Permissionless VS Permissioned

Definizione di Permissionless

Una blockchain **permissionless** è aperta alla partecipazione di chiunque, senza restrizioni di ingresso. Questo comporta che:

1. **Chiunque** può eseguire un nodo completo e partecipare alla validazione dei blocchi.
2. Non esiste un’autorità centrale.
3. Il meccanismo di consenso robusto a partecipanti potenzialmente malevoli e anonimi.

es Ethereum

Definizione di Permissioned

Una blockchain **permissioned** prevede un insieme di partecipanti che devono essere **autenticati** e **autorizzati**

Hyperledger Fabric si colloca in questa categoria:

- I nodi appartengono a organizzazioni che partecipano a un consorzio.
- L'autorizzazione a svolgere ruoli specifici è gestita tramite una PKI interna al sistema.
- La governance e gli aggiornamenti avvengono tramite procedure tra membri autorizzati.

Meccanismi di Consenso

- **Ethereum**:
  - **Proof-of-Work (PoW)**
  - Con l'introduzione di **Ethereum 2.0** (anche noto come la *Merge*), è passato a **Proof-of-Stake (PoS)**

- **Hyperledger Fabric**:
  - Spesso usa protocolli di tipo **Crash Fault Tolerant (CFT)** o **Byzantine Fault Tolerant (BFT)** per garantire consenso.
  - Il focus sulle **performance** e sulla gestione delle identità

Scalabilità e Performance

- **Permissionless (Ethereum)**:
  - Lento e meno scalabile su layer 1
  - Può raggiungere qualche decina di transazioni al secondo, sebbene esistano soluzioni layer 2 (rollup, sidechain)
- **Permissioned (Fabric)**:
  - Si ottengono performance migliori, con potenzialmente centinaia o migliaia di transazioni al secondo.
  - È spesso utilizzato in contesti enterprise dove la scalabilità e l'affidabilità contano più della decentralizzazione spinta.

Governance

- **Ethereum**:
  - Governata da una community aperta.
  - L’aggiornamento del protocollo richiede un coordinamento su scala globale
  - È un ecosistema eterogeneo senza un controllo centralizzato.

- **Fabric**:
  - Determinata da chi crea la rete e dai membri del consorzio.
  - Gli aggiornamenti si fanno in modo coordinato e disciplinato da contratti legali o accordi di collaborazione.
  - C’è meno “democrazia” nel senso tradizionale, ma più chiarezza in termini di responsabilità e compliance normativa.

Definizione di Smart Contract Permissionless e Permissioned

Smart Contract in Ethereum (Permissionless)

Scritti tipicamente in **Solidity** (o altri linguaggi compatibili con la Ethereum Virtual Machine) e caricati sulla blockchain. Qualunque utente può:

1. **Deployare** il contratto
2. **Invocare** le funzioni del contratto

La natura permissionless di Ethereum implica che:

- **Chiunque** può caricare uno smart contract
- Il codice e le transazioni sono **pubblici**
- L’esecuzione del contratto è verificata da ogni nodo

Smart Contract in Ethereum (Permissionless)

```
pragma solidity ^0.8.0;

contract SimpleStorage {

        uint256 public storedData;

        constructor(uint256 initialData) {

                storedData = initialData;

        }

        function set(uint256 x) public {

                storedData = x;

        }

        function get() public view returns (uint256) {

        return storedData;

        }

}
```

Smart Contract in Hyperledger Fabric (Permissioned)

Prendono il nome di **chaincode**.

1. Il chaincode è installato sui **peer** di una o più organizzazioni facenti parte del consorzio.
2. Solo gli endorser designati possono eseguire il chaincode
3. L'ordine delle transazioni è gestito dai nodi orderer.
4. I peer infine **committano** le transazioni validate, aggiornando il ledger.

In questo contesto:

- L'accesso e la partecipazione sono regolamentati da una **PKI interna**
- Puoi sviluppare chaincode in diversi linguaggi.
- Il codice e i dati possono essere nascosti a chi non fa parte di un **channel** specifico

Smart Contract in Hyperledger Fabric (Permissioned)

```
package main
import (
"fmt"
"strconv"
"github.com/hyperledger/fabric-contract-api-go/contractapi"
)
type SimpleStorageChaincode struct {
	contractapi.Contract }

func (s *SimpleStorageChaincode) SetData(ctx contractapi.TransactionContextInterface, key string, value string) error {
	return ctx.GetStub().PutState(key, []byte(value)) }

func (s *SimpleStorageChaincode) GetData(ctx contractapi.TransactionContextInterface, key string) (string, error) {
	data, err := ctx.GetStub().GetState(key)
	if err != nil {
		return "", err
	}
	return string(data), nil }

func main() { chaincode, err := contractapi.NewChaincode(new(SimpleStorageChaincode))
	if err != nil { fmt.Printf("Error create chaincode: %s", err.Error()) }
	if err := chaincode.Start(); err != nil { fmt.Printf("Error starting chaincode: %s", err.Error()) }
}
```

Sicurezza dei Dati nella Blockchain Permissionless e Permissioned

Permissionless

Aspetti di sicurezza:

- Integrità garantita dalla catena di hash e da PoW o PoS.
- Aperta a partecipanti anonimi, quindi deve difendersi da potenziali attacchi di Sybil.
- La trasparenza è massima

Rischi:

- **51% attack**
- **Centralizzazione de facto**

Permissioned

Aspetti di sicurezza:

- Integrità garantita da protocolli BFT o CFT.
- I partecipanti sono noti e certificati.
- È possibile implementare **canali privati**.

Rischi:

- **Collusione**
- **Gestione centralizzata delle chiavi**

Aspetti Critici della Tecnologia Permissioned e Permissionless

Scalabilità vs Decentralizzazione

Blockchain Trilemma → difficoltà nel bilanciare **scalabilità**, **sicurezza** e **decentralizzazione**:

- Le blockchain permissionless (tipo Ethereum) puntano molto su **decentralizzazione** e **sicurezza** crittografica, spesso a scapito della scalabilità.
- Le blockchain permissioned (tipo Fabric) puntano molto su **scalabilità e sicurezza**, ma rinunciano in parte all'ideale di decentralizzazione totale.

Governance e Aggiornabilità

- Permissionless → un cambio di regole (hard fork) richiede un consenso di fatto di tutta la community. Se una parte rifiuta di aggiornarsi, si può creare una catena parallela (splitting).

- Permissioned → la governance è più chiara **ma** ciò implica che un ristretto gruppo di attori possa decidere modifiche e imporle a tutti.

Privacy e Conformità Normativa

- Le reti permissionless sono completamente pubbliche **ma** difficile “dimenticare” o “cancellare” dati.

- Le reti permissioned offrono modalità di protezione e canali privati.

Modelli Ibridi: Permissionless + Permissioned

Si tratta di soluzioni che tentano di conciliare i benefici delle reti aperte con quelli delle reti private, offrendo:

1. **Ancoraggio di fiducia**
2. **Sidechain o Parachain**
3. **Interoperabilità e bridging**