;
    }
}
```

https://ethereum.org/en/developers/docs/smart-contracts/

Smart Contracts e Oracoli:

Gli smart contracts non possono ottenere autonomamente informazioni sugli eventi esterni (off-chain). *Restringere gli input delle dApps esclusivamente ai dati presenti on-chain ne limita però il potenziale in modo notevole, negli anni si sono quindi esplorati dei modi per bypassare la natura "chiusa" degli smart contracts.*

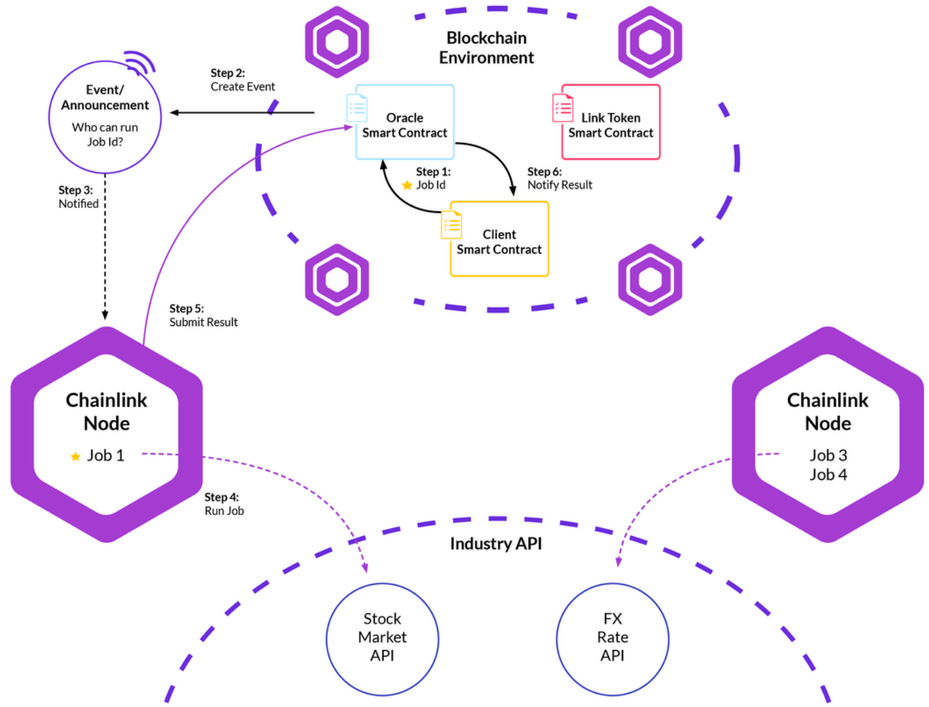
L'integrazione non è però banale perché l'inserimento di dati on-chain può diventare una finestra per gli hacker, aprendo alla possibilità di diffusione di false informazioni sulla rete; questo scenario può risultare disastroso se le suddette informazioni sono il trigger di uno smart contract.

Gli oracoli blockchain sono servizi di terze parti che forniscono informazioni esterne a smart contract. Fungono da ponti tra le blockchain e il mondo esterno.

Un oracolo non è la fonte di dati stessa, ma piuttosto il livello che richiede, verifica e autentica le fonti di dati esterni per poi trasmettere le informazioni.

Gli oracoli blockchain possono essere classificati a seconda di una serie di qualità diverse: Fonte – i dati provengono da software o hardware? Direzione delle informazioni – sono in entrata o in uscita? Fiducia – è centralizzato o decentralizzato?

CHAINLINK





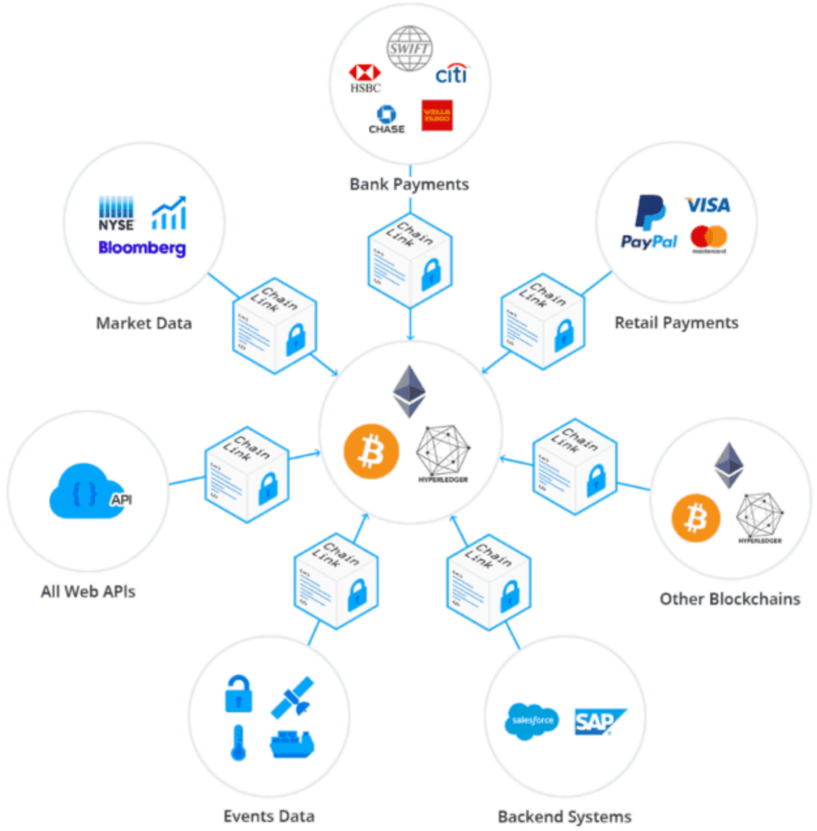
The DAO

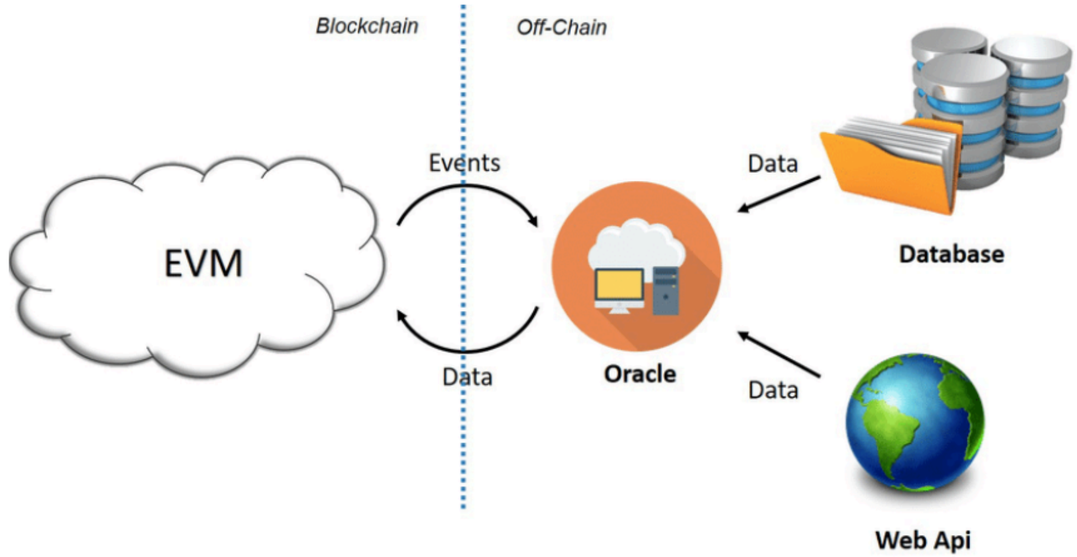
The DAO was a digital decentralized autonomous organization and a form of…

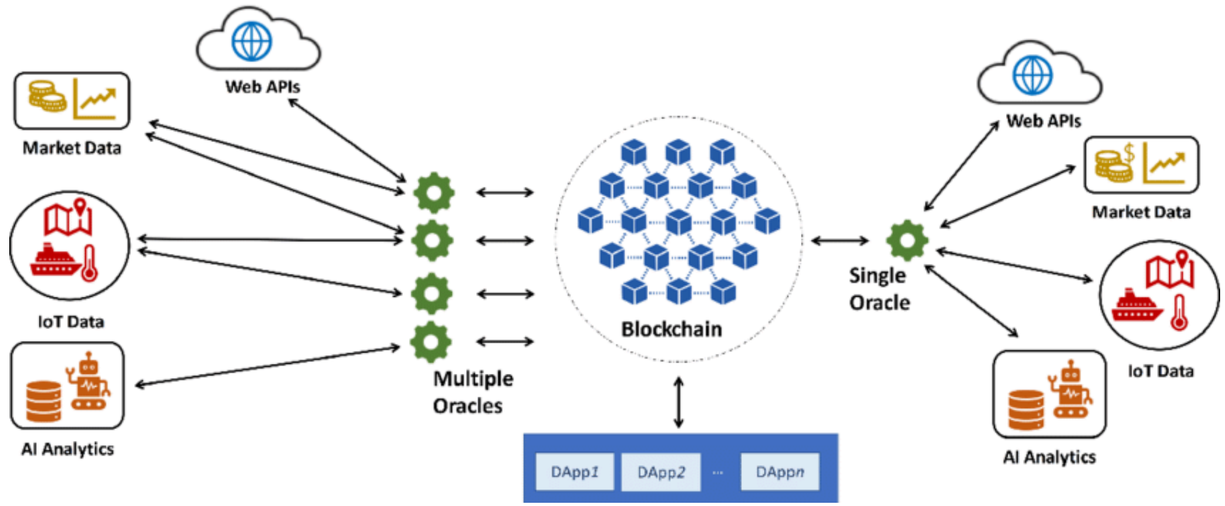
en.wikipedia.org

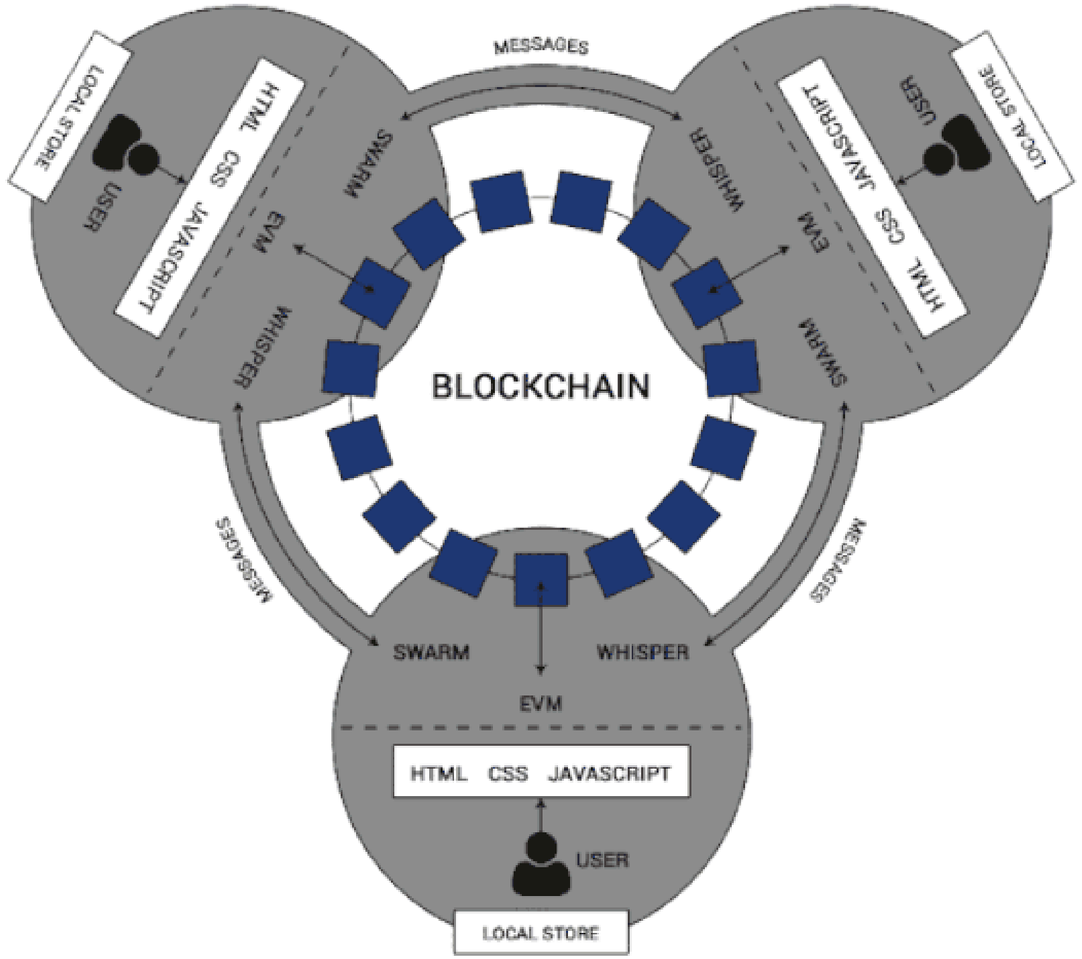
https://ethereum.org/en/developers/docs/smart-contracts/
https://academy.binance.com/it/articles/blockchain-oracles-explained#what-is-a-blockchain-oracle

SMART CONTRACTS
ORACOLI E EVM



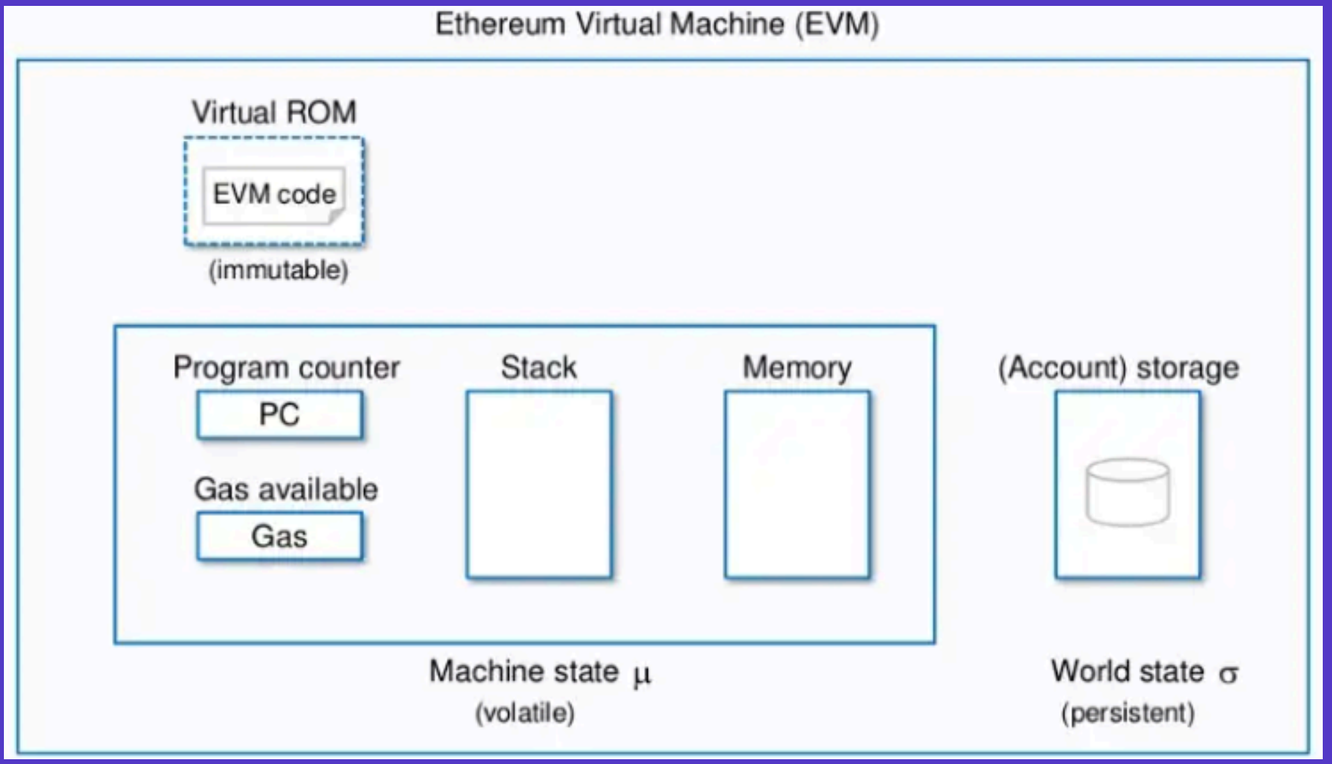






Ethereum Virtual Machine

Virtual Machine su Blockchain, un idea che ha permesso lo sviluppo di diversi, complessi ecosistemi



L'EVM è una macchina a stati:
Quando una transazione provoca l'esecuzione di uno smart contract, l'EVM modifica lo stato della blockchain Ethereum per soddisfare le esigenze di chiamata.

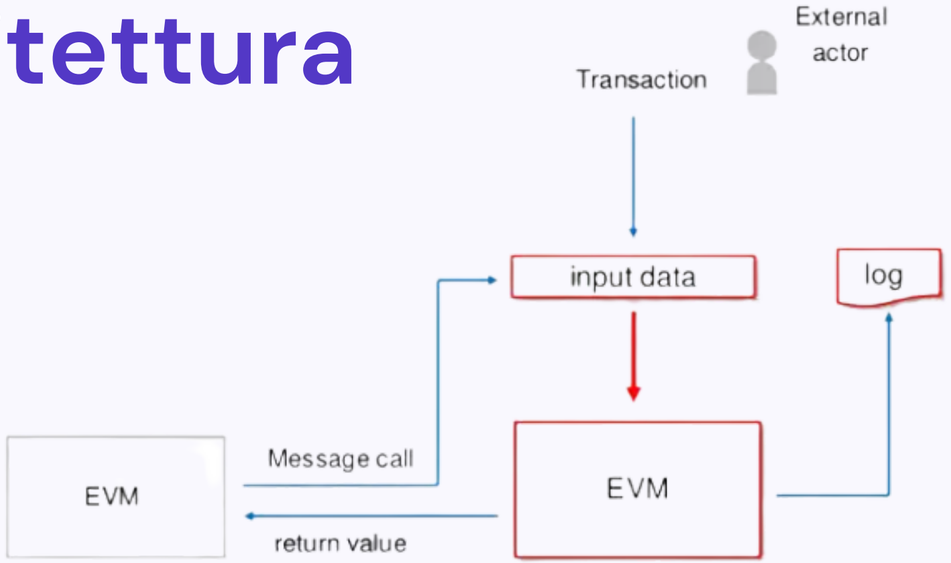
La funzione di transizione di stato dell'EVM consente a Ethereum di aggiornarsi a un nuovo stato valido da blocco a blocco, in risposta ai dati di input di uno smart contract.

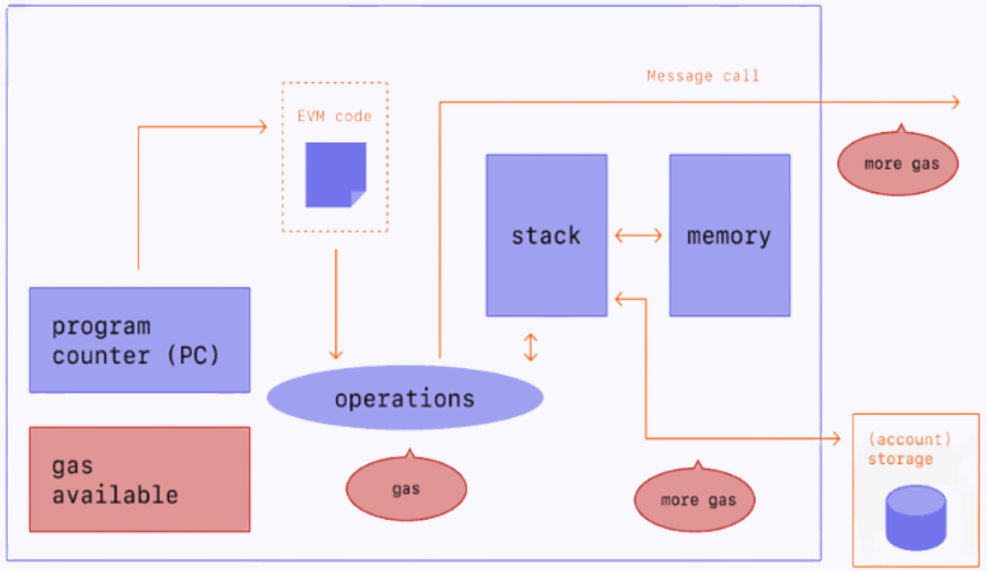
L'EVM è una **macchina turing completa a stack**.
Lo **stack ha una dimensione massima di 1024 elementi**, gli **elementi dello stack hanno una dimensione di 256 bit.**
L'EVM è quindi una **word machine a 256 bit.**

Il codice sorgente scritto in Solidity, Vyper o altri LP viene tradotto in Opcode (140 codici) comprensibile all'EVM, che a sua volta viene tradotto in binario eseguibile. Alla base del sistema EVM c'è il concetto di gas, l'unità di misura della quantità di sforzo computazionale richiesto per eseguire operazioni sulla rete Ethereum **(Problema della fermata)**.

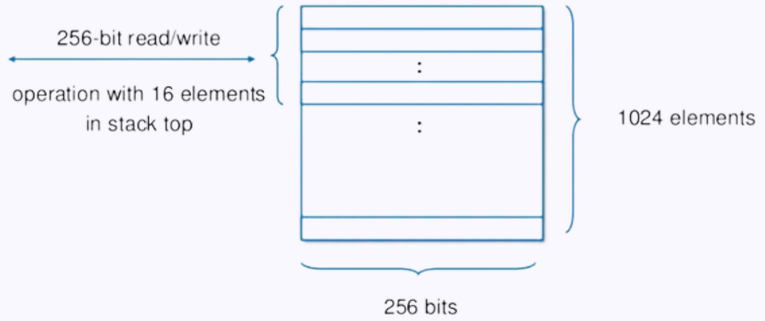
https://medium.com/swlh/getting-deep-into-evm-how-ethereum-works-backstage-ab6ad9c0d0bf

Architettura EVM





Stack



Stack



stack memory

256 bits x 1024 elements

Memory



volatile memory

byte addressing
linear memory

(Account) storage



persistent memory

256 bits to 256 bits
key-value store

All operation are performed on the stack.
Access with many instructions such as PUSH/POP/COPY/SWAP

https://medium.com/swlh/getting-deep-into-evm-how-ethereum-works-backstage-ab6ad9c0d0bf
https://ethereum.org/en/developers/docs/gas/

Standard nei Token:

La community di Ethereum ha adottato molti standard che aiutano a mantenere interoperabili i progetti (come i client di Ethereum e i portafogli) tra le implementazioni e ad assicurarsi che i contratti intelligenti e le dapp restino componibili.

Token, nel contesto della conformità ERC-20, indica semplicemente una rappresentazione blockchain di qualcosa che soddisfa gli standard sugli smart contract stabiliti dalla comunità Ethereum.

Molti standard di sviluppo Ethereum si incentrano sulle interfacce dei token. Questi standard aiutano ad assicurarsi che i contratti intelligenti restino componibili, così che, ad esempio, quando un nuovo progetto emette un token, esso rimanga compatibile con le borse decentralizzate esistenti.

01
- ERC-20 - Un'interfaccia standard per token fungibili (intercambiabili), come i token di voto, i token di staking o le valute virtuali.

02
- ERC-721 - Un'interfaccia standard per token non fungibili, come un atto relativo a opere d'arte o canzoni.

03
- ERC-777 - Uno standard per i token migliorato rispetto a ERC-20.

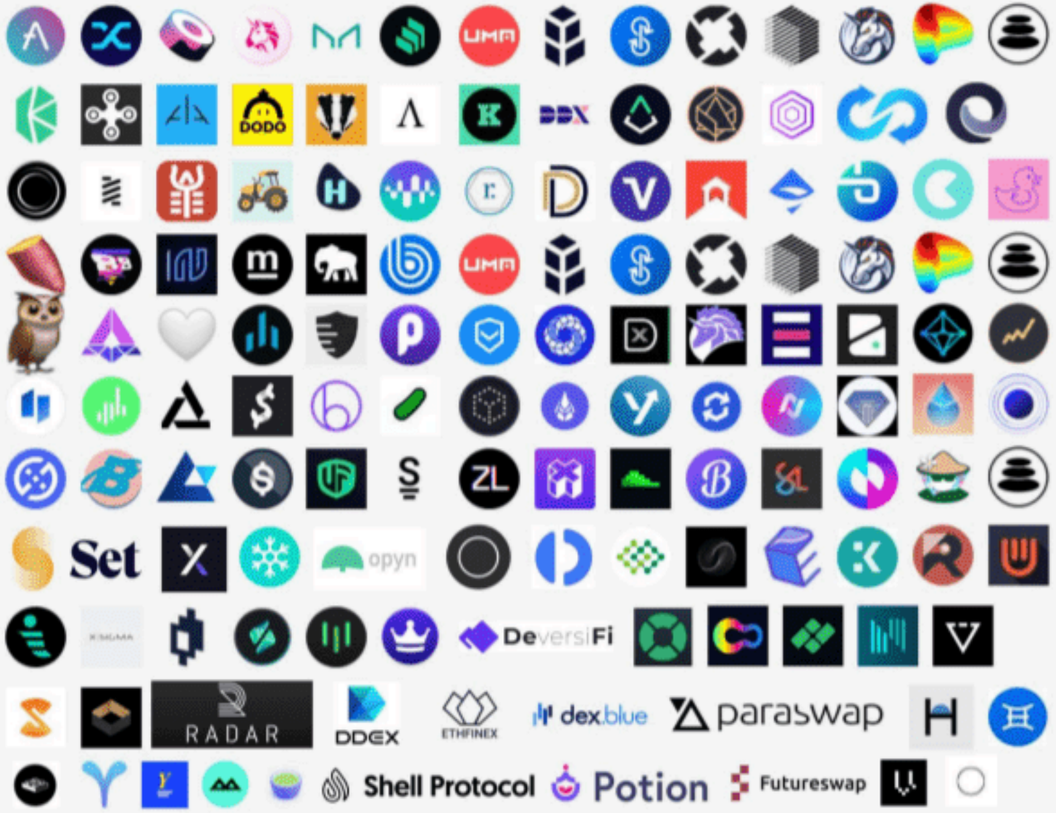
04
- ERC-1155 - Uno standard per i token che può contenere risorse sia fungibili che non fungibili.

05
- ERC-4626 - Uno standard di cassaforte tokenizzata progettato per ottimizzare e unificare i parametri tecnici delle cassaforti di resa.

https://ethereum.org/en/developers/docs/standards/

ETH ECOSYSTEM

DeFi



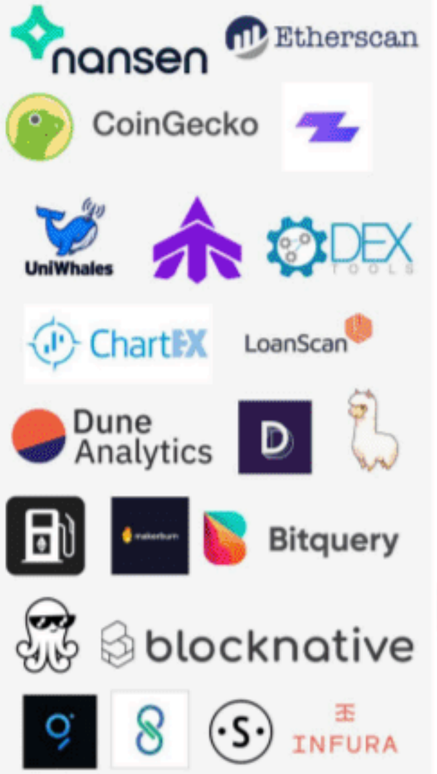
Centralized Exchanges



Active Investors



Data/Analytics



Auditors



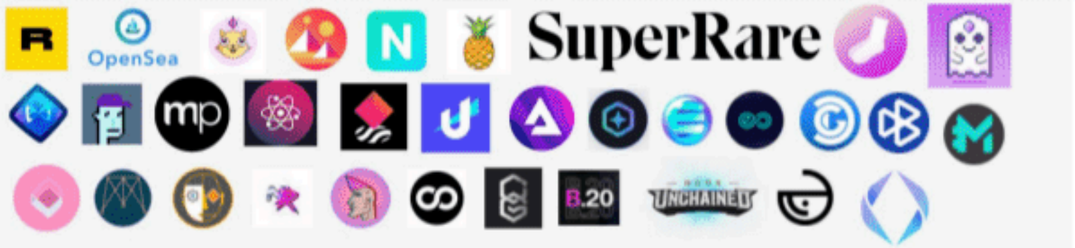
Events



Corporate Testing (per ConsenSys + Forbes)



NFTs



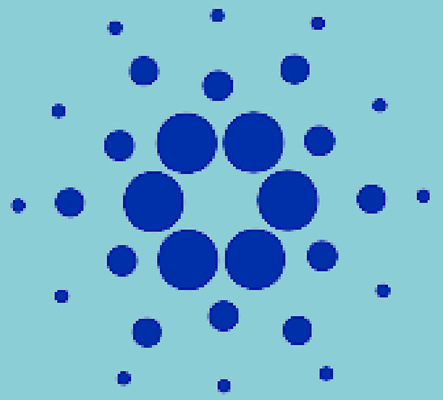
Scaling



Infrastructure



Polkadot



Terza Generazione



ethereum 2.0

I Problemi della seconda generazione

SCALING E PERFORMANCE

SCALABILITÀ

Il problema della scalabilità è uno dei principali. Fondamentalmente, non c'è abbastanza capacità sulla blockchain per gestire in modo efficiente tutte le transazioni

INTEROPERABILITÀ

Le tecnologie di terza generazione vengono sviluppate tenendo in considerazione la necessità di scambiare dati attraverso diversi protocolli blockchain oltre che fonti esterne.

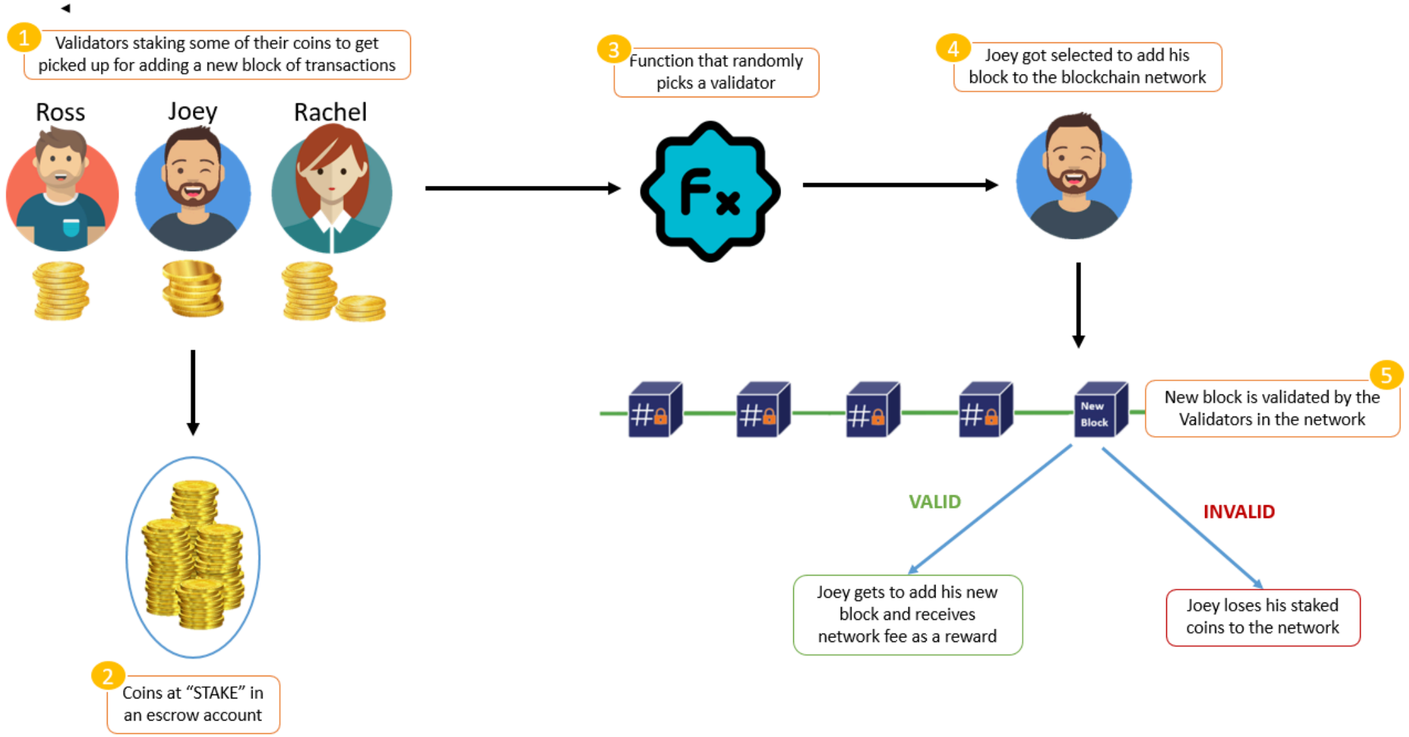
PARALLELISMO

Il parallelismo delle transazioni, che potrebbe aiutare ad aumentare l'efficienza della Blockchain, non funziona bene perché le transazioni e l'ordine in cui si verificano influiscono sullo stato della chain.

Proof of Stake

PoS si basa su validatori che detengono token associati ad una chain di riferimento; *con proof of stake, un validatore viene scelto casualmente, il criterio di scelta varia da blockchain a blockchain, i token fungono da collaterale e quando un partecipante o nodo viene scelto per convalidare una transazione, riceve una ricompensa;* ***il collaterale può essere distrutto in caso di comportamento disonesto.***

Il validatore diventa quindi responsabile del controllo di validità dei nuovi blocchi propagati sulla rete e, occasionalmente, della creazione e della propagazione di nuovi blocchi stessi.



Proof of stake richiede che più validatori concordino sull'accuratezza di una transazione e, una volta che un numero sufficiente di nodi la verifica, la transazione viene eseguita. Proof of stake offre vantaggi chiave rispetto a Proof of Work, la velocità di transazione è più elevata e i requisiti energetici più efficienti consentono blockchain più scalabili il che comporta una maggiore adozione tra i nuovi utenti. Inoltre, proof of stake offre l'opportunità di guadagnare bloccando i propri token in un pool di liquidità e ricevendo commissioni o premi di validazione.

https://ethereum.org/en/developers/docs/consensus-mechanisms/pos/
https://time.com/nextadvisor/investing/cryptocurrency/proof-of-work-vs-proof-of-stake/
https://docs.cardano.org/new-to-cardano/proof-of-stake

Charles Hoskinson



Nel 2014, uno dei programmatori di Ethereum abbandonò il progetto e insieme al suo business partner Jeremy Wood fondò un'impresa chiamata **IOHK** che successivamente iniziò a sviluppare la piattaforma **Cardano**, lanciata il 29 settembre 2017

CARDANO

SCALABILITÀ

"La rete Cardano, oltre a fornire una **larghezza di banda migliorata**, è in grado di elaborare un numero crescente di transazioni al crescere della domanda degli utenti; per i metadati invece, Cardano sta implementando varie tecniche come la **compressione ottimizzata dei dati.**"

INTEROPERABILITÀ

Cardano è stato sviluppato per **supportare trasferimenti cross-chain, diversi tipi di token e di linguaggi per la programmazione di smart contract.**
Cardano promuove inoltre l'interoperabilità verso entità bancarie centralizzate, ritenuta altrettanto importante per garantire legittimità e comodità d'uso.

PARALLELISMO

Cardano sta lavorando per introdurre Hydra, che abiliterà la funzionalità di **side chain** multiple.

https://cardano.org/

Cardano, Ouroboros in sintesi

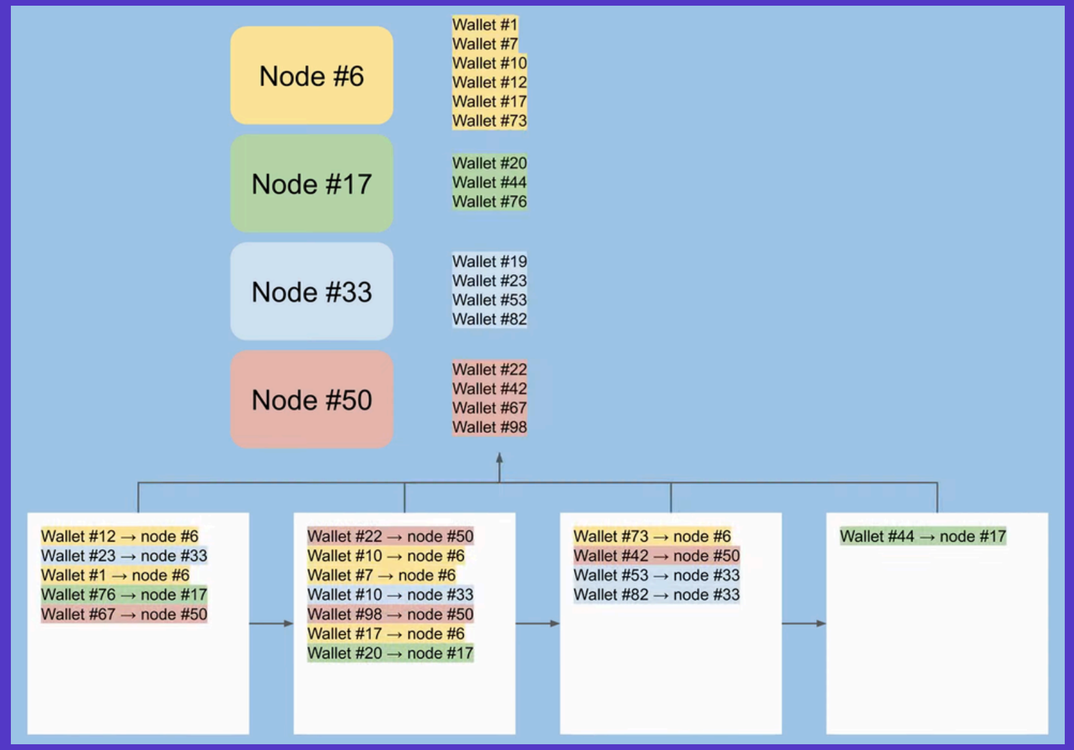
(UNLIMITED) DELEGATED PROOF OF STAKE

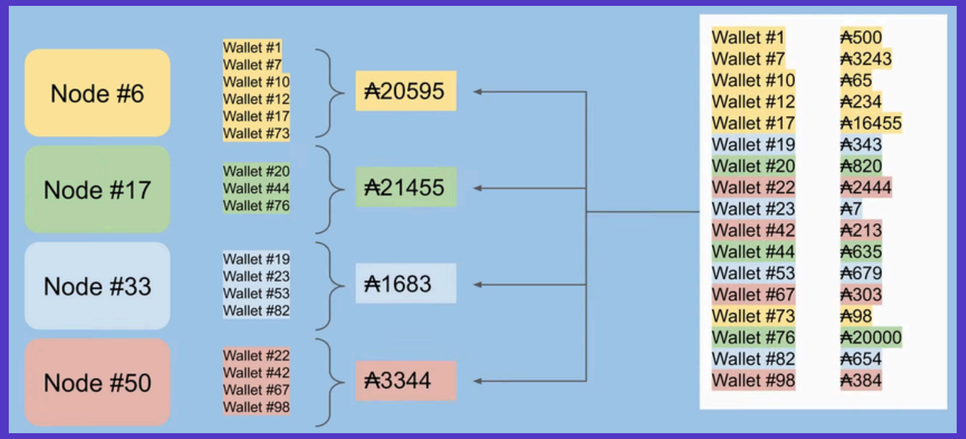
Al centro del protocollo ci sono le **staking pool, nodi affidabili gestiti da un operatore a cui i proprietari di ADA possono delegare i loro token.** I pool di staking vengono utilizzati per garantire che tutti possano partecipare al protocollo, indipendentemente dall'esperienza tecnica o dalla disponibilità a mantenere un nodo in esecuzione. Queste staking pool detengono la quota combinata di varie parti interessate in un'unica entità.

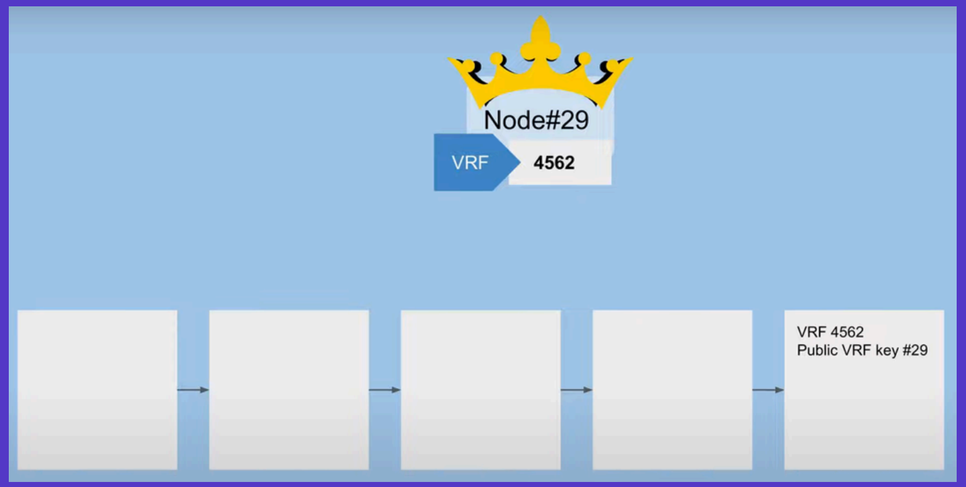
Il creatore di un nuovo blocco viene scelto in base a una combinazione di selezione casuale e quota in staking; *all'interno della catena si verifica un tipo di elezione del leader.*

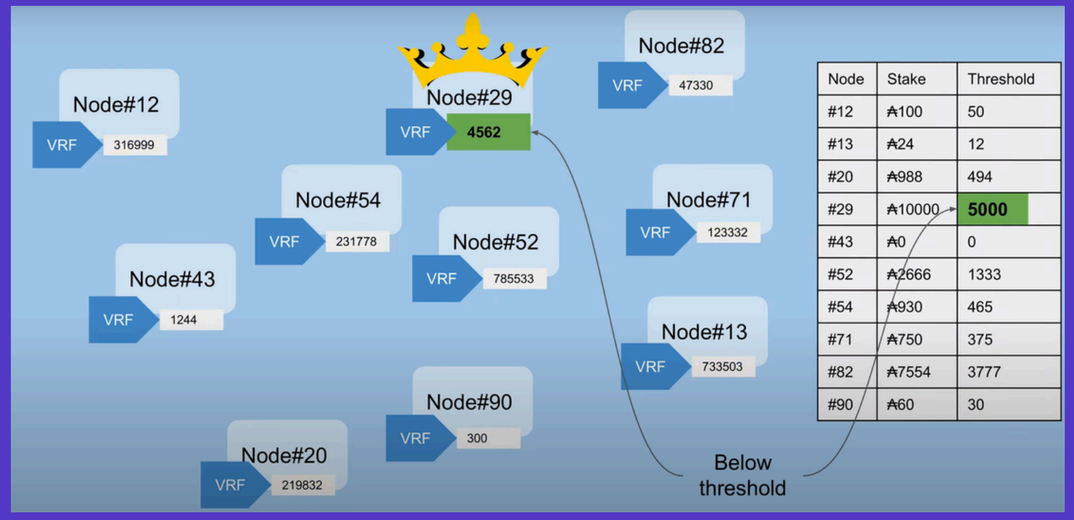
"A fundamental problem for PoS-based blockchain protocols is to simulate the leader election process. In order to achieve a fair randomized election among stakeholders, entropy must be introduced into the system, and mechanisms to introduce entropy may be prone to manipulation by the adversary"

https://iohk.io/en/blog/posts/2022/06/03/from-classic-to-chronos-the-implementations-of-ouroboros-explained/
https://iohk.io/en/research/library/papers/ouroboros-a-provably-secure-proof-of-stake-blockchain-protocol/

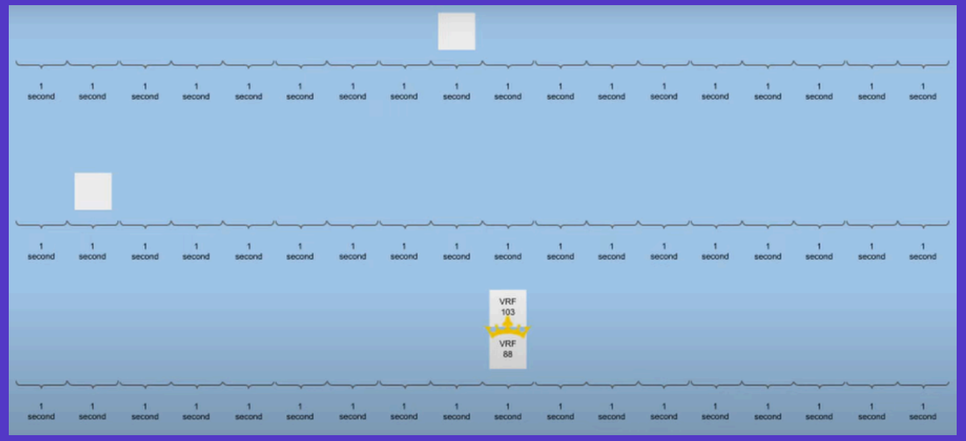


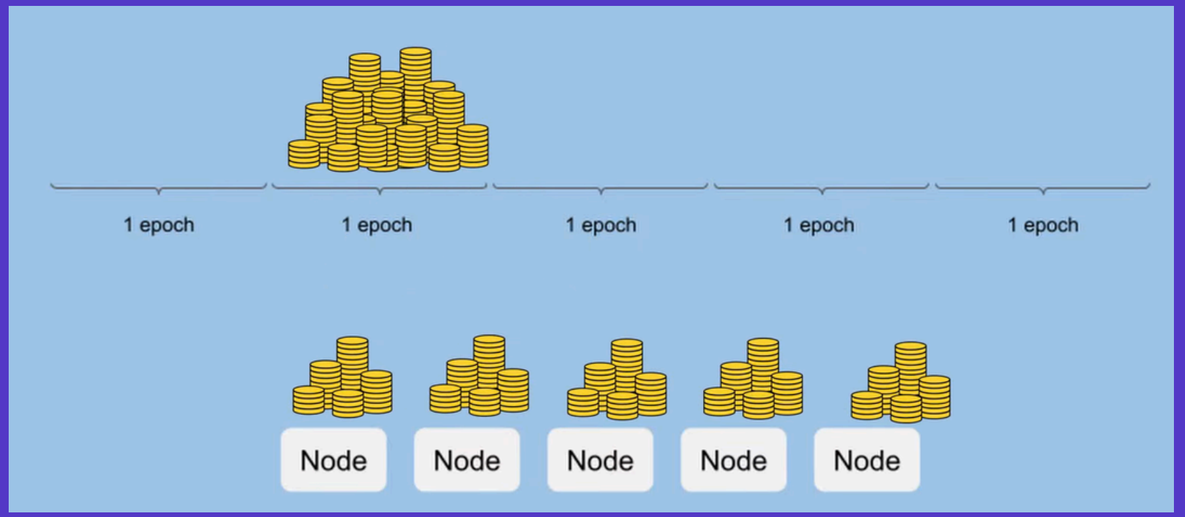


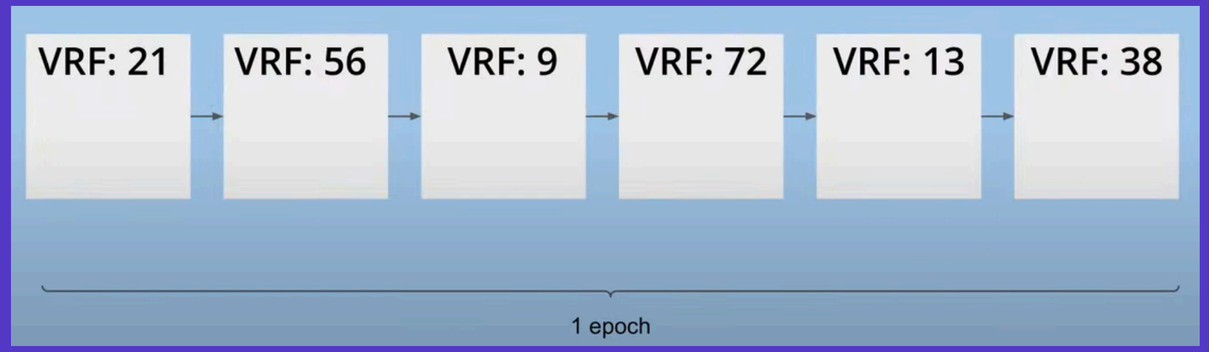


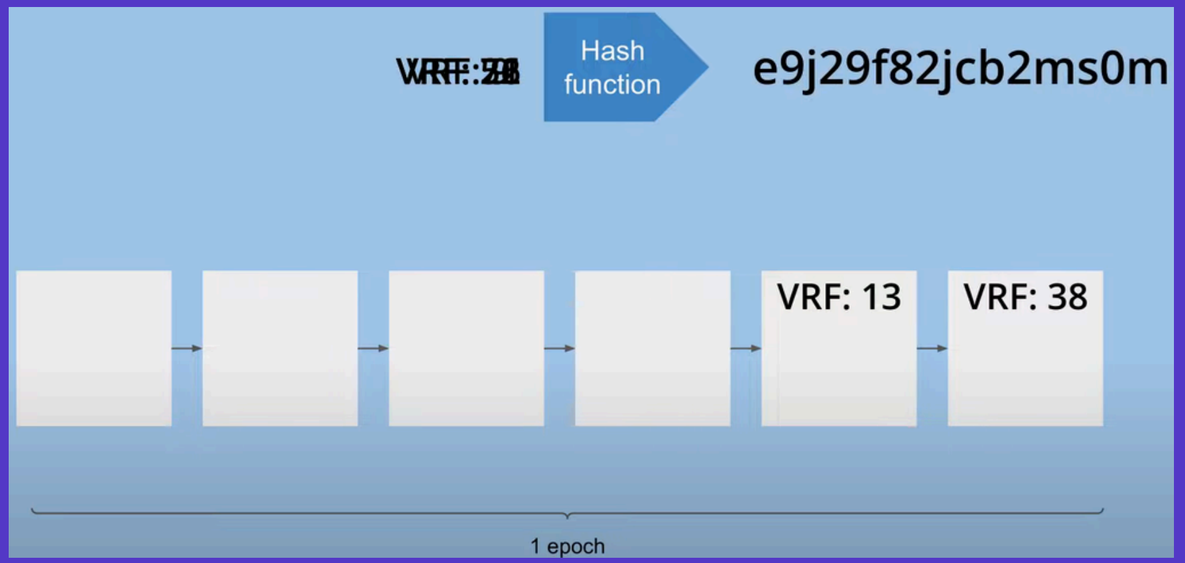


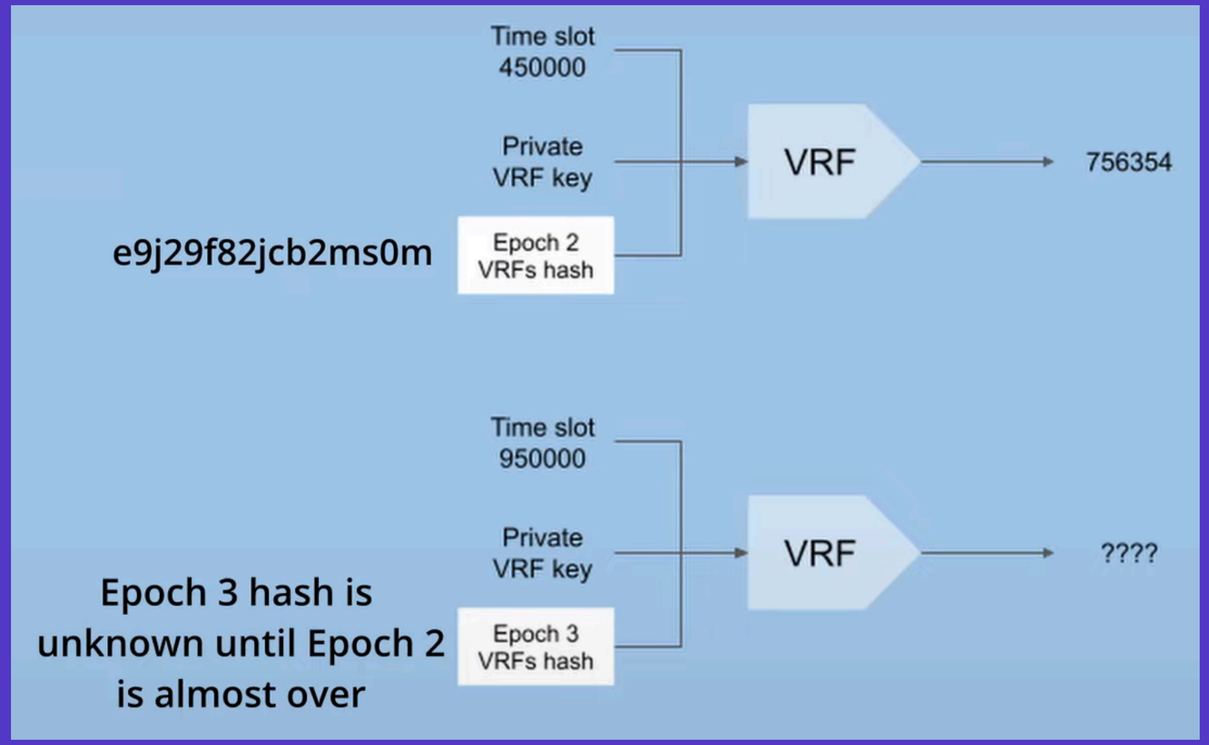
Node	Stake	Threshold
#12	₳100	50
#13	₳24	12
#20	₳988	494
#29	₳10000	5000
#43	₳0	0
#52	₳2666	1333
#54	₳930	465
#71	₳750	375
#82	₳7554	3777
#90	₳60	30




1 epoch
1 epoch
1 epoch
1 epoch
1 epoch
Node
Node
Node
Node
Node


VRF: 21
VRF: 56
VRF: 9
VRF: 72
VRF: 13
VRF: 38
1 epoch


Hash function
e9j29f82jcb2ms0m
VRF: 13
VRF: 38
1 epoch


Time slot 450000
Private VRF key
e9j29f82jcb2ms0m
Epoch 2 VRFs hash
VRF
756354
Time slot 950000
Private VRF key
Epoch 3 VRFs hash
VRF
????
Epoch 3 hash is unknown until Epoch 2 is almost over

Cardano layers:

LAYER 1

- Livello di rete mantiene le connessioni tra tutti i nodi distribuiti nella rete Cardano, ottiene nuovi blocchi dalla rete man mano che vengono prodotti, costruisce blocchi di nuove transazioni e trasmette i blocchi tra i nodi.

- Livello di consenso si occupa dell'esecuzione del protocollo di consenso Ouroboros: come adottare blocchi, come scegliere tra catene concorrenti e quando produrre blocchi propri. Inoltre gestisce le informazioni necessarie per le decisioni di consenso.

- Livello libro mastro specifica lo stato del libro mastro e come deve essere aggiornato per ogni nuovo blocco. Il "livello del libro mastro" è costituito esclusivamente da funzioni che specificano le transizioni tra stati del libro mastro successivi, come derivato dalle regole del modello di contabilità UTxO esteso (EUTxO). Nel complesso, questi tre strati formano la soluzione di livello 1 che è Cardano.



https://iohk.io/en/blog/posts/2022/08/05/layer-1-layer-2-all-you-need-to-know/

Cardano layers:

LAYER 1 SOLUTIONS

- Aumento della dimensione del blocco: più grande è il blocco, più transazioni può contenere. Il primissimo blocco coniato su Cardano aveva una dimensione di 665 byte (0,665 KB). Oggi i blocchi hanno una dimensione di 72 KB.
- Pipelining: migliora i tempi di propagazione dei blocchi sovrapponendo validazione e propagazione, l'obiettivo è che i blocchi vengano propagati ad almeno il 95% dei peer entro cinque secondi riducendo l'overhead di propagazione del blocco.
- Parametri di memoria/CPU per Plutus: miglioramenti alla memoria nella gestione dell'Unspent Transaction Output (UTXO), nella distribuzione delle ricompense di staking, nella gestione dei pool...
- Miglioramenti di nodo: aiutano a distribuire più efficientemente staking e ricompense fornendo così un margine maggiore per l'aumento delle dimensioni di blocco. Inoltre, rendono più efficiente l'uso della memoria.
- Archiviazione su disco: memorizzando parti dello stato del protocollo su disco, i nodi dovranno contenere meno in memoria, quindi i sistemi con vincoli di RAM saranno in grado di eseguire nodi a condizione che dispongano di spazio di archiviazione sufficiente e la memoria non rappresenterà un collo di bottiglia per la scalabilità.

https://iohk.io/en/blog/posts/2022/08/05/layer-1-layer-2-all-you-need-to-know/

Cardano layers:

LAYER 2

Un layer N è un ulteriore protocollo off-chain che funziona sopra la blockchain di livello N - 1. Le parti possono trasferire in modo sicuro token dalla blockchain a un protocollo off-chain, regolare le transazioni in questo protocollo indipendentemente dalla catena sottostante e se necessario ritrasferire in modo sicuro i fondi alla catena sottostante. I protocolli di livello 2 migliorano il throughput e la scalabilità complessivi perché riducono la congestione della rete.

In generale, le soluzioni di livello 2 risolvono il problema di scalabilità inerente alle blockchain di livello 1. Costruite sopra una blockchain esistente, i protocolli di livello 2 eseguono una grande quantità di lavoro di elaborazione che altrimenti avverrebbe sulla catena principale. Un ulteriore vantaggio è che, mentre la soluzione di livello 2 assorbe il carico di lavoro, il livello 1 mantiene la sue proprietà di sicurezza.



https://iohk.io/en/blog/posts/2022/08/05/layer-1-layer-2-all-you-need-to-know/

Cardano layers:

LAYER 2 SOLUTIONS

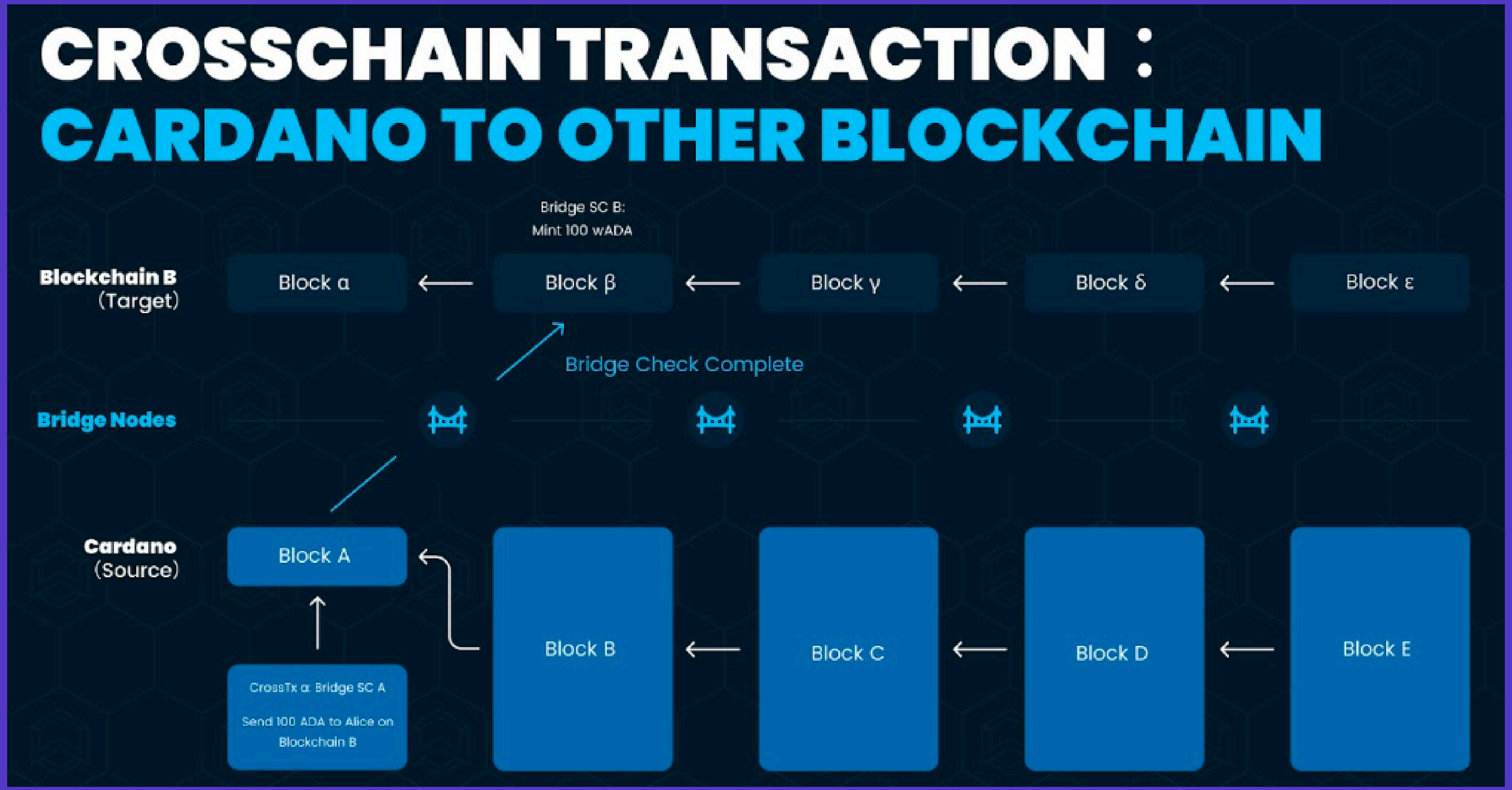
- **Sidechains: Una sidechain, definita come un modo per consentire a più blockchain di comunicare tra loro e fare in modo che una reagisca agli eventi nell'altra, è una blockchain separata collegata a una blockchain principale, nota anche come catena madre, attraverso un meccanismo a due vie (bridge) che consente ai token e ad altre risorse digitali di una catena di essere utilizzati in un'altra e di restituire i risultati alla catena originale.** Le risorse possono essere spostate tra le catene secondo necessità. Una singola catena madre può avere più sidechain interoperabili collegate ad essa, che possono operare in modi completamente diversi.

- Hydra: Hydra Head introduce il concetto di canali di stato isomorfi: ovvero, il riutilizzo della stessa rappresentazione di libro mastro per produrre copie uniformi off-chain, le "Heads". In particolare per Cardano, ciò significa che le risorse native, i token non fungibili (NFT) e gli script Plutus sono disponibili all'interno di ogni Head. L'isomorfismo consente un'estensione naturale del sistema; le Hydra Heads eccellono nel raggiungere una finalità quasi istantanea all'interno di una Head, il processo di creazione e chiusura può richiedere alcuni blocchi, ma una volta stabilite, le transazioni possono fluire rapidamente tra i partecipanti. Poiché le Hydra Head sono isomorfe e utilizzano il modello EUTXO, possono elaborare transazioni non in conflitto contemporaneamente, il che, insieme a una buona rete, consente un utilizzo ottimale delle risorse disponibili.

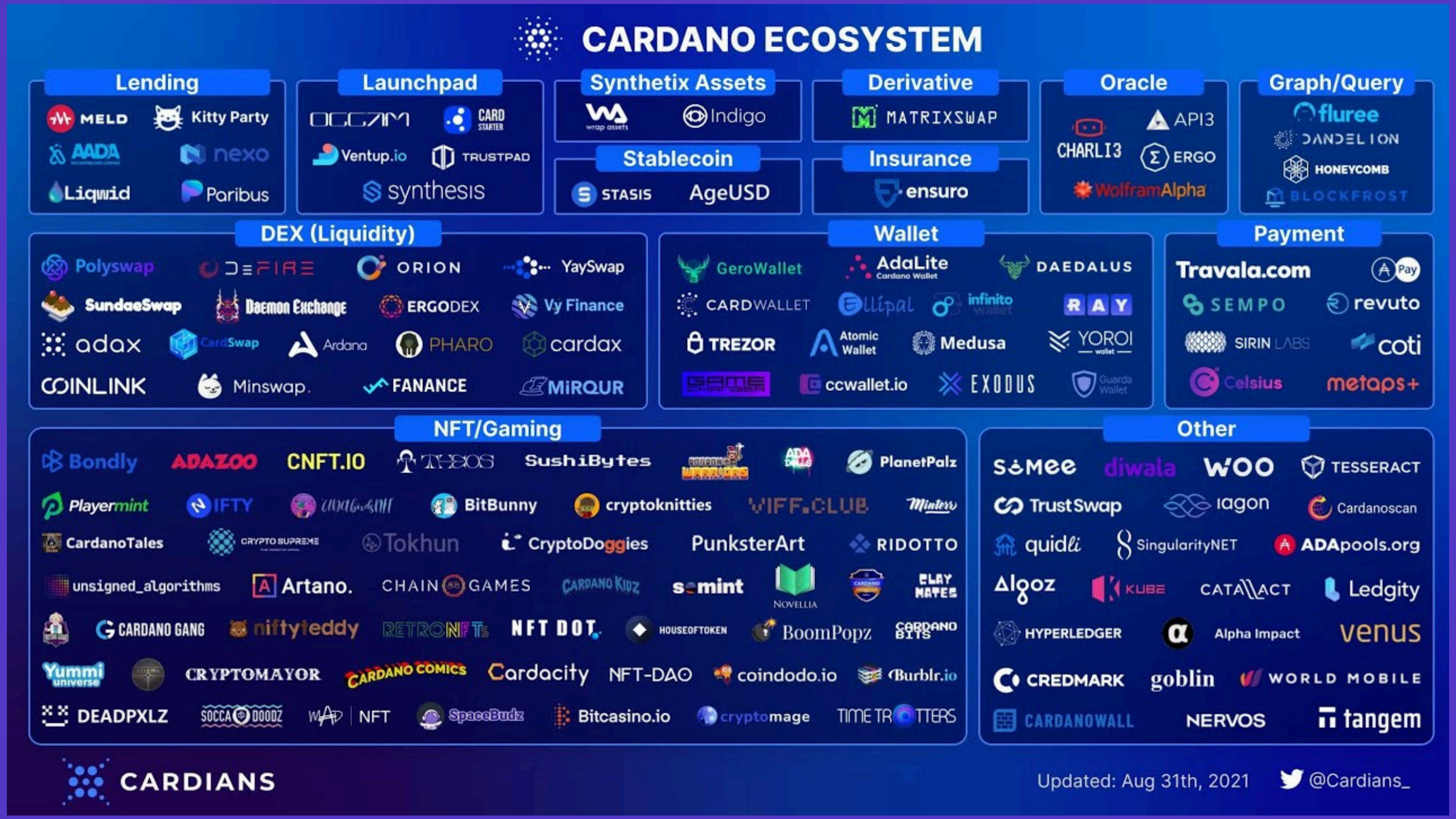
https://iohk.io/en/blog/posts/2022/08/05/layer-1-layer-2-all-you-need-to-know/



https://iohk.io/en/blog/posts/2022/07/08/bridges-and-sidechains-wanchain-making-cardano-interoperable/



https://iohk.io/en/blog/posts/2022/07/08/bridges-and-sidechains-wanchain-making-cardano-interoperable/


CARDANO ECOSYSTEM
Lending
MELD
Kitty Party
AADA
nexo
Liqwid
Paribus
Launchpad
OCCAM
CARD STARTER
Ventup.io
TRUSTPAD
synthesis
Synthetix Assets
wrap assets
Indigo
Stablecoin
STASIS
AgeUSD
Derivative
MATRIXSWAP
Insurance
ensuro
Oracle
API3
CHARLI3
ERGO
WolframAlpha
Graph/Query
fluree
DANDELION
HONEYCOMB
BLOCKFROST
DEX (Liquidity)
Polyswap
DEFIRE
ORION
YaySwap
SundaeSwap
Daemon Exchange
ERGODEX
Vy Finance
adax
CardSwap
Ardana
PHARO
cardax
COINLINK
Minswap.
FANANCE
MiRQUR
Wallet
GeroWallet
AdaLite
Cardano Wallet
DAEDALUS
CARDWALLET
Ellipal
infinito wallet
RAY
TREZOR
Atomic Wallet
Medusa
YOROI wallet
ccwallet.io
EXODUS
Guarda Wallet
Payment
Travala.com
Pay
SEMPO
revuto
SIRIN LABS
coti
Celsius
metaps+
NFT/Gaming
Bondly
ADAZOO
CNFT.IO
THEOS
SushiBytes
PlanetPalz
Playermint
IFTY
BitBunny
cryptoknitties
VIFF.CLUB
Minters
CardanoTales
CRYPTO SUPREME
Tokhun
CryptoDoggies
PunksterArt
RIDOTTO
unsigned_algorithms
Artano.
CHAIN GAMES
CARDANO KIDZ
semint
NOVELLIA
PLAY MATES
CARDANO GANG
niftyteddy
RETRONFTs
NFT DOT.
HOUSEOFTOKEN
BoomPopz
CARDANO BITS
Yummi universe
CRYPTOMAYOR
CARDANO COMICS
Cardacity
NFT-DAO
coindodo.io
Burblr.io
DEADPXLZ
SOCCA DOODZ
NFT
SpaceBudz
Bitcasino.io
cryptomage
TIME TROTTERS
Other
SOMee
diwala
WOO
TESSERACT
TrustSwap
iagon
Cardanoscan
quidli
SingularityNET
ADApools.org
Algoz
KUBE
CATALLACT
Ledgity
HYPERLEDGER
Alpha Impact
venus
CREDMARK
goblin
WORLD MOBILE
CARDANOWALL
NERVOS
tangem
CARDIANS
Updated: Aug 31th, 2021
@Cardians_

Polkadot

Polkadot è un protocollo di rete che consente di trasferire dati arbitrari, non solo token, attraverso blockchain. Ciò significa che Polkadot è un vero ambiente applicativo multi-chain in cui sono possibili cose come registri e calcolo cross-chain. Polkadot permette il trasferimento di questi dati attraverso blockchain di qualsiasi tipo, pubbliche, aperte, con/senza autorizzazione, private, autorizzate...

Polkadot unisce una rete di blockchain eterogenee chiamate parachain e parathread. Queste catene si collegano tra loro e sono protette dalla Relay Chain di Polkadot. Le chain possono anche essere connesse a reti esterne tramite bridge. Polkadot è una blockchain LAYER-0!

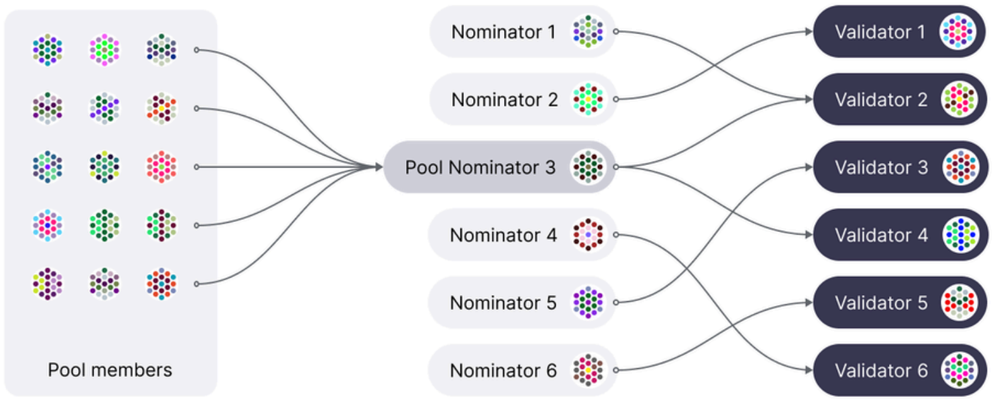


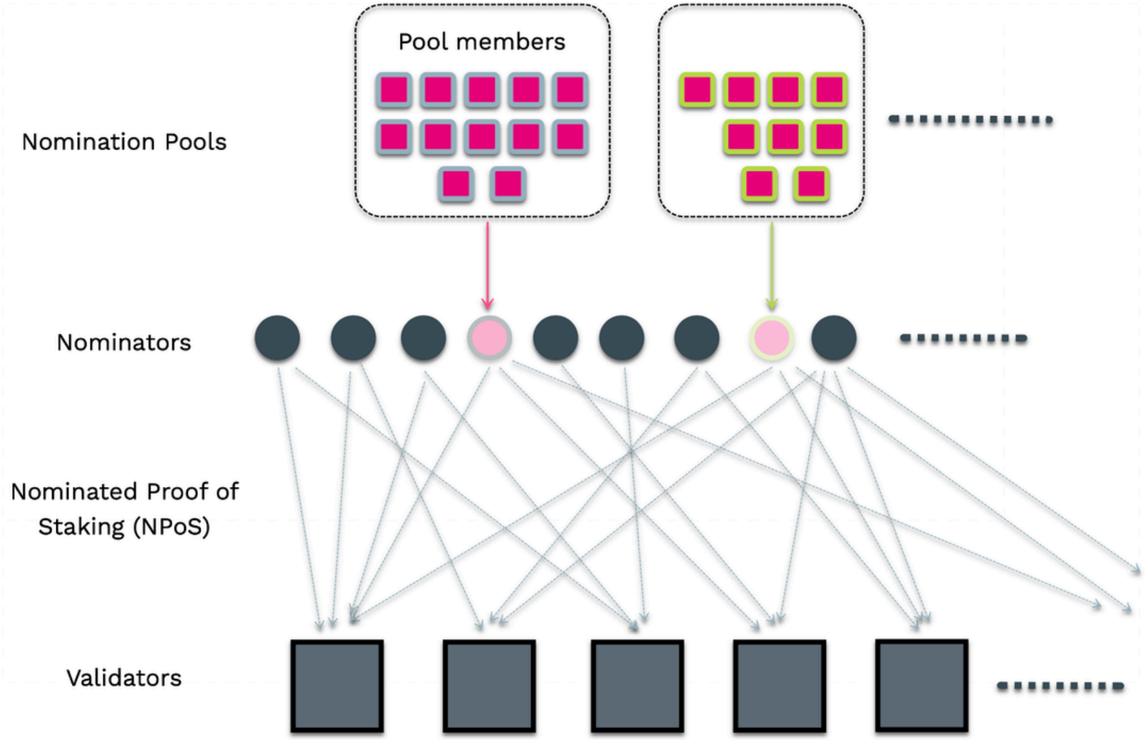
https://polkadot.network/

NOMINATED PROOF OF STAKE

Polkadot utilizza NPoS (Nominated Proof-of-Stake) come meccanismo per selezionare set di validatori. NPos è progettato con i ruoli di validatori e nominatori. Gli attori interessati al mantenimento della rete possono hostare un nodo validatore. I validatori assumono il ruolo di produttori di nuovi blocchi in BABE (Blind Assignment for Blockchain Extension) e di convalidatori di blocchi.

I nominatori possono scegliere di sostenere validatori selezionati con la loro "stake" di token, possono quindi sostenere i candidati di cui si fidano con i loro token.





https://wiki.polkadot.network/docs/learn-consensus

ECOSISTEMA DI POLKADOT



Nominators

Secure the Relay Chain by selecting trustworthy validators and staking DOT.



Validators

Secure the Relay Chain by staking DOT, validating proofs from collators and participating in consensus with other validators.



Collators

Maintain shards by collecting shard transactions from users and producing proofs for validator.





Relay Chain

The heart of Polkadot, responsible for the network's shared security, consensus and cross-chain interoperability.



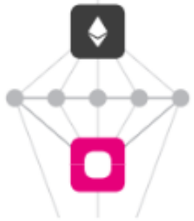
Parachains

Sovereign blockchains that can have their own tokens and optimize their functionality for specific use cases.



Parathreads

Similar to parachains but with a pay-as-you-go model. More economical for blockchains that don't need continuous connectivity to the network.



Bridges

Allow parachains and parathreads to connect and communicate with external networks like Ethereum and Bitcoin.

STANDARDIZZAZIONE BLOCKCHAIN E POLKADOT

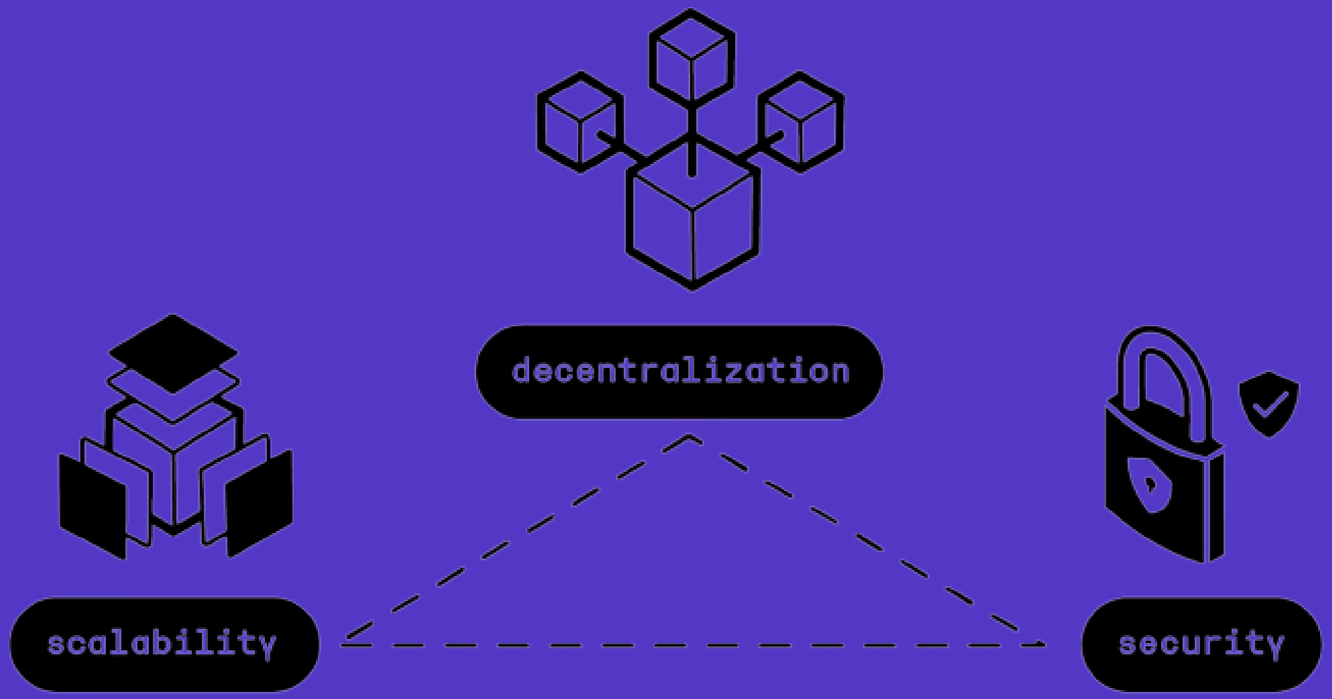
La Relay Chain di Polkadot è costruita con Substrate, un framework per la costruzione di blockchain che è il distillato degli apprendimenti di Parity Technologies nei trascorsi con Ethereum, Bitcoin e varie blockchain aziendali. La macchina a stati di Polkadot è compilata in WebAssembly (Wasm), un ambiente virtuale super performante sviluppato da grandi aziende, tra cui Google, Apple, Microsoft e Mozilla, che hanno creato un ampio ecosistema di supporto allo standard nell'industria. Il networking di Polkadot utilizza libp2p, un framework di rete multipiattaforma flessibile per applicazioni peer-to-peer. Quotato come il futuro standard per le applicazioni decentralizzate, libp2p gestisce la comunicazione nell'ecosistema Polkadot. L'ambiente di runtime di Polkadot viene codificato in Rust, C++ e Golang, rendendo Polkadot accessibile a un'ampia gamma di sviluppatori.



Parachain, elaborazione di transazioni parallele:

Le parachain sono blockchain specializzate che si connettono a Polkadot. Hanno caratteristiche specializzate per i loro casi d'uso e la capacità di controllare la propria governance. Le interazioni sulle parachain vengono elaborate in parallelo, consentendo sistemi altamente scalabili. Le transazioni possono essere distribuite tra le catene, consentendo l'elaborazione di molte più transazioni nello stesso periodo di tempo. (da cui Polkadot è una Blockchain LAYER 0).





Blockchain Trilemma

https://www.ledger.com/academy/what-is-the-blockchain-trilemma

Il Trilemma della Blockchain

La blockchain perfetta vanta tre elementi

SICUREZZA

DECENTRALIZZAZIONE

SCALABILITÀ

Trovare un equilibrio tra le tre è difficile:
Scalabilità e la decentralizzazione sono spesso frenati dalla sicurezza così come la sicurezza tende a essere compromessa da cambiamenti che aumentano la scalabilità. I progetti scelgono quindi, solitamente, di concentrarsi su due delle tre. Idee innovative come lo sharding, le side-chain e gli state-channel sono state implementate per cercare di affrontare il trilemma, ma sono ancora in fase sperimentale.

Quarta generazione? Algorand



Algorand è stata fondata da Silvio Micali, vincitore del premio Turing, del premio Gödel e del premio RSA, leader di fama mondiale nel campo della crittografia e della sicurezza informatica.
Micali e il suo team affermano di avere risolto il trilemma della blockchain inventando un nuovo protocollo di consenso **Pure Proof of Stake (PPoS)**, *il protocollo utilizzato dalla blockchain di Algorand*.

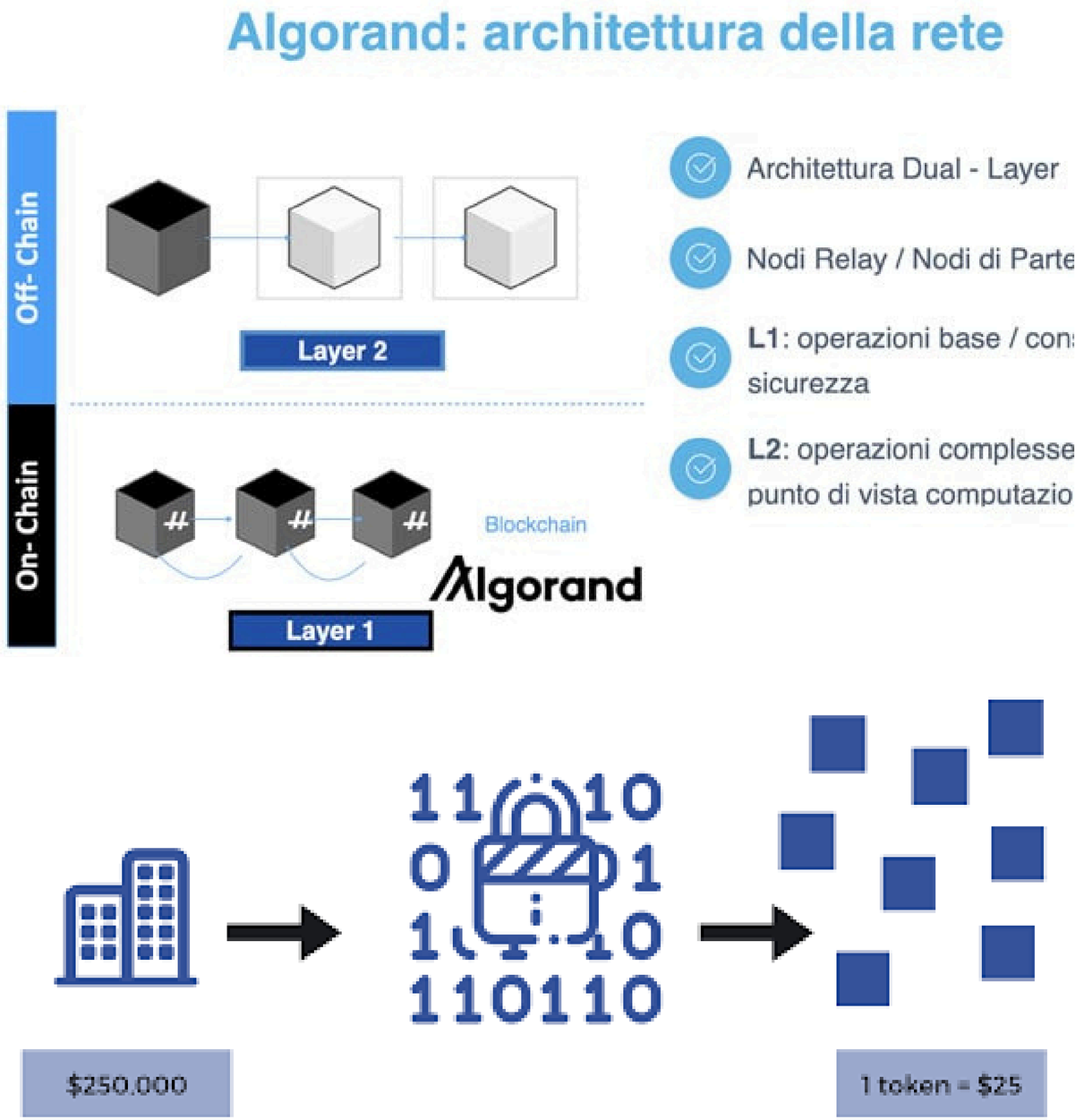
Algorand ha una **struttura blockchain a due livelli unica. Il livello di base supporta contratti intelligenti, creazione di asset e scambi atomici tra asset.**

Sul primo livello della rete Algorand, le piattaforme e gli utenti possono creare ASA che rappresentano token nuovi o esistenti sulla blockchain Algorand., similmente a come funzionano i token ERC-20 sulla rete Ethereum.

Per quanto riguarda la sicurezza, gli smart contracts semplici vengono eseguiti come contratti intelligenti Algorand di livello 1 (ASC1), il che significa che mantengono lo stesso livello di sicurezza del protocollo di consenso stesso.

Il secondo livello di Algorand è riservato agli smart contract più complessi e allo sviluppo di dApp.

Questa biforcazione della rete, tra Layer 1 e Layer 2, consente ad Algorand di elaborare le transazioni in modo molto efficiente, con contratti intelligenti più complessi che si svolgono fuori catena, le transazioni semplici possono essere elaborate più rapidamente sul livello 1 senza essere impantanate da contratti intelligenti più grandi e complessi.



https://www.gemini.com/cryptopedia/what-is-algorand-cryptocurrency-blockchain#section-algorand-protocol-structure

PURE PROOF OF STAKE: PARTECIPAZIONE

Ogni utente online che possiede ALGO può partecipare al protocollo di consenso.

Gli utenti non utilizzano le proprie chiavi di spesa per il consenso ma generano e registrano una chiave di partecipazione; con questa chiave, un account può partecipare alla proposta e al voto di blocchi. L'utilizzo delle chiavi di partecipazione garantisce che gli Algo di un utente siano sicuri anche se il nodo di partecipazione è stato compromesso.

Ogni blocco in Algorand contiene **seed** di selezione casuale e imprevedibile che determina quali utenti dovrebbero partecipare al successivo round del protocollo di consenso.

Quando un nuovo blocco viene integrato nella blockchain, questo seed diviene pubblico e gli utenti controllano segretamente se sono stati selezionati, valutando una funzione casuale verificabile (VRF) con la propria chiave di partecipazione segreta.

Il calcolo VRF produce un output pseudocasuale con una prova crittografica che chiunque può utilizzare per verificare il risultato. Inviando questa prova, un utente può dimostrare a chiunque di essere stato effettivamente selezionato per partecipare al consenso. VRF si comporta in modo simile a una lotteria ponderata; è come se ogni Algo in un account fosse un biglietto della lotteria. Maggiore è il numero di Algo in un account online, maggiori sono le possibilità che l'account venga selezionato per partecipare.

Un avversario non sa chi è responsabile della generazione del blocco successivo fino a quando l'utente selezionato non partecipa al protocollo di consenso, quindi, quando un avversario capisce quale utente è stato selezionato, è troppo tardi per trarne vantaggio, l'utente ha già adempiuto alle proprie responsabilità nel protocollo di consenso.

PURE PROOF OF STAKE: CONSENSO

Il consenso si riferisce al modo in cui i blocchi vengono selezionati e scritti nella blockchain. **Algorand utilizza VRF per selezionare gli account a cui proporre blocchi per un determinato round. Quando un blocco viene proposto alla blockchain, viene selezionato un comitato di elettori per votare sulla proposta di blocco.**

Il consenso richiede tre passaggi per proporre, confermare e scrivere un blocco sulla blockchain:

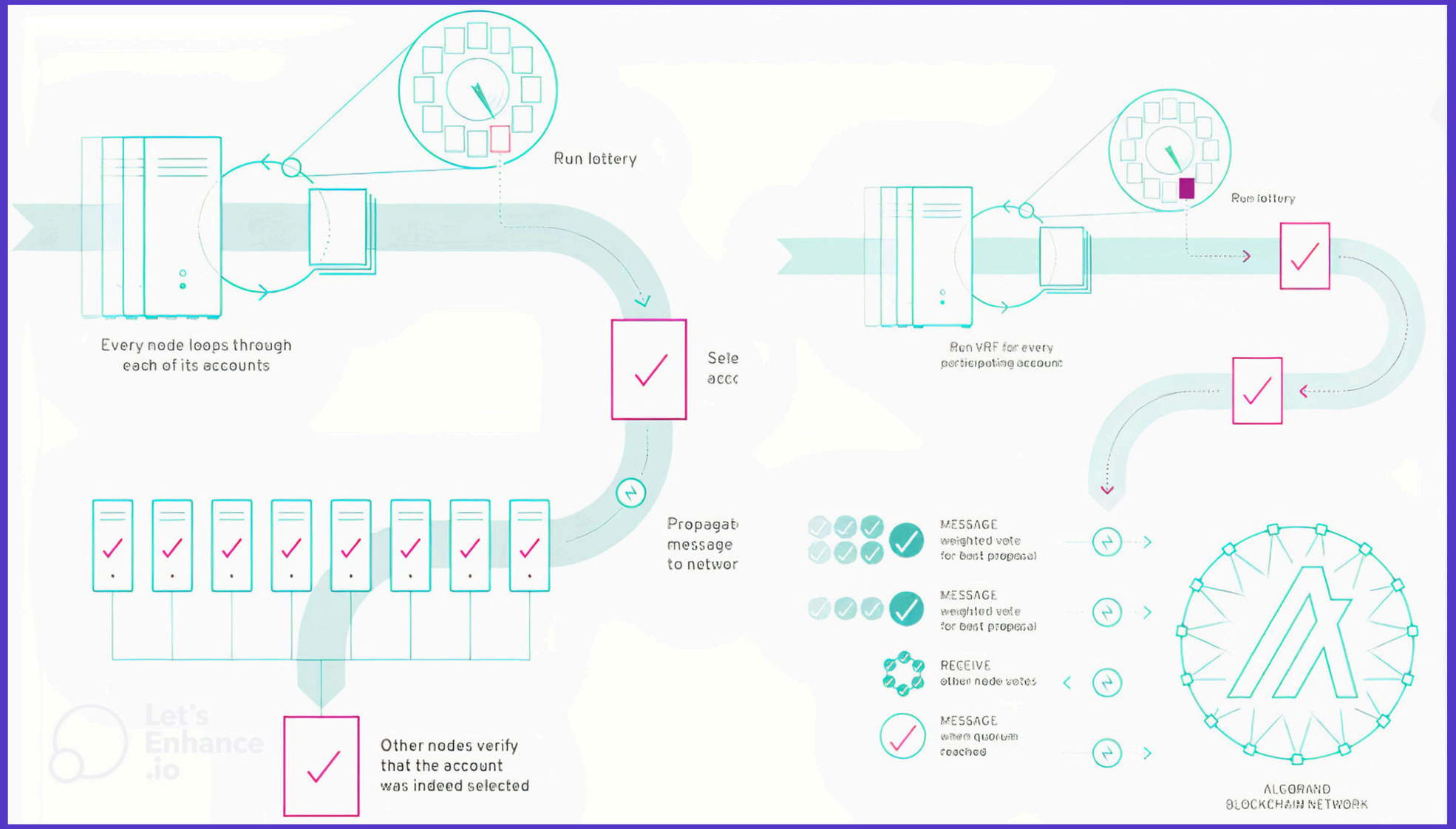
1) proporre, 2) voto morbido e 3) certificazione del voto.

Ognuno è descritto di seguito assumendo il caso ideale in cui non ci sono utenti malintenzionati e la rete non è partizionata (ovvero, nessuna parte della rete è inattiva a causa di problemi tecnici o attacchi DoS).

- **Proposta di blocco:** nella fase di proposta del blocco, gli account vengono selezionati per proporre nuovi blocchi alla rete. *Questa fase inizia con ogni nodo della rete che passa attraverso ciascuno degli account che gestisce e, per ogni account online e partecipante, esegue VRF per determinare se l'account è stato selezionato per proporre il blocco. Una volta selezionato un account, ogni nodo propaga il blocco proposto insieme all'output VRF, il che dimostra che l'account è un proponente valido. Ogni nodo nella rete riceverà proposte di blocco da altri nodi e quindi convaliderà l'output VRF per queste proposte.*

- **Voto morbido:** *Successivamente, ogni nodo eseguirà VRF per ogni account associato per vedere se sono stati scelti per partecipare al comitato di voto. Un account scelto avrà un voto ponderato in base al numero di Algo che possiede. Ogni account scelto filtrerà le proposte fino a una votando per confermarne il blocco. Questi voti saranno per la proposta di blocco con VRF calcolata al timeout più bassa e verranno inviati agli altri nodi insieme alla prova VRF. Ogni nodo convaliderà la prova VRF di appartenenza al comitato prima di aggiungerla al conteggio dei voti. Una volta raggiunto il quorum per il voto soft, il processo passa alla fase di voto di certificazione.*

- **Certifica del voto:** *viene selezionato un nuovo comitato per controllare la proposta di blocco che è stata votata nella fase di voto morbido (eccesso di spesa, doppia spesa o qualsiasi altro problema). Se valida, il comitato vota nuovamente per certificare il blocco. Questo viene fatto in modo simile al voto soft in cui ogni nodo itera attraverso i suoi account gestiti per selezionare un comitato e inviare voti. Questi voti vengono raccolti e convalidati da ciascun nodo fino al raggiungimento del quorum, determinando la fine del round e richiedendo al nodo di creare un certificato per il blocco e di scriverlo nel libro mastro. A quel punto, viene avviato un nuovo round e il processo ricomincia.*



HTTPS://WWW.COINBUREAU.COM/REVIEW/ALGORAND-ALGO/

Monday 02 May 2022, 05:00

FIFA announces partnership with blockchain innovator Algorand



Markets

Algorand Rises After Italy Selects Blockchain Protocol for Digital Guarantees Platform

The new platform is expected to become operational in early 2023.

By Lyllah Ledesma Dec 13, 2022 at 4:01 p.m. Updated Dec 13, 2022 at 6:27 p.m.

Algorand | ABOUT | TECHNOLOGY & RESEARCH | ECOSYSTEM | NEWS & RESOURCES | CONTACT | BUILD ON ALGORAND | THE GREEN BLOCKCHAIN

Planning CBDC with Confidence

Central banks and governments have been thrust into a new reality with new opportunities to foster economic growth and prosperity for their people, preserve monetary value, and maintain governance over their currency. Algorand is armed with technical proficiencies, thought leaders, and capabilities to help central banks around the world plan, test, and adopt blockchain in order to thrive and participate in a new digital economy.

Algorand's 2022 Central Bank Digital Currency (CBDC) Report Recommends a Hybrid Approach to Increase Efficiency and Competitiveness in the Digital Economy

NEWS PROVIDED BY
Algorand →
Jul 12, 2022, 09:50 ET

Integrità

La crittografia protegge dagli attacchi passivi, un requisito diverso è la protezione da attacchi attivi come la falsificazione di dati e transazioni.
L'autenticazione di un messaggio è una procedura che consente alle parti comunicanti di **verificare che i messaggi ricevuti/archiviati siano autentici.**

I due aspetti importanti sono:

- **verificare che il contenuto del messaggio non sia stato alterato.**
- **la fonte sia autentica.**

Si potrebbe anche voler verificare la tempestività di un messaggio e la sequenzialità; tutte queste preoccupazioni rientrano nella categoria dell'integrità dei dati.

La cifratura dei messaggi di per sé non fornisce una forma di autenticazione, tuttavia, è possibile combinare autenticazione e confidenzialità in un singolo algoritmo crittografando un messaggio con il relativo tag di autenticazione.



Funzioni di Hash

INTRODUZIONE

Una funzione di hash accetta in input un messaggio di dimensione variabile M e ne produce un digest di dimensione fissa H(M) come output.

A differenza del MAC, una funzione di hash non necessita di una chiave segreta in input.

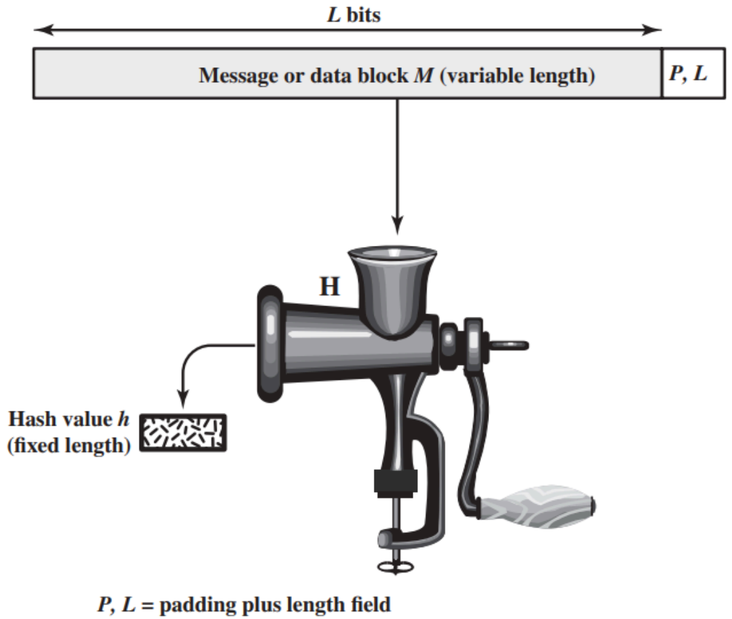


Figure 2.4 **Cryptographic Hash Function; $h = H(M)$**

Il digest del messaggio può essere crittografato utilizzando la crittografia simmetrica; se la chiave segreta non è compromessa, l'autenticità è assicurata.

Il digest del messaggio può anche essere crittografato utilizzando la crittografia a chiave pubblica. L'approccio a chiave pubblica presenta due vantaggi: oltre all'autenticazione del messaggio fornisce un meccanismo di firma digitale e non richiede la distribuzione delle chiavi a soggetti comunicanti.

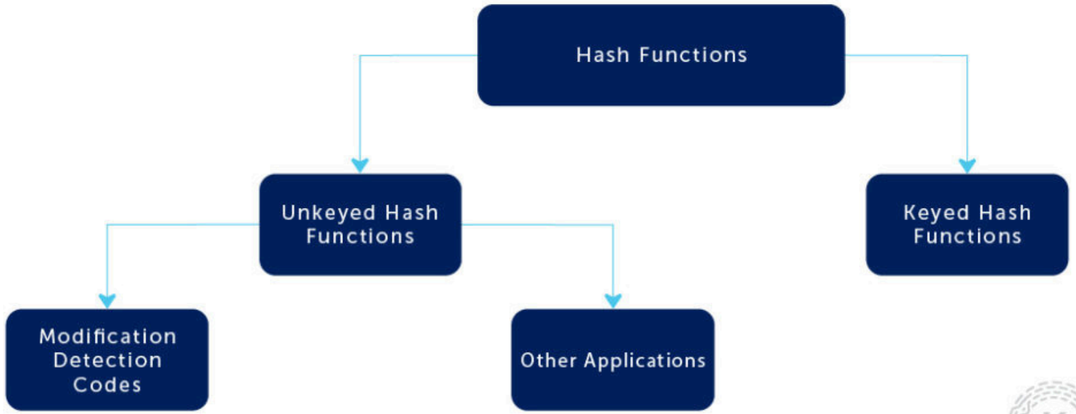
$$h : \{0,1\}^* \rightarrow \{0,1\}^k$$

such that it is computationally infeasible to:

1. given $\mathbf{c} \in \{0,1\}^k$ determine $\mathbf{x} \in \{0,1\}^*$ such that $h(\mathbf{x}) = \mathbf{c}$;
2. given $\mathbf{x} \in \{0,1\}^*$ determine $\mathbf{y} \in \{0,1\}^*$ with $\mathbf{x} \neq \mathbf{y}$ and $h(\mathbf{x}) = h(\mathbf{y})$;
3. determine $\mathbf{x}, \mathbf{y} \in \{0,1\}^*$ such that $h(\mathbf{x}) = h(\mathbf{y})$.

- Provide *digests* of messages to simplify signatures
- Provide a way to construct *robust pointers*
- Provide problems which are computationally expensive to solve

Funzioni di Hash



Common Scenarios - 1

- **Integrity** of files: hash functions may be used to check the integrity of a network-transmitted file through
  - Downloading both the file f and $d{=}h(f)$ from the providing server
  - Computing $h(\cdot)$ on the downloaded file and checking that it matches with d
- **Pseudo-unique** file **indexing**: a unique index for a file f is obtained as $h(f)$ (e.g., video references on youtube, tiny-urls, Dropbox)
  - As collisions do not practically occur, then $h(f)$ is a good unique index for f

Unkeyed Hash Functions

- Consider $h : M \to D$, an unkeyed hash function: the following problems should be (computationally) impossible to solve:

1. **Preimage Problem**: given $d{\in}D$, find $m{\in}M$ s.t. $h(m){=}d$
   - **One-way** property of h: you cannot reconstruct a valid m from d
2. **Second Preimage Problem**: given $m_1{\in}M$, $d{=}h(m_1)$, find $m_2{\in}M$ s.t. $m_1{\neq}m_2$, $h(m_2){=}h(m_1){=}d$
   - **Weak Collision Resistance** property of h: you cannot find a message hashing to the same digest of an already known message
3. **Collision Problem**: find $m_1, m_2{\in}M$ s.t. $m_1{\neq}m_2$, $h(m_1){=}h(m_2)$
   - **(Strong) Collision Resistance** property of h: you cannot find two arbitrary messages with the same digest

Common Scenarios - 2

- **Efficient digital signatures**: instead of signing the file f, sign $h(f)$
- **Safe password storage**: instead of storing a password p, store $d = h(p)$. Even if an attacker retrieves d, he cannot find p
  - having a very fast hash function may be an issue: it allows to quickly test a set of guesses for p extracted from a dictionary
- **Commitment Schemes**. Alice wants to commit to the choice of a value v, without revealing it instantly. She must be able to prove that she committed to that value
- Example: Alice and Bob want to perform a coin toss via e-mail:
  - Alice bets on a value, say, "heads", and sends to Bob $h_a{=}h(\mathtt{cat}(\text{“heads”}, r_a))$
  - Bob bets on a value say, "tails", sends to Alice $h_b{=}h(\mathtt{cat}(\text{“tails”}, r_b))$
  - The coin is tossed, and then Alice and Bob broadcast the values of r_a, r_b and the values (h_a, h_b) they committed to, allowing the commitment check

Funzioni di Hash

Practical 1st Preimage Problem - (black box analysis)

- A real hash function $h(\cdot)$ does not have an infinite digest space
- Trying to find a preimage for $\overline{d} \in D$ calling the hash function with a random input, I will obtain the preimage with probability $\frac{1}{|D|}$
- Probability of calling $h(\cdot)$ q times without success: $\left(1 - \frac{1}{|D|}\right)^q$
- Probability of getting **at least** one valid preimage, m_i, for d with q calls is:

$$\Pr(m_i \text{ s.t. } d = h(m_i)) = 1 - \left(1 - \frac{1}{|D|}\right)^q$$

Practical 2nd Preimage Problem - (black box analysis)

- Let $m \in M$, $d = h(m)$ be the msg-digest pair for which we want a 2nd preimage (i.e., find $m' \in M$ s.t. $m' \neq m$, $h(m') = h(m) = d$)
- Pick q msg.s at random: $m, m_1, m_2, \ldots m_{q-1}$ s.t. $m \neq m_i$, $i \in \{1, \ldots, q-1\}$
  - For each chosen message m_i, compute $h(m_i)$
  - If one of the $h(m_i)$ is d, return m_i; otherwise fail
- Again, the probability of getting **at least** one valid preimage of d is:

$$\Pr(\forall\, i\ m_i \neq m,\ h(m_i) = h(m)) = 1 - \left(1 - \frac{1}{|D|}\right)^{q-1} \approx \frac{q-1}{|D|}$$

Practical Collision Problem - (black box analysis)

- The approach to find a collision in a hash function is
  - Pick q messages at random m_i, $i \in \{1, \ldots, q\}$
  - If any two digests, $h(m_i)$ and $h(m_j)$ with $i \neq j$ $(i, j \in \{1, \ldots, q\})$, are equal then return (m_i, m_j); otherwise, fail

- $q = 1$, $\Pr(\text{no collision}) = 1$
- $q = 2$, $\Pr(\text{no collision}) = 1 \cdot \left(1 - \frac{1}{|D|}\right)$
- $q = 3$, $\Pr(\text{no collision}) = 1 \cdot \left(1 - \frac{1}{|D|}\right) \cdot \left(1 - \frac{2}{|D|}\right)$
- $\ldots$
- q picks, $\Pr(\text{no collision}) = 1 \cdot \left(1 - \frac{1}{|D|}\right) \cdot \left(1 - \frac{2}{|D|}\right) \cdots \left(1 - \frac{q-1}{|D|}\right)$

Finding the number of trials, q needed to get a collision

- From basic calculus (Taylor series), recall that $1 - \frac{a}{x} \leq e^{-\frac{a}{x}}$
- Consequentially, we can rewrite the previous result as follows:

$$\Pr(\text{no coll.}) = \prod_{i=0}^{q} \left(1 - \frac{i}{|D|}\right) \leq \prod_{i=0}^{q-1} e^{-\frac{i}{|D|}} = e^{-\frac{1}{|D|}\sum_{i=0}^{q-1} i} = e^{-\frac{q(q-1)}{2|D|}}$$

- We look for: $\Pr(\textbf{no coll.}) \geq \textbf{50\%} \Leftrightarrow \Pr(\textbf{at least one coll.}) \leq \textbf{50\%}$
- we find that: $e^{-\frac{q(q-1)}{2|D|}} \geq \frac{1}{2} \Rightarrow \frac{q(q-1)}{2|D|} \leq \ln 2$
- Solving for q we conclude that $\Pr(\text{no collision}) \geq \frac{1}{2}$ if $0 < q \leq \frac{1}{2} + \sqrt{\frac{1}{4} + 2|D| \ln 2} \Leftrightarrow$ (roughly) $\mathbf{0 < q \leq 1.1774\sqrt{|D|}}$

Summing up

- To obtain a sound cryptographic hash function we need to find an efficiently computable map $h : M \to D$ such that:
  - It is computationally unfeasible to find a **first/second preimage** in **"less than"** $|D|$ calls to the hash function
  - It is computationally unfeasible to find a **collision** in **less than** $1.17\sqrt{|D|}$ calls to the hash function ($\approx \sqrt{|D|} = \sqrt{2^{\log_2 |D|}} = 2^{\frac{1}{2}\log_2 |D|}$ calls)

How to design an hash function?

- The design of an infinite domain (unlimited message length) hash function is done in two phases:
  1. designing a finite domain **compression function**
  2. designing a scheme to combine the compression function in order to act on an infinite message space
- Willing to formalize the problem:
  - for simplicity, we consider bit strings as inputs/outputs
  - given a **compression function** $h : M \mapsto D$, each msg is encoded by $l + t$ bits, while each digest is encoded using only l bits: $h : \{0,1\}^{l+t} \mapsto \{0,1\}^{l}$
  - we want to construct a hash function with unlimited domain $\widetilde{h} : \{0,1\}^{*} \mapsto \{0,1\}^{s}$, for some integer s (digest size in bits)

Digesting the blocks

- Once we have the block sequence $\overline{m}_1, \overline{m}_2, \ldots, \overline{m}_r$, we need to devise a scheme to feed them into $h(\cdot, \cdot)$
- Let IV be some public initial value encoded with l-bit, then compute

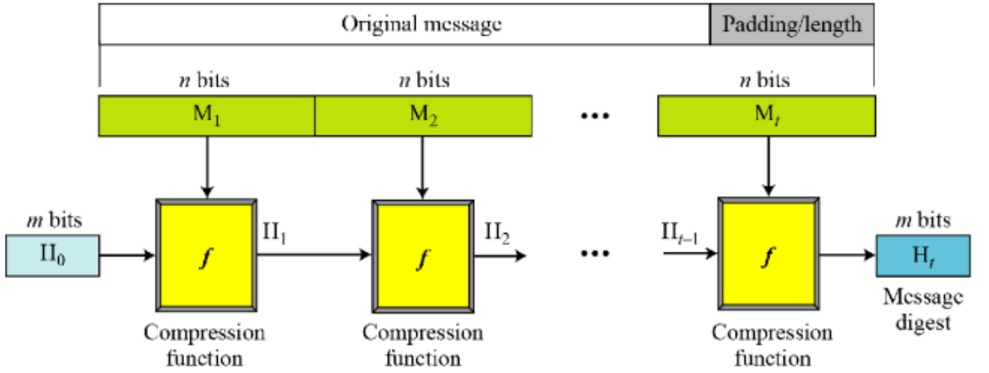
$$z_0 \leftarrow \mathrm{IV}$$
$$z_i \leftarrow h\left(z_{i-1}, \bar{m}_i\right), \ 1 \leq i \leq r$$

  where the first l-bit parameter of $\mathrm{h}(\cdot, \cdot)$ is called *inner state*, while the second t-bit parameter is the next *message chunk*
- After the last sub-string, $\bar{m}_r$, has been processed, a further output transformation $g : \{0,1\}^{l} \rightarrow \{0,1\}^{s}$ can be applied (usually, it's the identity map, provided $l{=}s$)

From an unlimited string to a sequence of blocks

- We want to compute $\widetilde{h} : \{0,1\}^{*} \mapsto \{0,1\}^{s}$ with a sequence of applications of $h : \{0,1\}^{l+t} \mapsto \{0,1\}^{l}$
- The input message should thus be preprocessed so that the length is a multiple of t
- Use an injective padding function to obtain an input string with length multiple of t (e.g.: concatenate enough zeros, or the size of the message followed by enough zeros)
- Split the padded input string $\overline{m}$ into substrings with t bits each: $\overline{m}_1, \overline{m}_2, \ldots, \overline{m}_r$

Merkle-Damgård Construction



- The vast majority of currently used hash functions are based on this construction
- The most common attacks against this structure rely on appending a *chosen block* to the original message to obtain a collision

- It is possible to employ a common block cipher in place of the ad-hoc designed compression function h
- To this end, the message should be padded to be a multiple of the length of the cipher block, and split accordingly in a sequence of blocks $x_1, \ldots, x_r$
- The proposed schemes maintain an inner state which is initialized to a constant known value $H_0 = \mathrm{IV}$
- The update rule of the inner state takes a block of the padded message x_i and computes $H_i = f(H_{i-1}, x_i)$
- The final digest of the message is: H_{r+1}.

Note: there are several ways to use the chosen block cipher primitive as a compression function f

DM (Davies-Meyer) Scheme

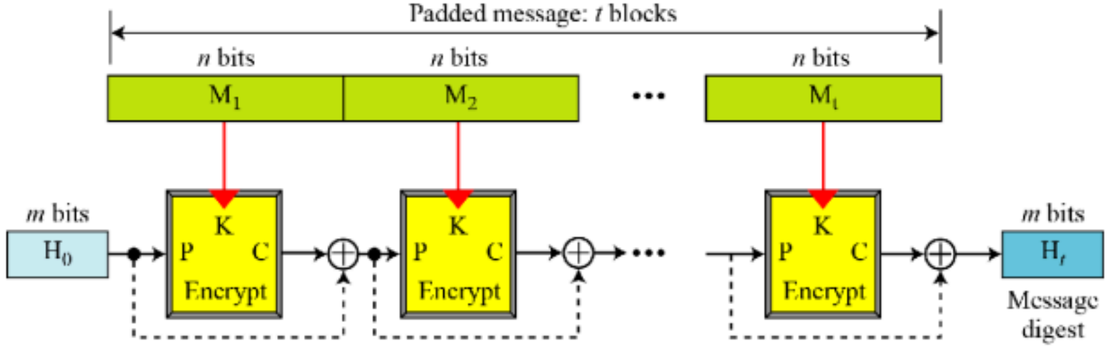


Figure: $f(x_i, H_{i-1}) = E_{x_i}(H_{i-1}) \oplus H_{i-1}$

- Solves the issue of generating collision via appending a block

MMO (Matyas-Meyer-Oseas) Scheme

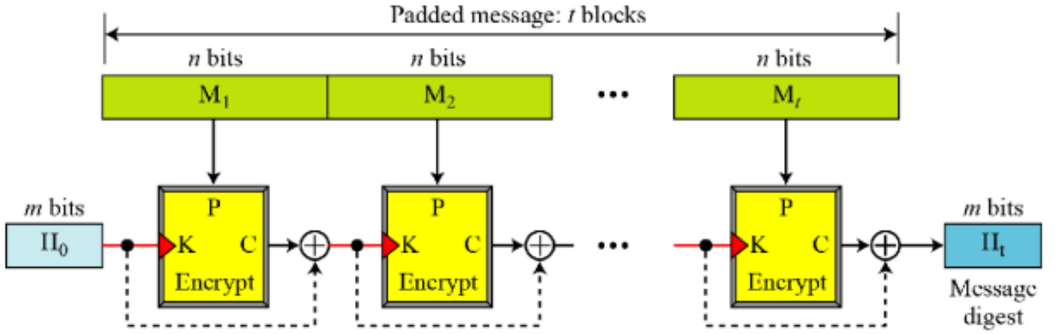


Figure: $f(x_i, H_{i-1}) = E_{H_{i-1}}(x_i) \oplus H_{i-1}$

- Avoids using the message as the key of the block cipher, uses the previous state as key

MP (Miyaguchi-Preneel) Scheme

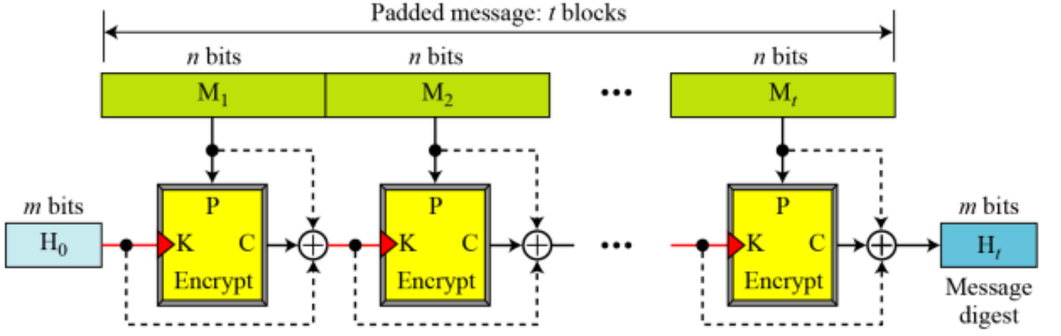


Figure: $f(x_i, H_{i-1}) = E_{H_{i-1}}(x_i) \oplus x_i \oplus H_{i-1}$

- Also adds the message to the output of the single compression function

Keys for Comparison	MD5	SHA-1	SHA-2 (224 & 256/384 & 512)	SHA-3 (224/256/384/512)
Available Since	1992	1995	2002	2008
Block Size	512 bits	512 bits	512/1024 bits	1152/1088/8 32/576 bits (this is referred to as a Rate [R] for SHA-3 algorithms)
Hash Digest Size (Output)	128 bits (i.e., 16 bytes), or 32 hexadecimal digits	160 bits (i.e., 20 bytes), or 40 hexadecimal digits	256 bits (i.e., 32 bytes), or 64 hexadecimal digits/512 bits (i.e., 64 bytes), or 128 hexadecimal digits	224/256/384/512 bits (i.e., 28/32/48/64 bytes), or 56/64/96/128 hexadecimal digits
Rounds of Operations	64	80 (4 groups of 20 rounds)	64 (for SHA-224 and SHA-256)/80 (SHA0384/SHA-512)	24
Construction	Merkle–Damgård	Merkle–Damgård	Merkle–Damgård	Sponge (Keccak)
Collision Level	High — They can be found in seconds, even using an ordinary home computer.	Cheap and easy to find as demonstrated by a 2019 study.	Low — No known collisions found to date.	Low
Successful Attacks	Many. Researchers showed concrete evidence in 2004.	Yes, many. The first one called SHAttered happened in 2017.	SHA-256 has never been broken.	Few collision type attacks have been demonstrated.

Security Level	Low	Low	High	High
ApplicationsApplications	Previously used for data encryption, MD5 is now mostly used for verifying the integrity of files against involuntary corruption.	Previously widely used in TLS and SSL. Still used for HMAC (even if it's recommended to move to a more secure algorithm), and for verifying the integrity of files against involuntary corruption.	Widely used in: ▪ Security applications and protocols (e.g., TLS, SSL, PGP, SSH, S/MIME, Ipsec) ▪ Cryptocurrencies transactions validation ▪ Digital certificates ▪ Other applications	Used to replace SHA-2 when necessary (in specific circumstances).
Deprecated?	Yes	Yes	No	No

Algorithm and variant		Output size (bits)	Internal state size (bits)	Block size (bits)	Rounds	Operations	Security against collision attacks (bits)	Security against length extension attacks (bits)	Performance on Skylake (median cpb)[1] Long messages	Performance on Skylake (median cpb)[1] 8 bytes	First published
MD5 (as reference)		128	128 (4 × 32)	512	64	And, Xor, Or, Rot, Add (mod 2^{32})	≤ 18 (collisions found)[2]	0	4.99	55.00	1992
SHA-0		160	160 (5 × 32)	512	80	And, Xor, Or, Rot, Add (mod 2^{32})	< 34 (collisions found)	0	≈ SHA-1	≈ SHA-1	1993
SHA-1							< 63 (collisions found)[3]		3.47	52.00	1995
SHA-2	*SHA-224*	224	256 (8 × 32)	512	64	And, Xor, Or, Rot, Shr, Add (mod 2^{32})	112	32	7.62	84.50	2004
	SHA-256	256					128	0	7.63	85.25	2001
	SHA-384	384	512 (8 × 64)	1024	80	And, Xor, Or, Rot, Shr, Add (mod 2^{64})	192	128 (≤ 384)	5.12	135.75	2001
	SHA-512	512					256	0[4]	5.06	135.50	2001
	SHA-512/224	224					112	288	≈ SHA-384	≈ SHA-384	2012
	SHA-512/256	256					128	256			
SHA-3	*SHA3-224*	224	1600 (5 × 5 × 64)	1152	24[5]	And, Xor, Rot, Not	112	448	8.12	154.25	2015
	SHA3-256	256		1088			128	512	8.59	155.50	
	SHA3-384	384		832			192	768	11.06	164.00	
	SHA3-512	512		576			256	1024	15.88	164.00	
	SHAKE128	*d* (arbitrary)		1344			min(*d*/2, 128)	256	7.08	155.25	
	SHAKE256	*d* (arbitrary)		1088			min(*d*/2, 256)	512	8.59	155.50	

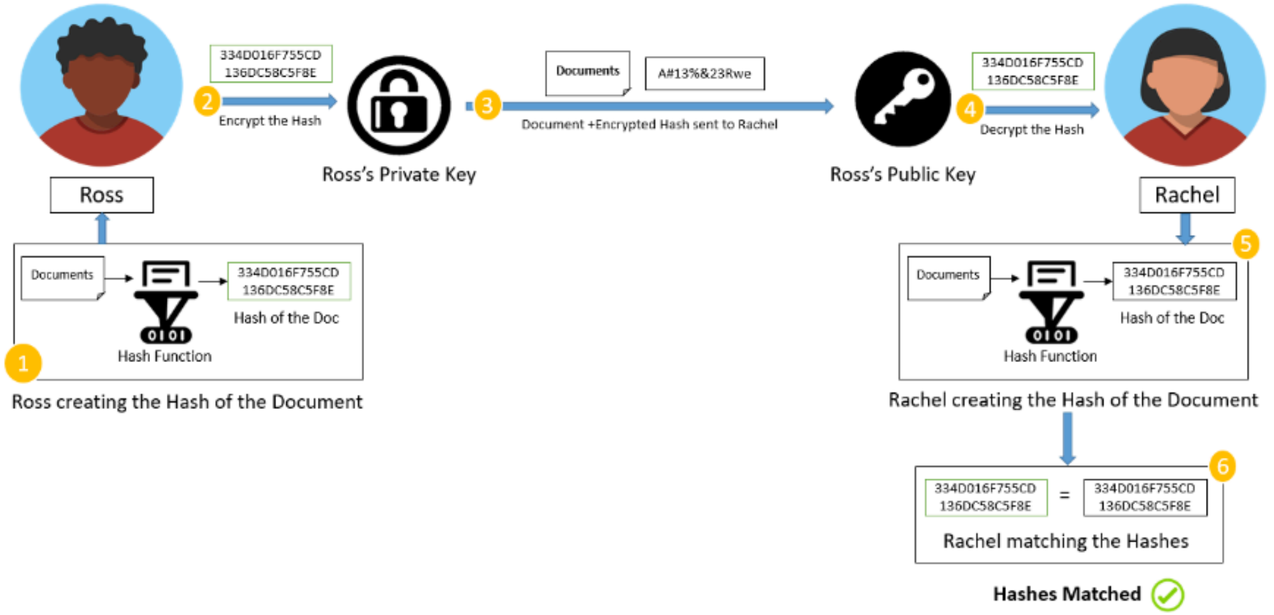
https://codesigningstore.com/hash-algorithm-comparison

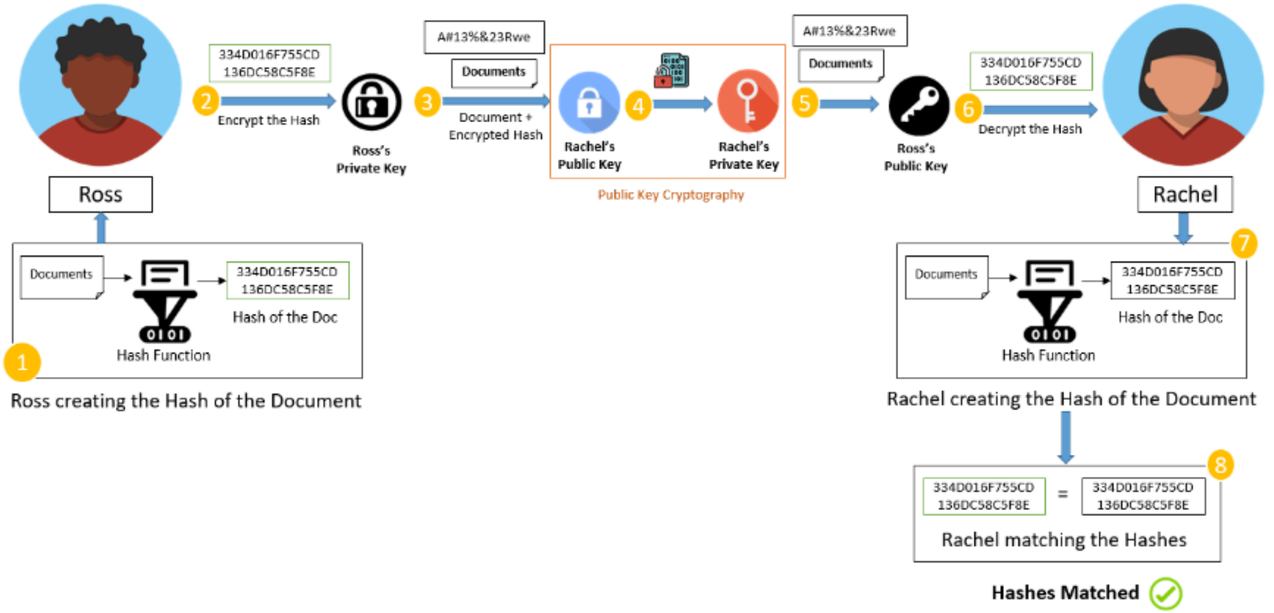
Firma Digitale

- Non Ripudiabilità
- Non Falsificabilità
- Integrità

- Autenticazione

- Confidenzialità





Digital Signature

- Informally: electronic equivalent of a signature; can be used to authorize or authenticate operations.
- Formally: three algorithms $\texttt{Gen}$, $\texttt{Sign}$, $\texttt{Verify}$ such that
  1. $\texttt{Gen}$: takes as input a security parameter k and returns a pair of keys $(sk, pk) \leftarrow \texttt{Gen}(k)$.
  2. $\texttt{Sign}$: takes as input a security parameter k, the secret key sk and a message $m \in M_k$ and outputs a signature $\sigma \leftarrow \texttt{Sign}_{sk}(m)$
  3. $\texttt{Verify}$: takes as input a public key pk, a message m and a purpoted signature σ and returns a bit $b = \texttt{Verify}_{pk}(m, \sigma)$.
  4. $\forall (sk, pk) \leftarrow \texttt{Gen}(k), \forall m \in M_k, \forall \sigma \leftarrow \texttt{Sign}_{sk}(m)$,

$$\texttt{Verify}_{pk}(m, \sigma) = 1.$$

- sk is the secret key and it is private
- pk is the public key and it is public
- only the owner of sk can sign a message m; so a correct signature on m attests that the owner of sk has seen/read/endorsed m
- everybody can verify the signature on a message

We get:

1. Authentication
2. Non-repudiation

A signature is

- Existentially unforgeable if for all probabilistic polynomial time adversaries A the following is negligible

$$\varepsilon_A(k) = \Pr\left[\begin{array}{ll} \{m_i\}_{i=1}^{\ell} \leftarrow M_k; (pk, sk) \leftarrow \texttt{Gen}(k) & \texttt{Verify}_{pk}(m, \sigma) = 1 \\ \forall i \in \{1, \ldots, \ell\} : \sigma_i \leftarrow \texttt{Sign}_{sk}(m_i) \ : & \text{and} \\ (m, \sigma) \leftarrow A(pk, \{m_i, \sigma_i\}_{i=1}^{\ell}) & m \notin \{m_i\}_{i=1}^{\ell} \end{array}\right]$$

- Strongly unforgeable if for all probabilistic polynomial time adversaries A the following is negligible

$$\varepsilon_A(k) = \Pr\left[\begin{array}{ll} \{m_i\}_{i=1}^{\ell} \leftarrow M_k; (pk, sk) \leftarrow \texttt{Gen}(k) & \texttt{Verify}_{pk}(m, \sigma) = 1 \\ \forall i \in \{1, \ldots, \ell\} : \sigma_i \leftarrow \texttt{Sign}_{sk}(m_i) \ : & \text{and} \\ (m, \sigma) \leftarrow A(pk, \{m_i, \sigma_i\}_{i=1}^{\ell}) & (m, \sigma) \notin \{(m_i, \sigma_i)\}_{i=1}^{\ell} \end{array}\right]$$

Existential forgery [edit]

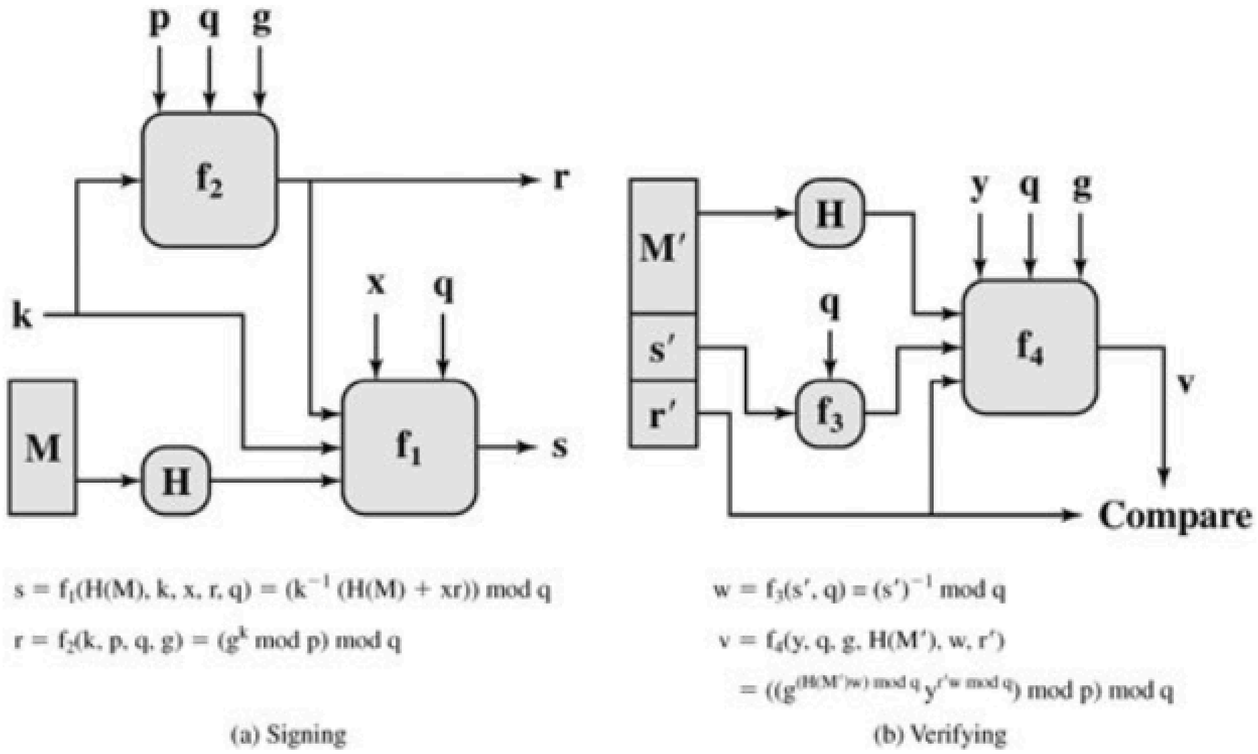
Existential forgery (existential unforgeability, EUF) is the creation (by an adversary) of at least one message/signature pair, (m, σ), where m has never been signed by the legitimate signer. The adversary can choose m freely; m need not have any particular meaning; the message content is irrelevant — as long as the pair, (m, σ), is valid, the adversary has succeeded in constructing an existential forgery. Thus, creating an existential forgery is easier than a selective forgery, because the attacker may select a message m for which a forgery can easily be created, whereas in the case of a selective forgery, the challenger can ask for the signature of a "difficult" message.

Weak existential forgery (strong existential unforgeability, strong unforgeability; sEUF, or SUF) [edit]

This notion is a stronger (more secure) variant of the existential forgery detailed above. Weak existential forgery is the creation (by an adversary) of at least one message/signature pair, (m, σ'), given a message and different signature (m, σ) produced by the legitimate signer.

Strong existential forgery is essentially the weakest adversarial goal, therefore the strongest schemes are those that are *strongly existentially unforgeable*.

Come viene implementata la firma digitale



(a) Signing

(b) Verifying

- p a prime modulus, where $2^{L-1} < p < 2^L$, and L is the bit length of p. Values for L are provided in Section 4.2.
- q a prime divisor of $(p - 1)$, where $2^{N-1} < q < 2^N$, and N is the bit length of q. Values for N are provided in Section 4.2.
- g a generator of a subgroup of order q in the multiplicative group of GF(p), such that $1 < g < p$.
- x the private key that must remain secret; x is a randomly or pseudorandomly generated integer, such that $0 < x < q$, i.e., x is in the range $[1, q–1]$.
- y the public key, where $y = g^x \bmod p$.
- k a secret number that is unique to each message; k is a randomly or pseudorandomly generated integer, such that $0 < k < q$, i.e., k is in the range $[1, q–1]$.

- Calcolare $u_1 = (H(m) * w) \bmod q$
- Calcolare $u_2 = (r * w) \bmod q$
- Calcolare $v = ((g^{u_1} * y^{u_2}) \bmod p) \bmod q$
- La firma è verificata se $v = r$

$L = 1024, N = 160$

$L = 2048, N = 224$

$L = 2048, N = 256$

$L = 3072, N = 256$

Curve Ellittiche

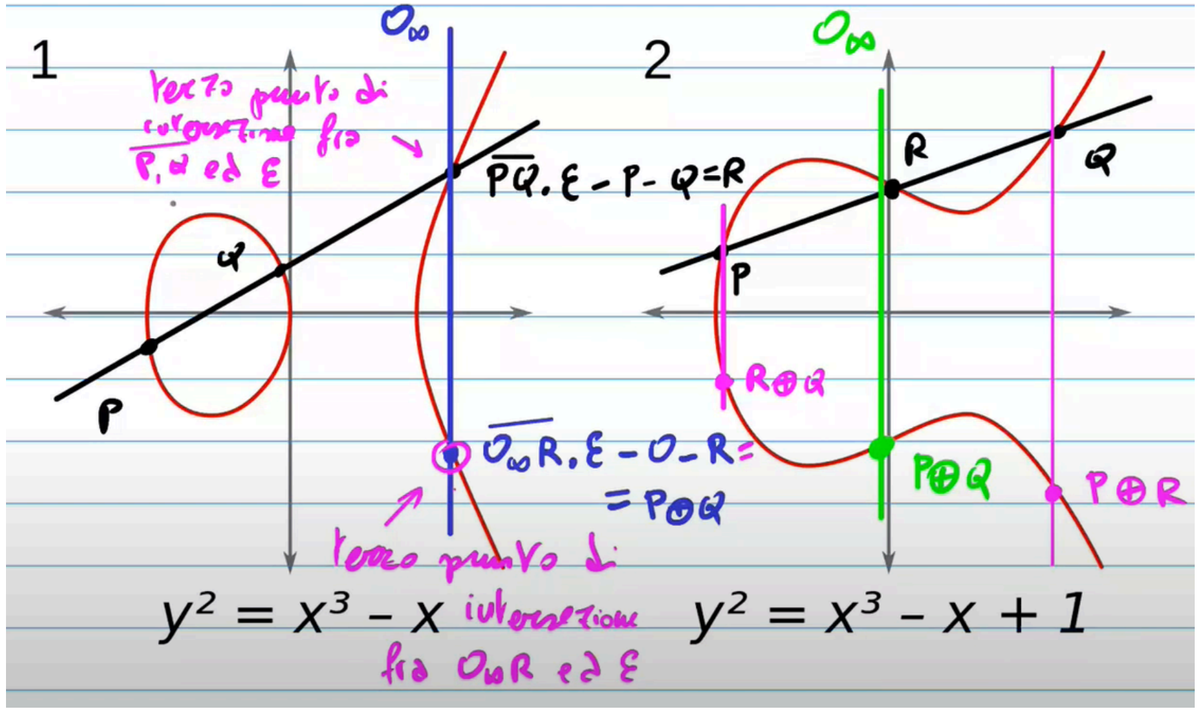
Let $\mathbb{F}_p$ be a prime finite field so that p is an odd prime number, and let $a, b \in \mathbb{F}_p$ satisfy $4a^3+27b^2 \not\equiv 0 \pmod{p}$. Then an elliptic curve $E(\mathbb{F}_p)$ over $\mathbb{F}_p$ defined by the parameters $a, b \in \mathbb{F}_p$ consists of the set of solutions or points $P = (x, y)$ for $x, y \in \mathbb{F}_p$ to the equation:

$$y^2 \equiv x^3 + ax + b \pmod{p}$$

Let $\mathbb{F}_{2^m}$ be a characteristic 2 finite field, and let $a, b \in \mathbb{F}_{2^m}$ satisfy $b \neq 0$ in $\mathbb{F}_{2^m}$. Then a (non-supersingular) elliptic curve $E(\mathbb{F}_{2^m})$ over $\mathbb{F}_{2^m}$ defined by the parameters $a, b \in \mathbb{F}_{2^m}$ consists of the set of solutions or points $P = (x, y)$ for $x, y \in \mathbb{F}_{2^m}$ to the equation:

$$y^2 + xy = x^3 + ax^2 + b \text{ in } \mathbb{F}_{2^m}$$

together with an extra point $\mathcal{O}$ called the point at infinity. (Here the only elliptic curves over $\mathbb{F}_{2^m}$ of interest are non-supersingular elliptic curves.)



ECDSA

Input: A valid set of elliptic curve domain parameters.
Output: A key pair (Q, d) associated with the elliptic curve domain parameters.

1. Select a statistically unique and unpredictable integer d in the interval $[1, n-1]$.
2. Compute the point $Q = (x_Q, y_Q) = dG$. (See Annex D.3.2.)
3. The key pair is (Q, d), where Q is the public key, and d is the private key.

5.3 Signature Generation

This section describes the ECDSA signature generation process.
The signature generation process consists of:

1. Message digesting.
2. Elliptic curve computations.
3. Modular computations.

The inputs to the signature process are:

1. The message, M, of an arbitrary length, which is represented by a bit string.
2. A valid set of elliptic curve domain parameters.
3. An elliptic curve private key, d, associated with the elliptic curve domain parameters.

The output of the signature process are two integers r and s (the digital signature), where $1 \le r \le n-1$, $1 \le s \le n-1$.

Compute the elliptic curve point $(x_1, y_1) = kG$. (See Annex D.3.2.)

Convert the field element x_1 to an integer $\bar{x}_1$, as described in Section 4.3.5.

Set $r = \bar{x}_1 \bmod n$.

If $r = 0$, then go to step 1 of Section 5.3.2.

Compute $s = k^{-1}(e + dr) \bmod n$. (See Annex D.1.2. for one method to compute $k^{-1} \bmod n$.)

If $s = 0$, then go to step 1 of Section 5.3.2.

5.4 Signature Verification

This section describes the ECDSA signature verification process.
The signature verification process consists of:

1. Message digesting.
2. Modular computations.
3. Elliptic curve computations.
4. Signature checking.

The input to the signature verification process is:

1. The received message, M′, represented as a bit string.
2. The received signature for M′, represented as the two integers, r' and s'.
3. A valid set of elliptic curve domain parameters.
4. A valid public key, Q, associated with the elliptic curve domain parameters.

The output of the signature verification process is an indication of signature verification success or failure.

5.4.2 Modular Computations

1. If r' is not an integer in the interval $[1, n-1]$, then reject the signature.
2. If s' is not an integer in the interval $[1, n-1]$, then reject the signature.
3. Compute $c = (s')^{-1} \bmod n$. (See Annex D.1.2.)
4. Compute $u_1 = e'c \bmod n$ and $u_2 = r'c \bmod n$.

5.4.3 Elliptic Curve Computations

1. Compute the elliptic curve point $(x_1, y_1) = u_1G + u_2Q$ (see Annex D.3.2). (If $u_1G + u_2Q$ is the point at infinity, then reject the signature.)

5.4.4 Signature Checking

1. Convert the field element x_1 to an integer $\bar{x}_1$, as described in Section 4.3.5.
2. Compute $v = \bar{x}_1 \bmod n$.
3. If $r' = v$, then the signature is verified, and the verifier has a high level of confidence that the received message was sent by the party holding the secret key d corresponding to Q.
   If r' does not equal v, then the message may have been modified, the message may have been incorrectly signed by the signatory, or the message may have been signed by an impostor. The message shall be considered invalid.

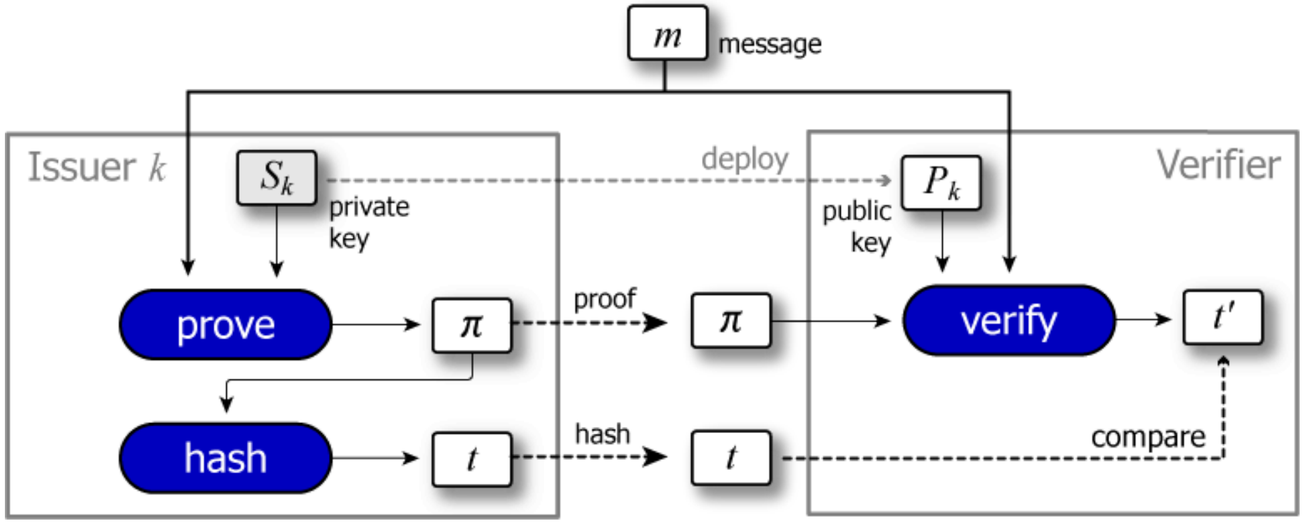
Primitive	Parameters	Legacy System Minimum	Future System Minimum
RSA Problem	N, e, d	$\ell(n) \geq 1024$, $e \geq 3$ or 65537, $d \geq N^{1/2}$	$\ell(n) \geq 3072$ $e \geq 3$ or 65537, $d \geq N^{1/2}$
Finite Field DLP	p, q, n	$\ell(p^n) \geq 1024$ $\ell(p), \ell(q) > 160$	$\ell(p^n) \geq 3072$ $\ell(p), \ell(q) > 256$
ECDLP	p, q, n	$\ell(q) \geq 160$, $\star$	$\ell(q) > 256$, $\star$
Pairing	p, q, n, d, k	$\ell(p^{k \cdot n}) \geq 1024$ $\ell(p), \ell(q) > 160$	$\ell(p^{k \cdot n}) \geq 3072$ $\ell(p), \ell(q) > 256$

Verifiable Random Functions

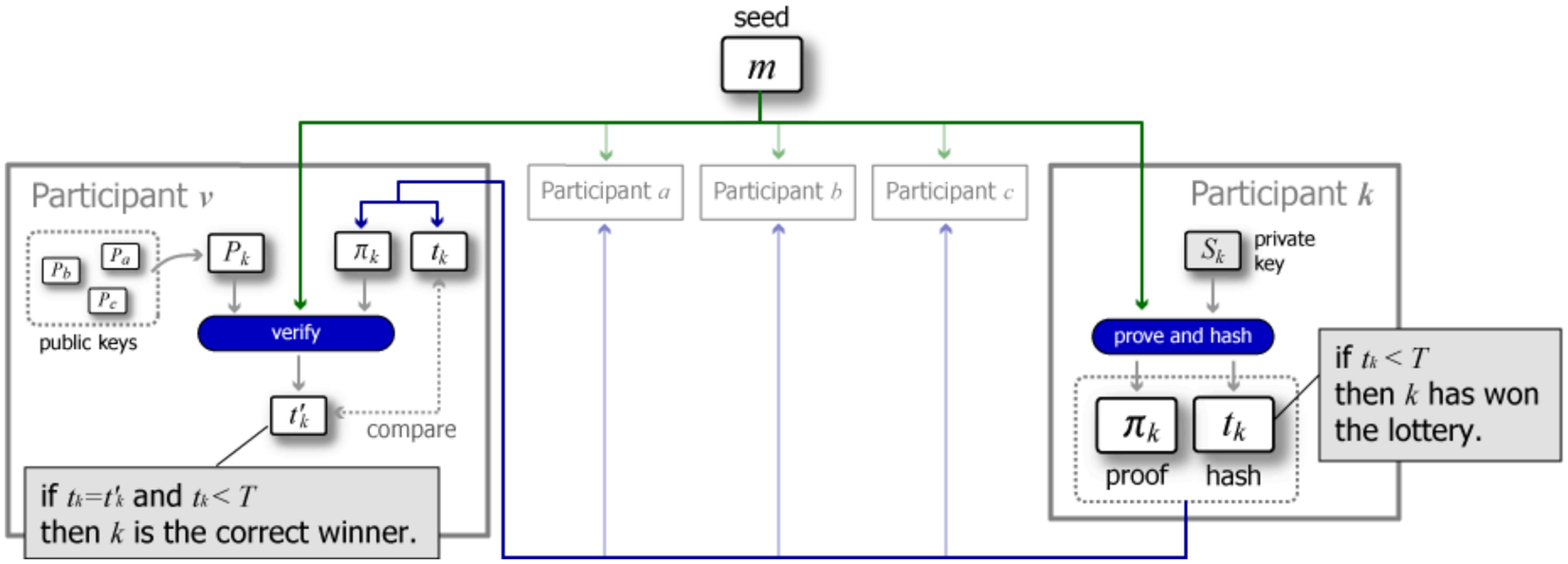
- **Verifiable**—Anyone can verify that the random number generated by a VRF is valid. All they need to do is inspect the proof and verify the correctness of the hash output. While only the holder of the VRF secret key can compute the hash, anyone with the public key can verify the correctness of the hash.

- **Random**—The output of a VRF is entirely unpredictable (uniformly distributed) to anyone who doesn't know the seed or private key and follows no pattern. In a VRF, every possible output is equally likely. The randomness is generated by combining the seed and private key in a unique manner.

- **Function**—VRFs rely on a mathematical algorithm to produce both the random number and a proof that verifies its authenticity. For a function to be considered a VRF, the RNG must keep the seed hidden (implicit) to preserve its unpredictability, while the proof must be explicit and calculable by everyone (explicit) to ensure its verifiability.

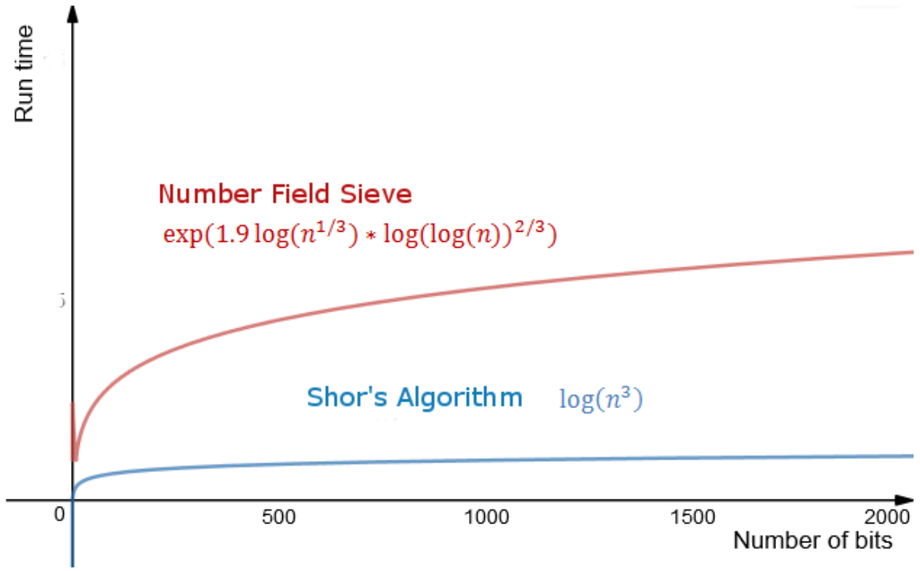


$$\begin{cases} Pi &= \text{PRove}(S_{.K}, M) \\ T &= \text{HUNSH}(Pi) \end{cases}$$



https://hazm.at/mox/security/vrf/index.html

La questione Quantum



One common characteristic of many post-quantum cryptography algorithms is that they require larger key sizes than commonly used "pre-quantum" public key algorithms. There are often tradeoffs to be made in key size, computational efficiency and ciphertext or signature size. The table lists some values for different schemes at a 128 bit post-quantum security level.

Algorithm	Type	Public Key	Private Key	Signature
NTRU Encrypt[37]	Lattice	766.25 B	842.875 B	
Streamlined NTRU Prime[*citation needed*]	Lattice	154 B		
Rainbow[38]	Multivariate	124 KB	95 KB	
SPHINCS[19]	Hash Signature	1 KB	1 KB	41 KB
SPHINCS+[39]	Hash Signature	32 B	64 B	8 KB
BLISS-II	Lattice	7 KB	2 KB	5 KB
GLP-Variant GLYPH Signature[10][40]	Ring-LWE	2 KB	0.4 KB	1.8 KB
NewHope[41]	Ring-LWE	2 KB	2 KB	
Goppa-based McEliece[14]	Code-based	1 MB	11.5 KB	
Random Linear Code based encryption[42]	RLCE	115 KB	3 KB	
Quasi-cyclic MDPC-based McEliece[43]	Code-based	1,232 B	2,464 B	
SIDH[44]	Isogeny	564 B	48 B	
SIDH (compressed keys)[45]	Isogeny	330 B	48 B	
3072-bit Discrete Log	**not PQC**	384 B	32 B	96 B
256-bit Elliptic Curve	**not PQC**	32 B	32 B	65 B