

Law, Science and Technology
MSCA ITN EJD n. 814177



Mirko Zichichi

Blockchain e
Smart contracts

Outline

- Introduzione ai Sistemi Distribuiti
 - Sistemi Distribuiti
 - Blockchain
- Approccio “Law + Technology”
- La Specie Smart Contract e le sue Varietà
 - Funzionamento
 - Immutabilità
 - Varietà della specie
 - Interazioni tra varietà e con il mondo esterno

Concetti di base necessari

Cryptographic **Hash** function

SHA256

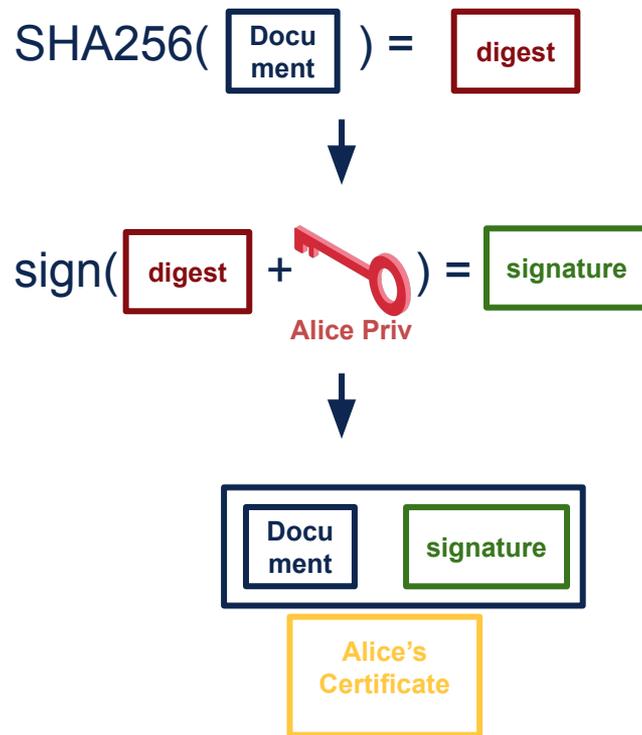
$$f(\textit{string}) = \textit{digest}$$

$$SHA256(\textit{moat}) = 98E2\dots16F3$$

$$SHA256(\textit{m\textcircled{a}ot}) = E671\dots A9C9$$


la lunghezza del digest è
sempre la stessa
(256 bit per SHA256)

Firma Digitale

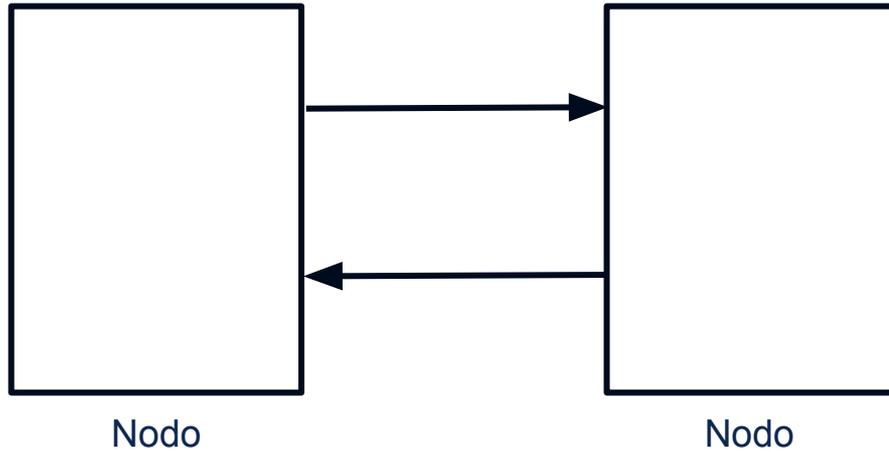


Introduzione ai Sistemi Distribuiti e Blockchain

+ Sistema Distribuito

Sistema informatico costituito da un insieme di **processi** interconnessi tra loro in cui le comunicazioni avvengono solo esclusivamente tramite lo scambio di opportuni messaggi.

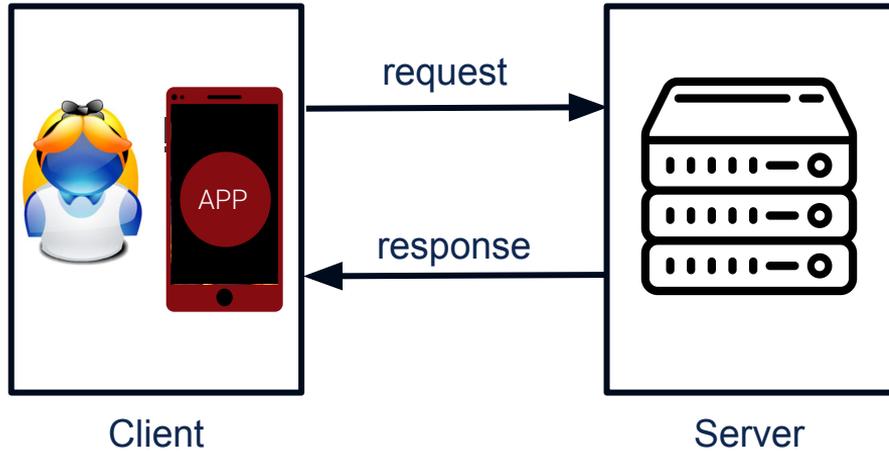
Nodo



Qualsiasi **dispositivo** hardware del sistema in grado di **comunicare** con gli altri dispositivi che fanno parte della rete

Dispongono di una memoria propria, di un proprio sistema operativo e di risorse locali

Architettura Client/Server

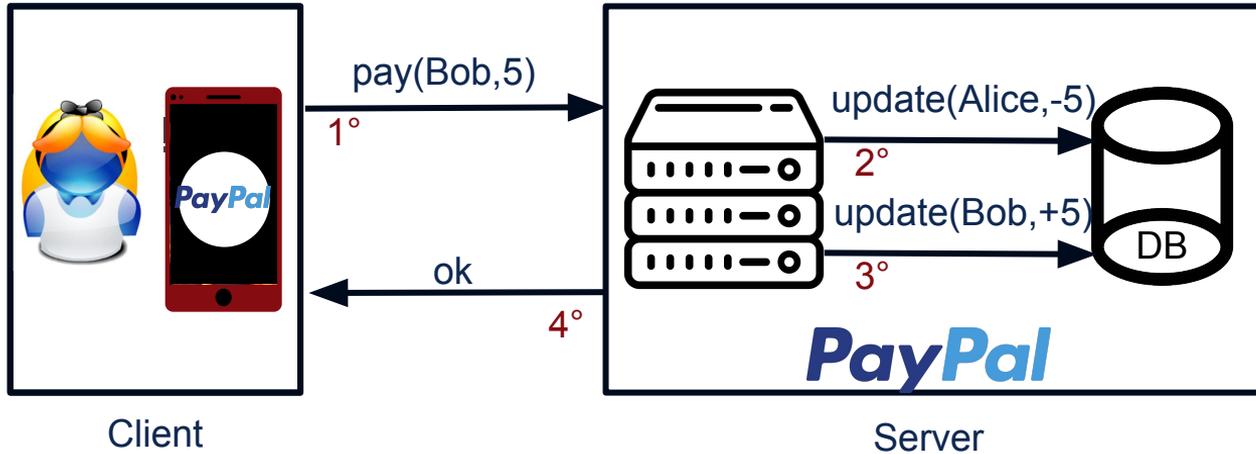


Un'architettura di sistema è il **modello concettuale** che definisce la struttura, il comportamento e più prospettive di un unico sistema

Architettura Client/Server

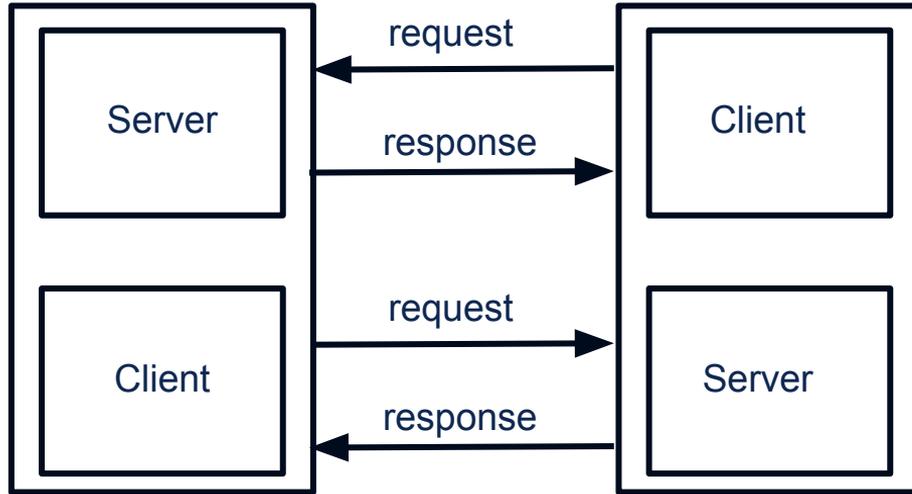
Esempio

Alice
paga
Bob
5 euro



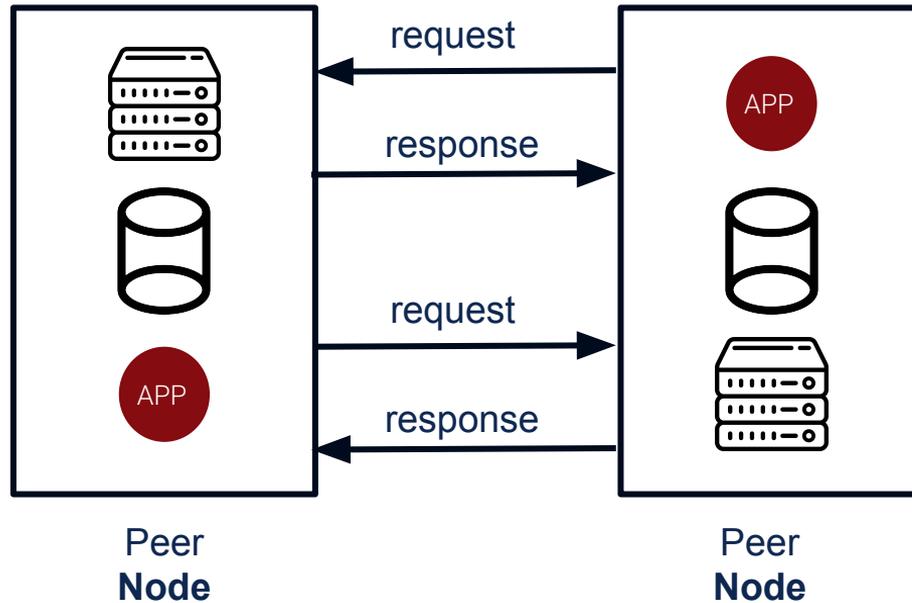
Architettura Client/Server

I nodi Peers sono
Client e Server
simultaneamente



Architettura Peer to Peer (P2P)

I nodi Peers sono
Client e Server
simultaneamente



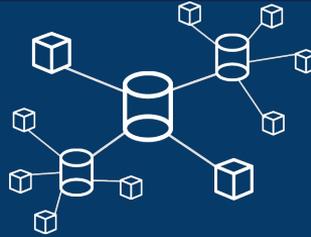
+ Sistema Distribuito

Sistema informatico costituito da un insieme di **processi** interconnessi tra loro in cui le comunicazioni avvengono solo esclusivamente tramite lo scambio di opportuni messaggi.



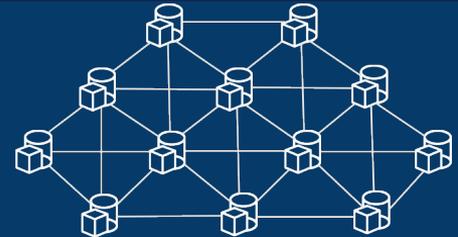
Centralizzato

Un nodo si occupa di tutto

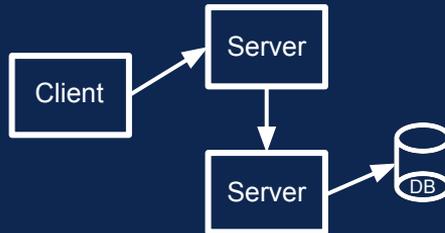


De-centralizzato

Il carico totale dei compiti viene decentralizzato e coinvolge i sotto-nodi



De-centralizzato Distribuito (P2P)





BLOCKCHAIN

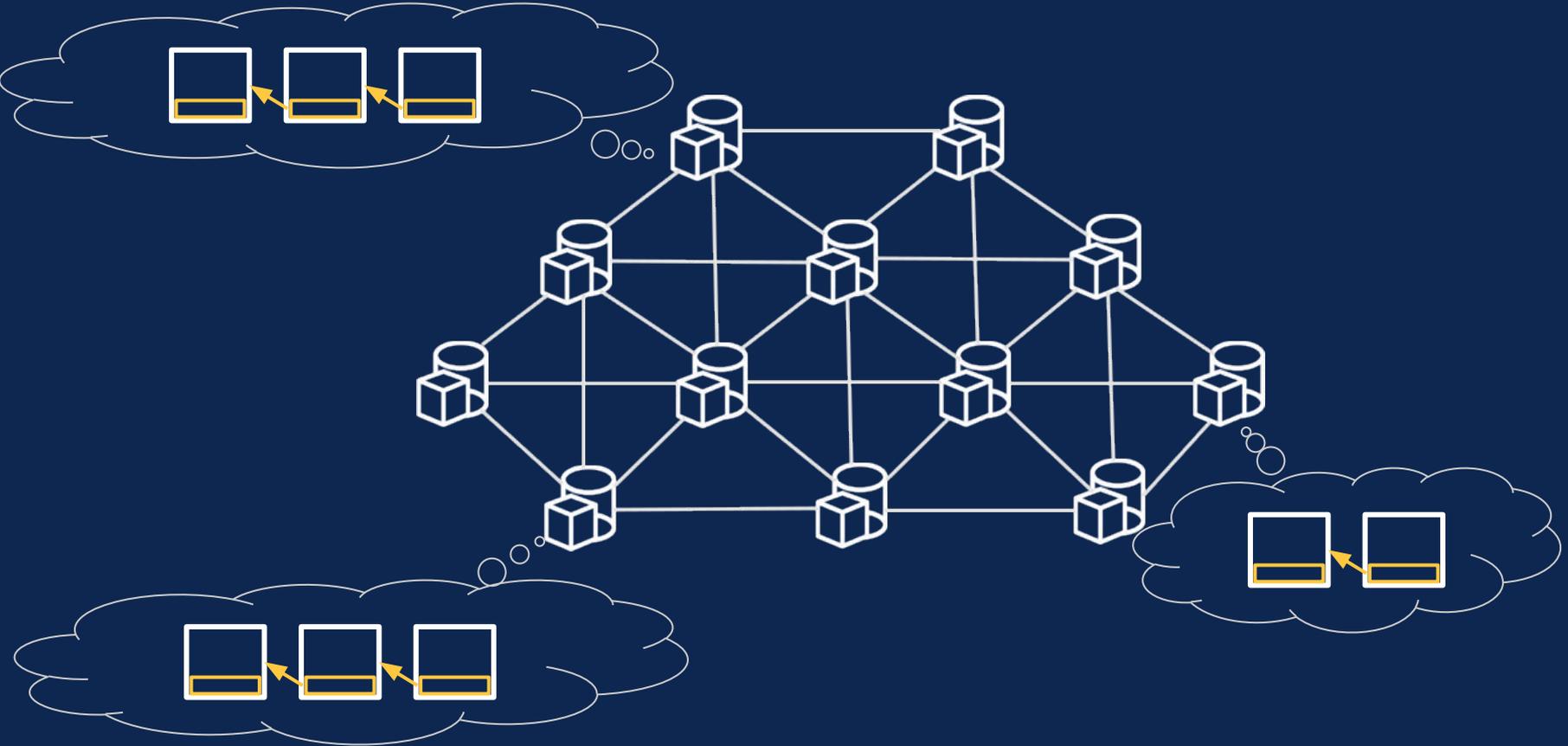
Sistema Distribuito basato su una rete di nodi P2P

- È una tecnologia che fa parte del regno delle DLTs:
Distributed Ledger Technologies
- Nelle DLTs un registro viene distribuito tra i nodi di una rete P2P, che aggiornano la loro **copia locale** secondo un unico meccanismo di consenso
- Una blockchain è una DLT in cui il registro assume la forma di un **insieme di blocchi di dati (relativamente) ordinati cronologicamente**

+

BLOCKCHAIN

Sistema Distribuito basato su una rete di nodi P2P



+ Blockchain

- Cosa scrivere sul registro → **transazioni**

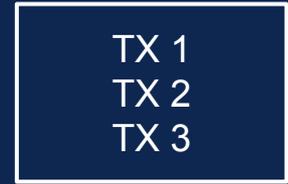


- Struttura del registro → **chain of blocks**



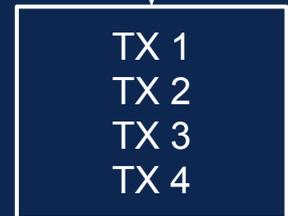
+ Transazioni

- Se il registro mantiene lo stato del sistema → allora una transazione è l'operazione che modifica questo stato
- Lo stato del sistema in un certo momento (snapshot) è un elenco di transazioni
- Una nuova transazione si riferisce ad una precedente e aggiorna lo stato del sistema
- Una transazione valida viene firmata utilizzando la firma digitale dell'account a cui fa riferimento la transazione precedente

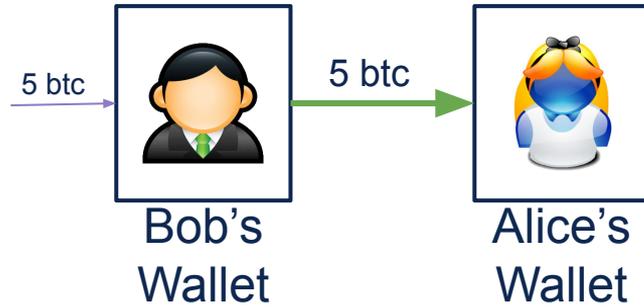


+

TX 4



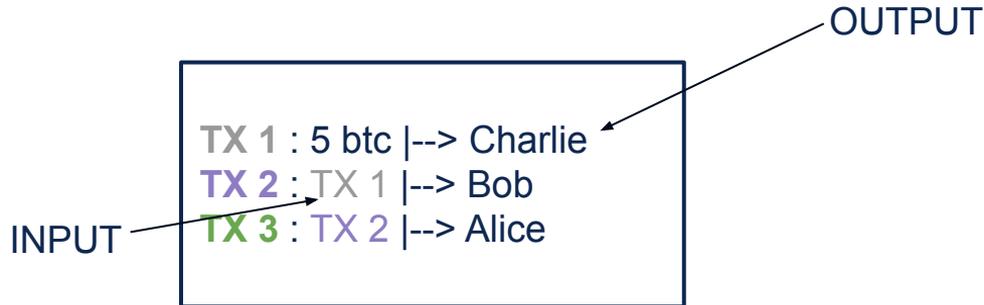
Transazioni: Esempio



TX 1 : 5 btc |--> Charlie
TX 2 : Charlie |--> Bob
TX 3 : Bob |--> Alice

UTXO: Unspent Transaction (TX) Output

TX 1 : 5 btc |--> Charlie
TX 2 : Charlie |--> Bob
TX 3 : Bob |--> Alice



Ogni transazione nella blockchain contiene almeno un OUTPUT

Gli output vengono poi spesi dagli INPUT di transazioni successive

Transazioni

Gli INPUT devono essere sbloccati con una firma digitale

TX 1 : 5 btc |--> Charlie Pub
TX 2 : TX 1 |--> Bob Pub
TX 3 : TX 2 |--> Alice Pub

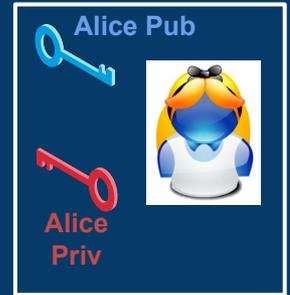


+ TX 4 : TX 3 |--> Dana Pub

sign(TX 4, Alice Priv)



TX 1 : 5 btc |--> Charlie Pub
TX 2 : TX 1 |--> Bob Pub
TX 3 : TX 2 |--> Alice Pub
TX 4 : TX 3 |--> Dana Pub



Alice's
Wallet

+ Blockchain

- Cosa scrivere sul registro → **transazioni**

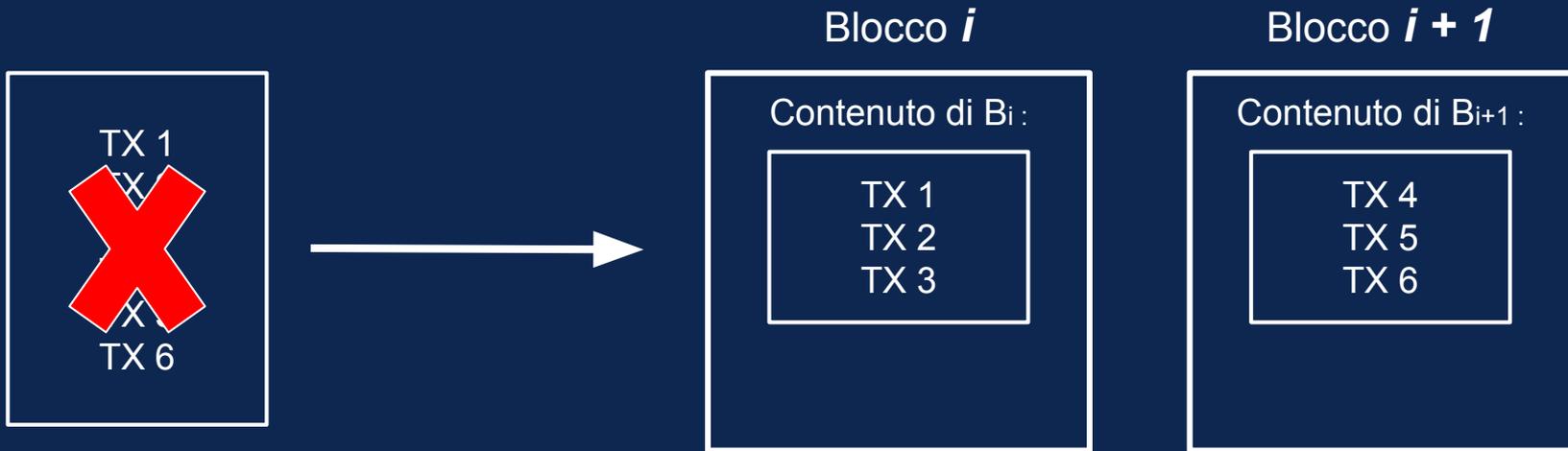


- Struttura del registro → **chain of blocks**



+ Registro a Blocchi

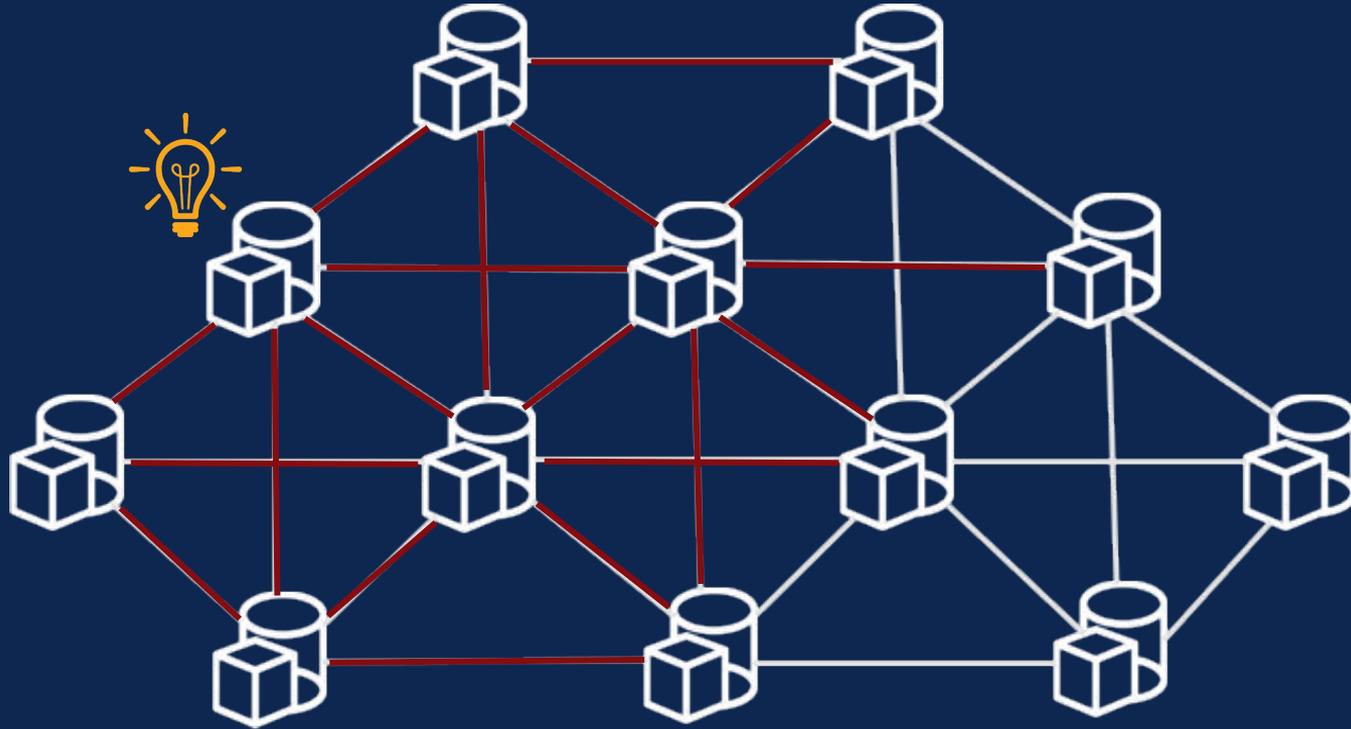
invece di avere un unico documento contenente tutto il registro di transazioni, la blockchain lo divide in BLOCCHI:



+ Registro distribuito: creazione di un nuovo blocco

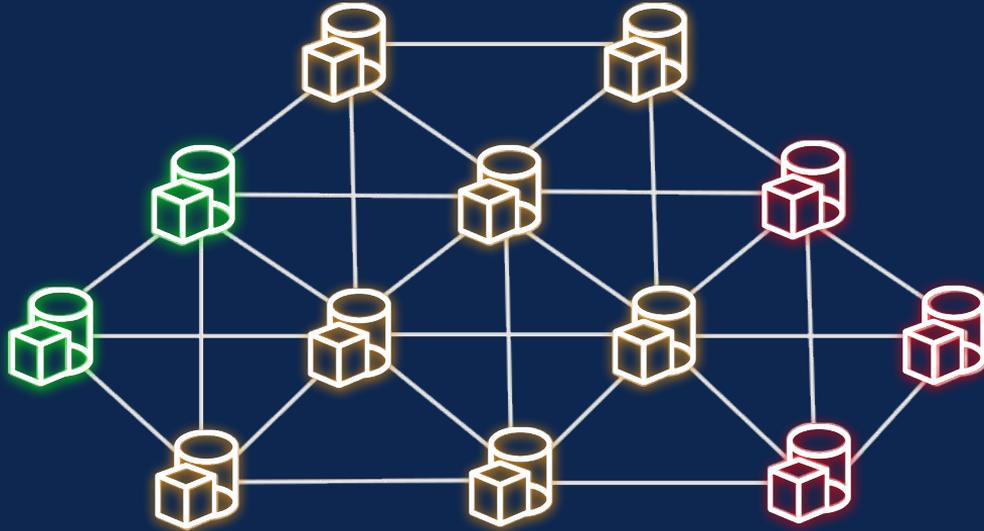
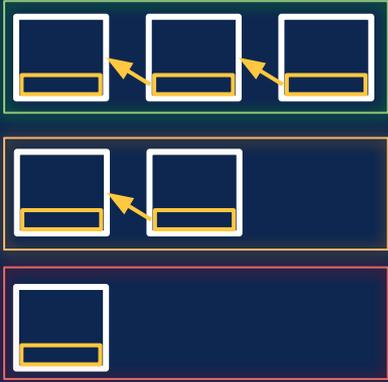


+ Registro distribuito: propagazione di un blocco



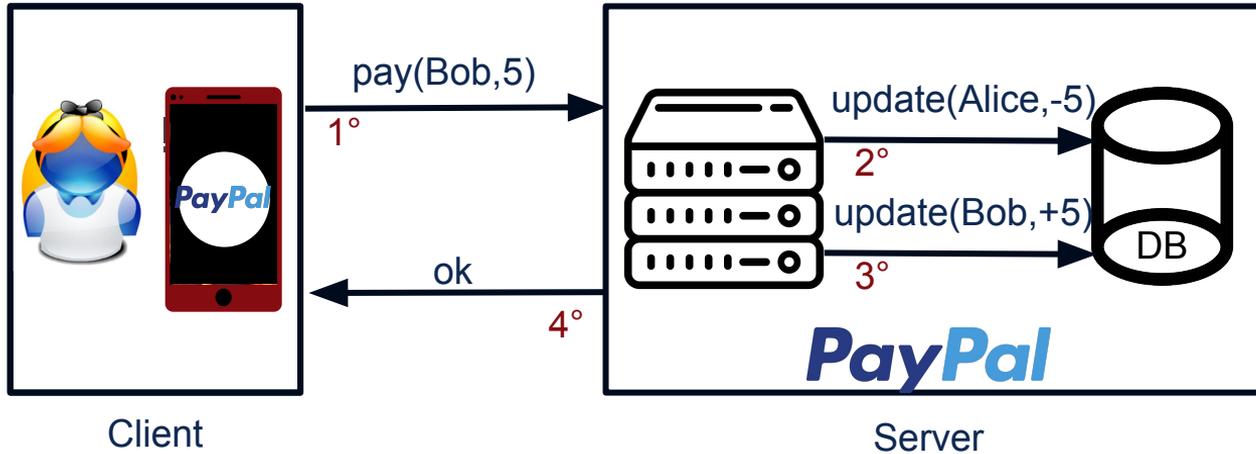
Registro distribuito: sincronizzazione

+



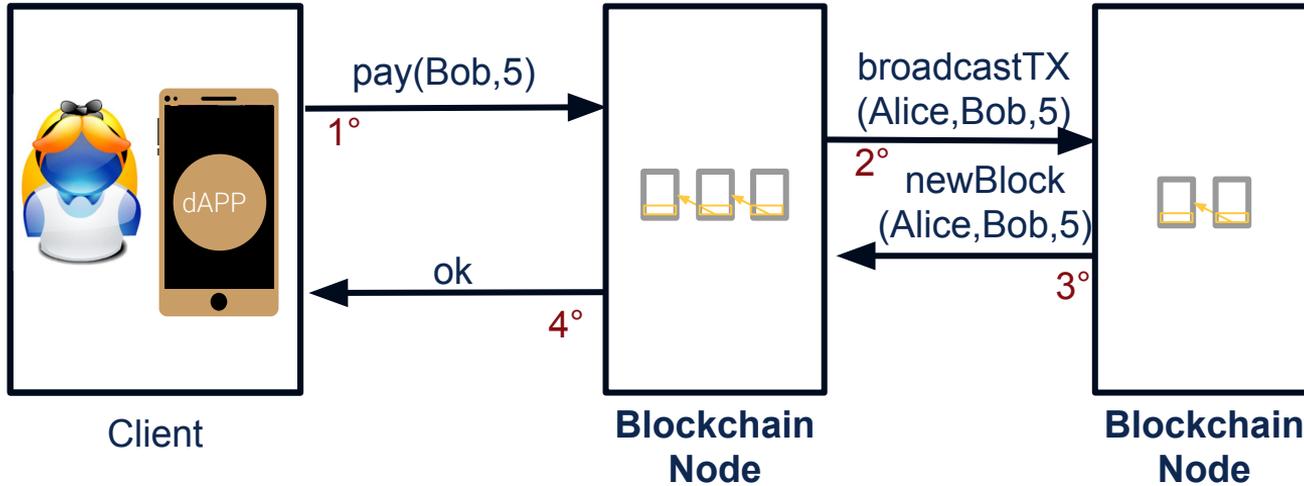
Comparazione Blockchain e Client/Server

Alice
paga
Bob
5 euro



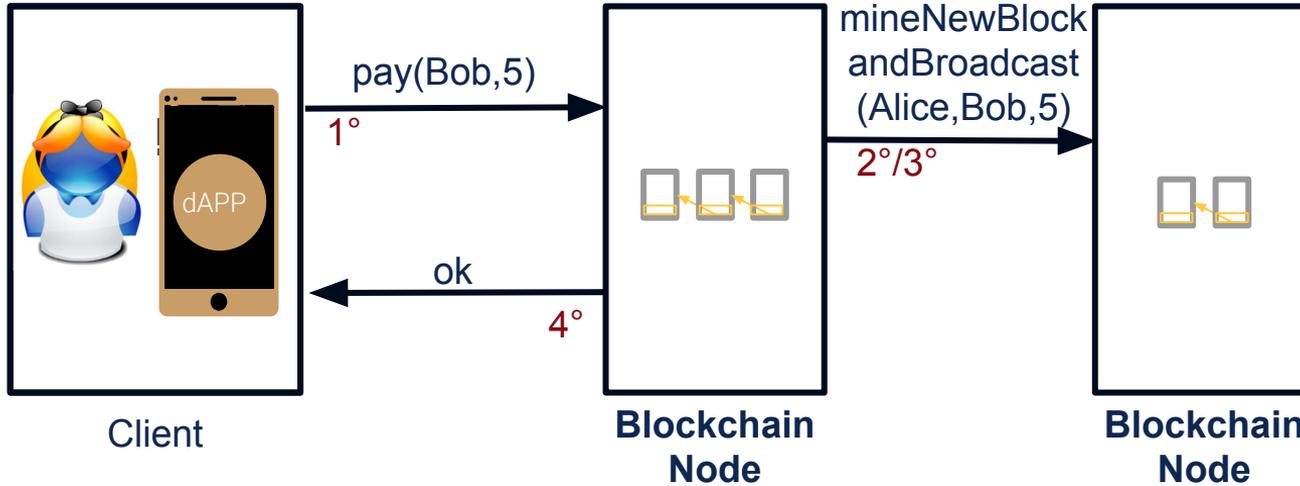
Comparazione Blockchain e Client/Server

Alice
paga
Bob
5 bitcoin



Comparazione Blockchain e Client/Server

Alice
paga
Bob
5 bitcoin



Law + technology

Approccio

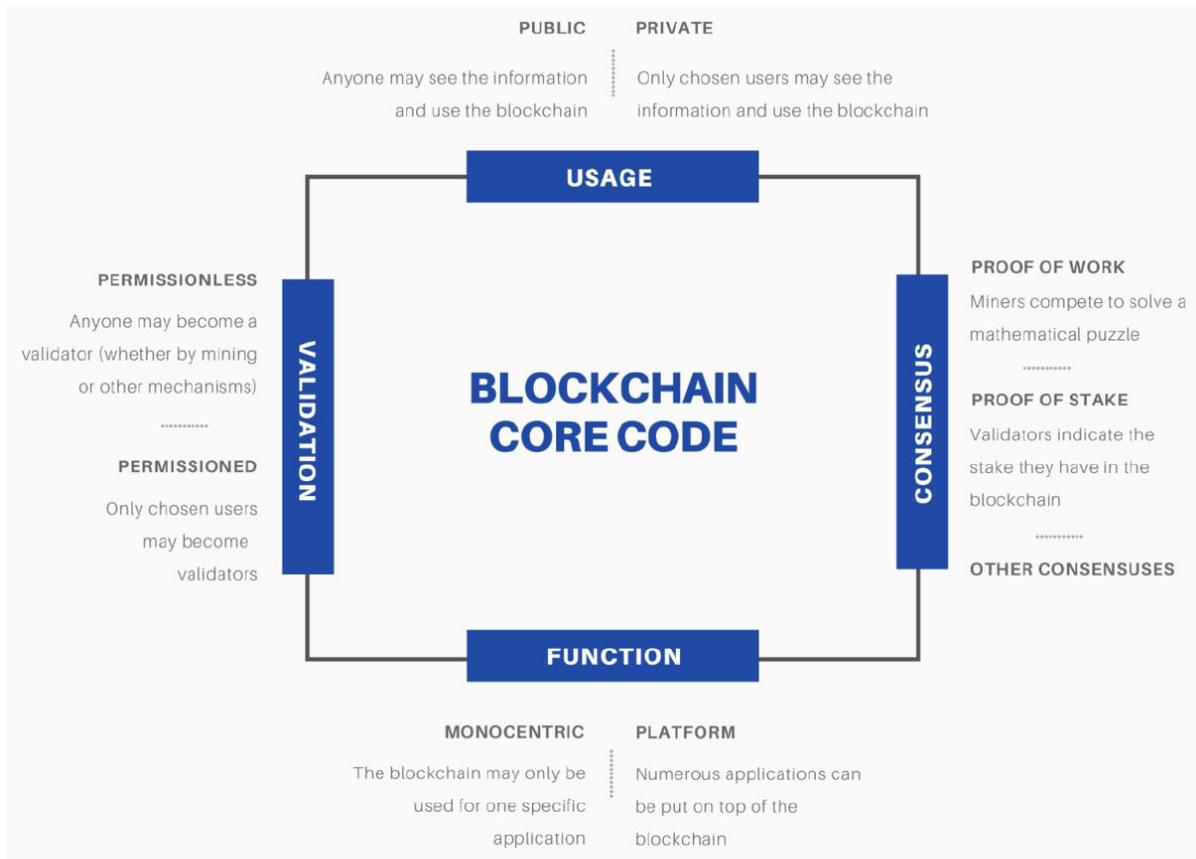
*“Smart Contracts and the Digital Single Market Through the Lens of a ‘Law + Technology’ Approach”, DR. THIBAUT SCHREPEL, LL.M.
https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3947174*



Un punto di vista evolutivo

- Bitcoin si riferisce e fa uso di ricerche, concetti e tecniche del passato, combinando questi elementi preesistenti per dare origine alla blockchain
- **Una volta che una nuova classe di tecnologia è emersa, segue un processo darwiniano di selezione naturale**
 - La tecnologia → **specie**
 - si muove in diverse direzioni simultaneamente, portando all'emergere di diverse → **varietà**
 - Le varietà che sopravvivono si moltiplicano e cercano di espandere il loro territorio, entrano in contatto con altre specie e iniziano a competere con loro.

La Specie Blockchain e le sue Varietà



A look at blockchain varieties

© Thibault Schrepel⁷

+ La competizione della blockchain

- Blockchain sta appena iniziando a competere con i mezzi transazionali centralizzati
 - cryptovalute vs. denaro fiat
- La competizione che è inizialmente forte tra le varietà di blockchain sta reggiungendo una competizione tra specie
 - blockchain vs. ecosistemi centralizzati
- Blockchain **sopravviverà** solo se manterrà forti elementi di **differenziazione per ottenere un vantaggio competitivo** sulle altre specie in un dato ambiente



La specie Smart Contracts

- La tecnologia Smart Contract sfrutta le blockchains così come una specie dipende da un'altra
- L'ambiente degli smart contracts ha
 - dimensioni legali, cioè soft law, regolamenti, case law, ecc.
 - dimensioni tecniche, la blockchain
- Devono essere combinati → in assenza di **cooperazione tra legge e tecnologia**, questi due aspetti lotterebbero per prendere il sopravvento
- Un approccio più **cooperativo e armonizzato** è quindi preferibile in modo che gli smart contract possano crescere in un ambiente coeso e duraturo



Possibili Approcci

Assolutista

- Law perspective:

Creare leggi senza cercare il modo di avvicinarsi alla tecnologia

- Technology perspective:

Il fondamentalismo tecnico consiste nel progettare la tecnologia senza fare affidamento su leggi, portando alla creazione di "zone temporaneamente autonome" (TAZ).

Svantaggi:

→ comporta l'applicazione di regole e standard legali senza cercare di preservare gli elementi di differenziazione necessari per la sopravvivenza della tecnologia

→ non appena la tecnologia estende il suo territorio e lascia la TAZ, l'applicazione della legge può portare all'estinzione della tecnologia.

Cooperativo

- legge e la tecnologia si completano a vicenda cercando di preservare la loro sfera d'influenza e costruendo sui punti di forza dell'altro

- mantenere le caratteristiche distintive della blockchain mentre viene permessa l'applicazione della legge

Vantaggi:

→ si possono usare gli smart contracts dove il diritto contrattuale è difficile da far rispettare, per esempio, perché le giurisdizioni sono poco amichevoli

→ usati dove la legge non può raggiungere un obiettivo da sola, come prevenire la corruzione

La Specie Smart Contract e le sue Varietà

+ Uno Smart Contract è (semplicemente) un programma che viene eseguito da tutti i nodi di una rete di una DLT

```
/**
 * @dev Log a vote for a challenge
 * @param challengeID The challenge position in the list
 * @param inFavour Boolean value indicating if in favour or not
 */
function vote(uint256 challengeID, bool inFavour)
    public
    onlyEligible(challengeID)
    notExecuted(challengeID)
{
    Challenge storage chall = _challenges[challengeID];
    Vote storage v = chall.votes[msg.sender];
    require(!v.voted, "Voting: already voted");

    _vote(challengeID, inFavour);

    emit Voted(challengeID, msg.sender, inFavour);
}
```



Origine della specie Smart Contract

- **Nick Szabo, 1994** → "un protocollo computerizzato che esegue i termini di un contratto, implementato con programmi su una rete di computer, 'più intelligente' dei suoi antenati su carta"
 - *smart* ← *intelligere* ← *scegliere tra*
 - gli smart contract **automatizzano la scelta** in base a **condizioni predefinite**
- **Esempio: distributore automatico**
 - il produttore predetermina le condizioni (inserire una moneta X)
 - la macchina può eseguire il compito (consegnare il prodotto) *se queste condizioni sono soddisfatte*



Vending Machine



Alice



Manufacturer

La certezza di una corretta esecuzione è limitata alla fiducia che si ha nel produttore

Se il produttore dovesse cambiare i termini del distributore automatico, potrebbe ingannare il potenziale acquirente

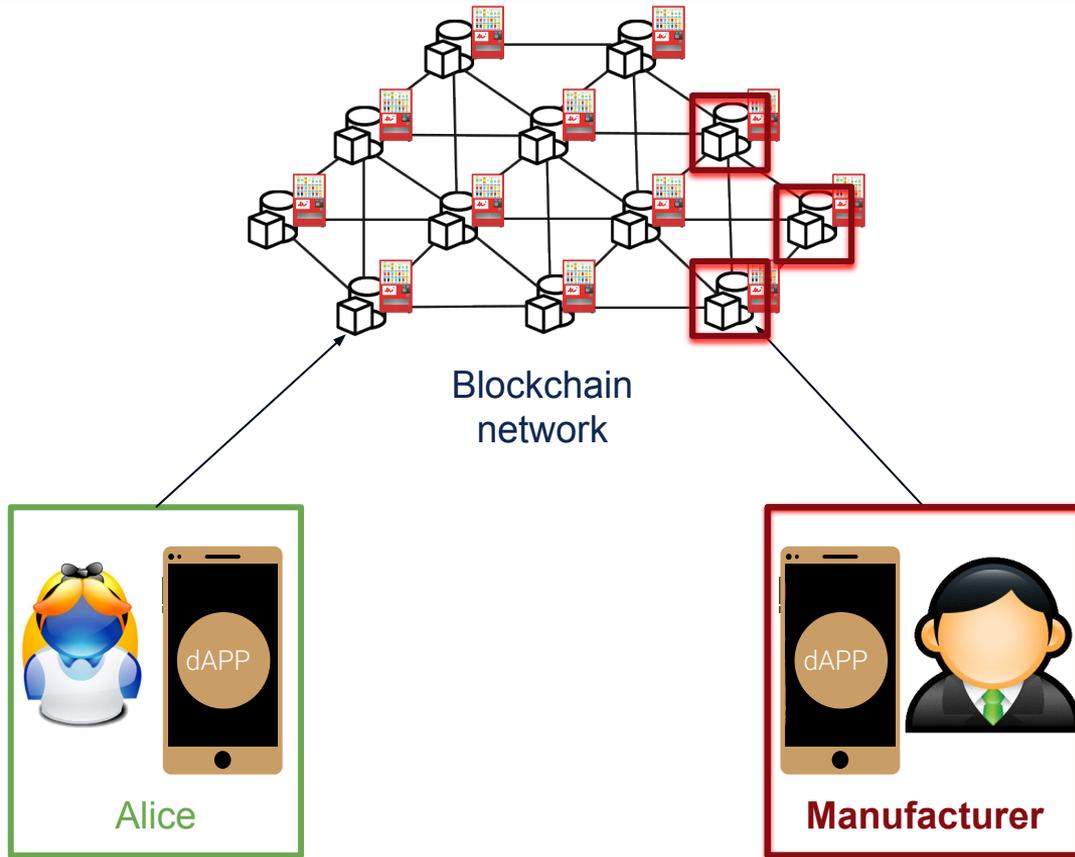


Fiducia contrattuale senza una terza parte fidata?

1. Se si dispone di N nodi indipendenti in una rete e la **maggioranza** ($\frac{2}{3} + 1$) di essi **segue** lo stesso **"meccanismo di consenso"**
2. Se si ha fiducia nel meccanismo di consenso
 - a. si conosce il **codice sorgente** su cui è costruito → *open source*
 - b. se questo permette di verificare il **codice sorgente dello smart contract** → *immutabilità dei dati*

allora

gli smart contracts scoraggiano i comportamenti opportunistici **impedendo deviazioni di natura tecnologica dall'accordo iniziale**



Esempio pratico
del perché
fidarsi di uno
smart contract

Gli smart contract non
possono essere
cambiati o rimossi
unilateralmente.

Questo limite alle
azioni unilaterali
rafforza la fiducia.



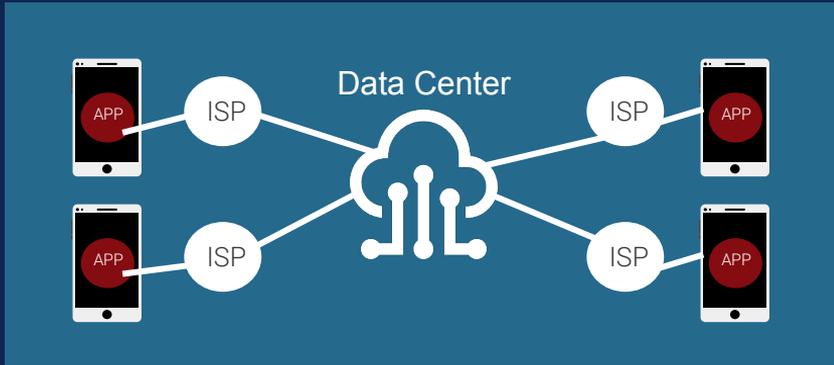
Caratteristiche principali della specie Smart Contract

1. Funzionamento
2. Immutabilità
3. Varietà della specie
4. Interazioni tra varietà e con il mondo esterno

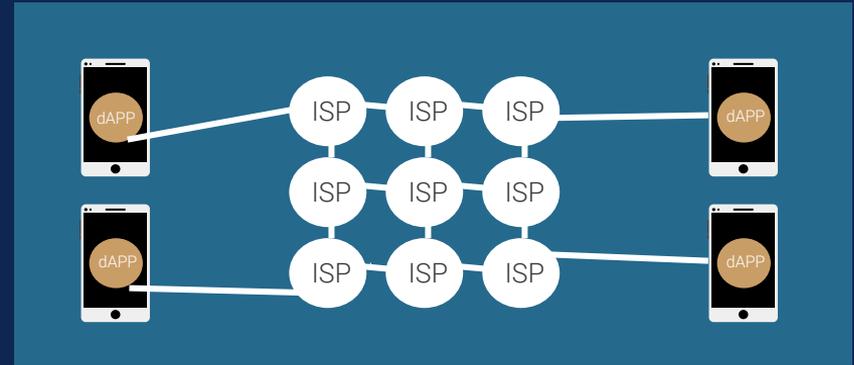


1. Funzionamento

+ Computazione Decentralizzata



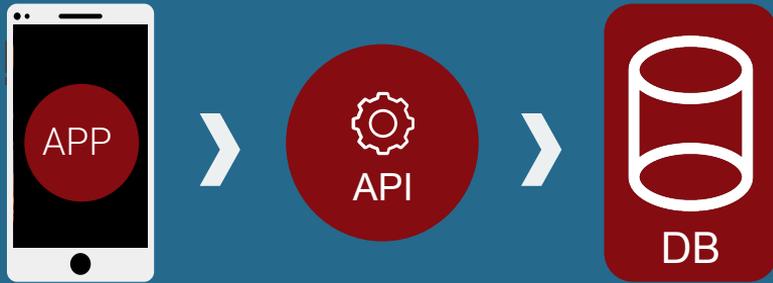
Le app tradizionali fanno richieste che vengono elaborate da uno o “pochi” server



Le dApp (decentralized App) fanno richieste che vengono elaborate da tutti i nodi della rete blockchain

+ Decentralized Applications

Sono interfacce rivolte all'utente finale che lo collegano alla tecnologia blockchain attraverso una combinazione di Smart Contracts sottostanti



Il rapporto tra dApp, Smart Contracts e Blockchain è simile alle applicazioni web tradizionali. Le app client/server interagiscono con un particolare server per accedere al suo database.

Analogamente, le dApp utilizzano gli Smart Contracts per connettersi alla particolare Blockchain su cui si basano (ad es. Ethereum).



Ethereum Smart Contracts

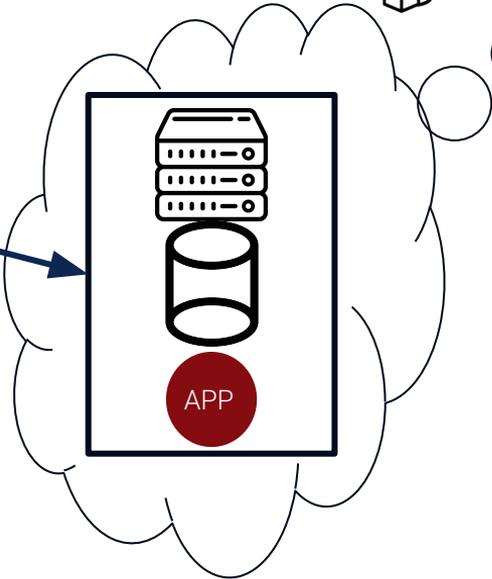
- Permette di mantenere facilmente delle strutture dati nella blockchain
- Una nuova transazione si riferisce ad una precedente e aggiorna lo stato del sistema
 - In questo caso lo stato del sistema considera non solo le transazioni monetarie, ma anche le strutture dei dati negli smart contracts
 - La transazione precedente si riferisce ad una che mantiene il codice e lo stato dello smart contract
 - La nuova transazione indica un insieme di istruzioni da eseguire nel contratto

Struttura di Ethereum

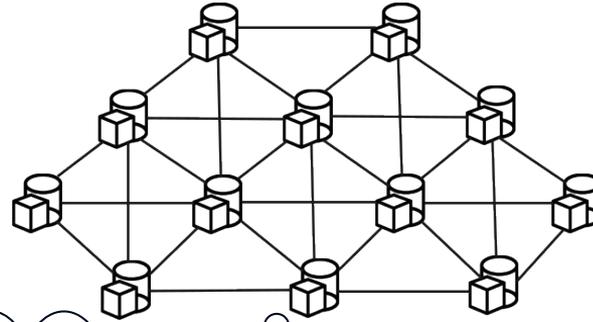


User

= **Wallet**



Peer Node = **Miner**



Interfaccia **nodi**
blockchain



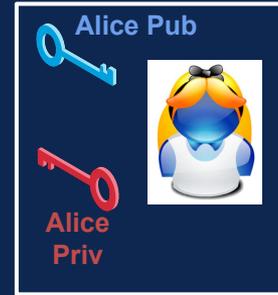
Interfaccia **utenti**



Wallet

applicazione che contiene uno o più Accounts e che permette di inviare o ricevere informazioni al/dal sistema

- **Account** -> modifica lo stato del sistema
 - **Externally Owned Account (EOAs)** ----->
 - **Contract Account (CAs)**
- **Messaggi e Transazioni** -> permettono di scambiare dati
 - Le transazioni sono definite come **pacchetti di dati firmati e inviati da un EOA**
 - I messaggi vengono scambiati solo **internamente al sistema tra CAs**



+ Ethereum Virtual Machine (EVM)

maggiore differenza con Bitcoin

- **Linguaggio Turing Completo**

- Ogni sistema o linguaggio di programmazione in grado di calcolare qualsiasi cosa calcolabile, date sufficienti risorse, è detto Turing completo
- La EVM permette di **scrivere regole ed eseguire programmi *quasi-Turing-completi*** -> Smart Contracts

- **GAS**

- Il *quasi* si riferisce al fatto che ogni passo di computazione nella EVM ha un costo -> il GAS è l'unità di misura
- Ogni transazione deve includere un valore di GAS che indichi il limite di gas che l'esecuzione può raggiungere
- Il GAS utilizzato viene moltiplicato per un certo GASPRICE ed il risultato viene trasferito ai Miners come tassa per aver eseguito la computazione



2. Immutabilità

2. Immutabilità

Gli smart contract inseriti in una blockchain si dice che siano immutabili di default.

Il codice sorgente (bytecode) di uno smart contract è infatti registrato in una transazione che viene “minata” in un blocco insieme ad altre transazioni:

TX 1 : 5 btc |--> Charlie Pub
TX 2 : TX 1 |--> Bob Pub
TX 3 : TX 2 |--> Alice Pub

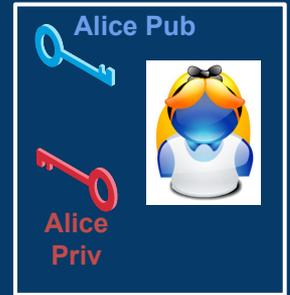
+ TX 4 :

Bytecode

|-->

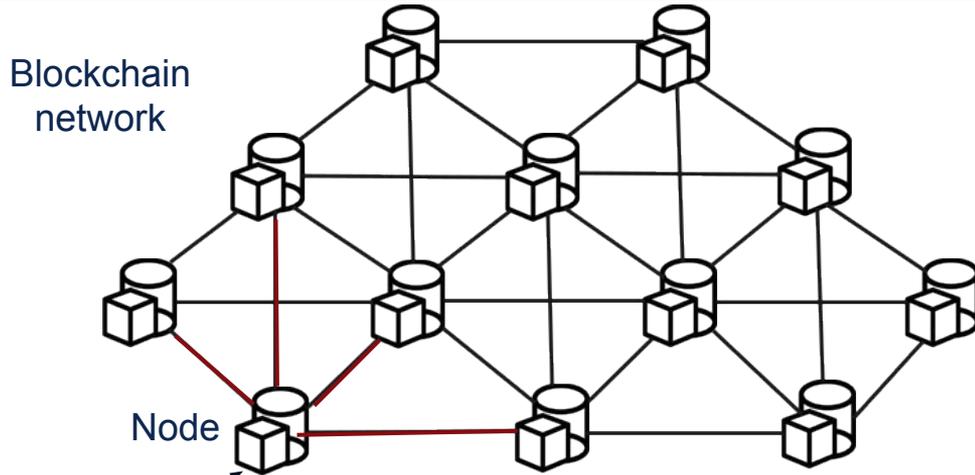
Indirizzo
Contratto

sign(TX 4, )
Alice
Priv



Alice's
Wallet

Esempio: Un'operazione di voto in uno Smart Contract

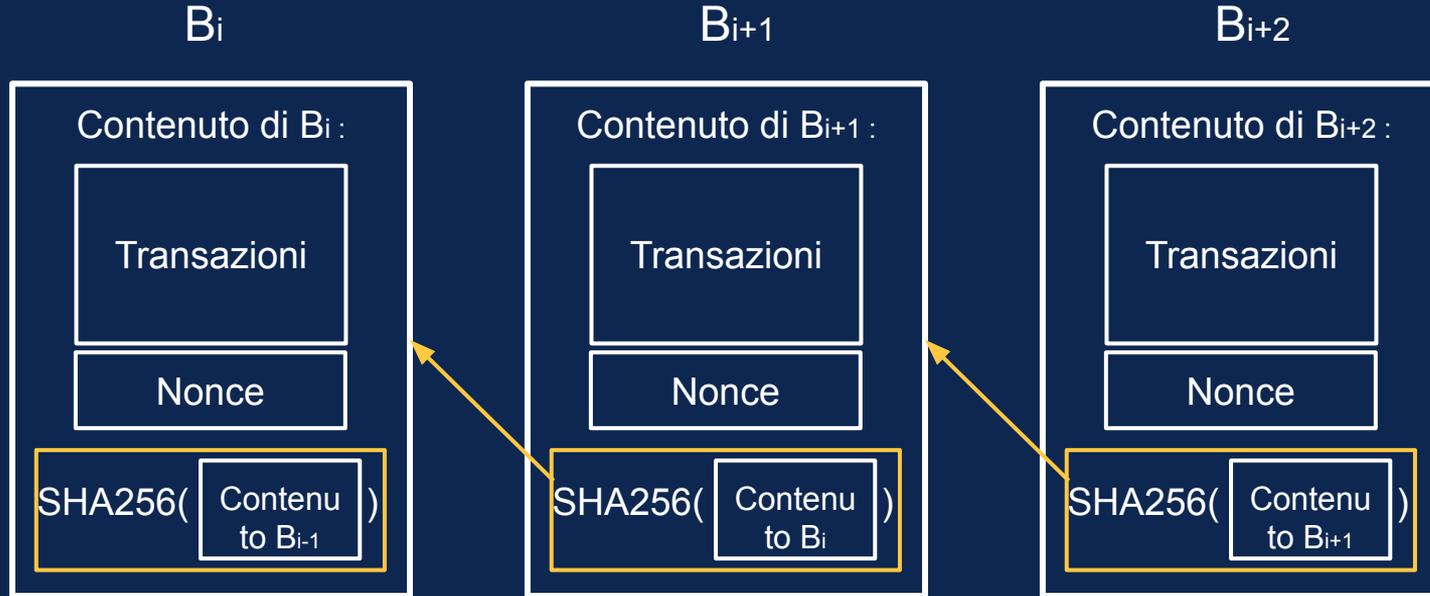


```
execute(  
  SmartContractAddress: VotingContract,  
  methodToExecute: vote,  
  parameters: challengeID, true  
)
```

Esempio: Un'operazione di voto in uno Smart Contract

```
function vote(Challenge challenge, bool inFavour) public {  
    if (inFavour) {  
        challenge.inFavour.add(msg.sender); // Alice  
    } else {  
        challenge.against.add(msg.sender); // Alice  
    }  
}
```

+ Struttura Blocchi

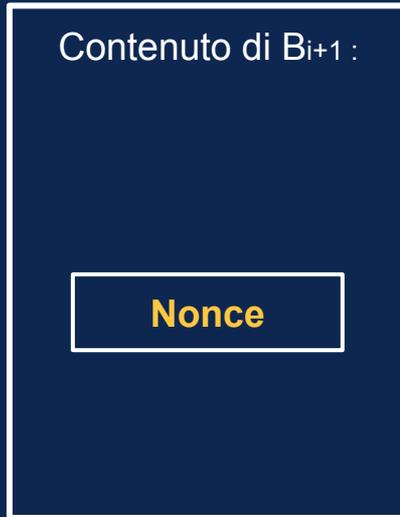


+ Struttura Blocchi

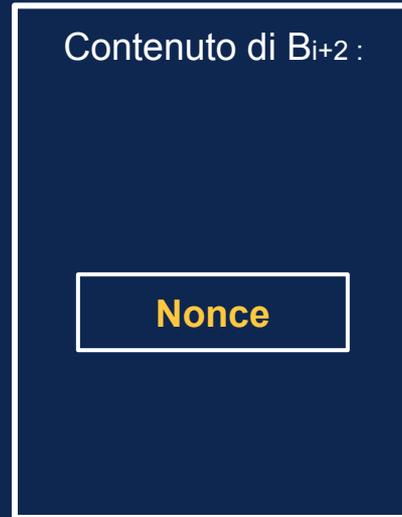
B_i



B_{i+1}



B_{i+2}



+ Mining

Risolvere un puzzle crittografico, es. trovare “l'ago in un pagliaio”



Puzzle:
questo digest deve iniziare con
9 zeri



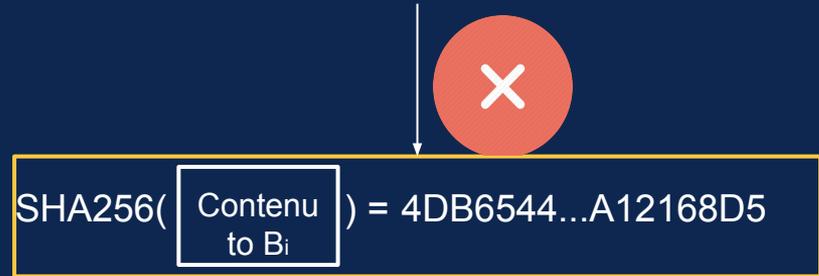
$$\text{SHA256(Contenu
to } B_i) = 5\text{AE3454B}\dots\text{9B8163F}$$

+ Mining

Risolvere un puzzle crittografico, es. trovare “l'ago in un pagliaio”



Puzzle:
questo digest deve iniziare con **9 zeri**



+ Mining

Risolvere un puzzle crittografico, es. trovare “l'ago in un pagliaio”

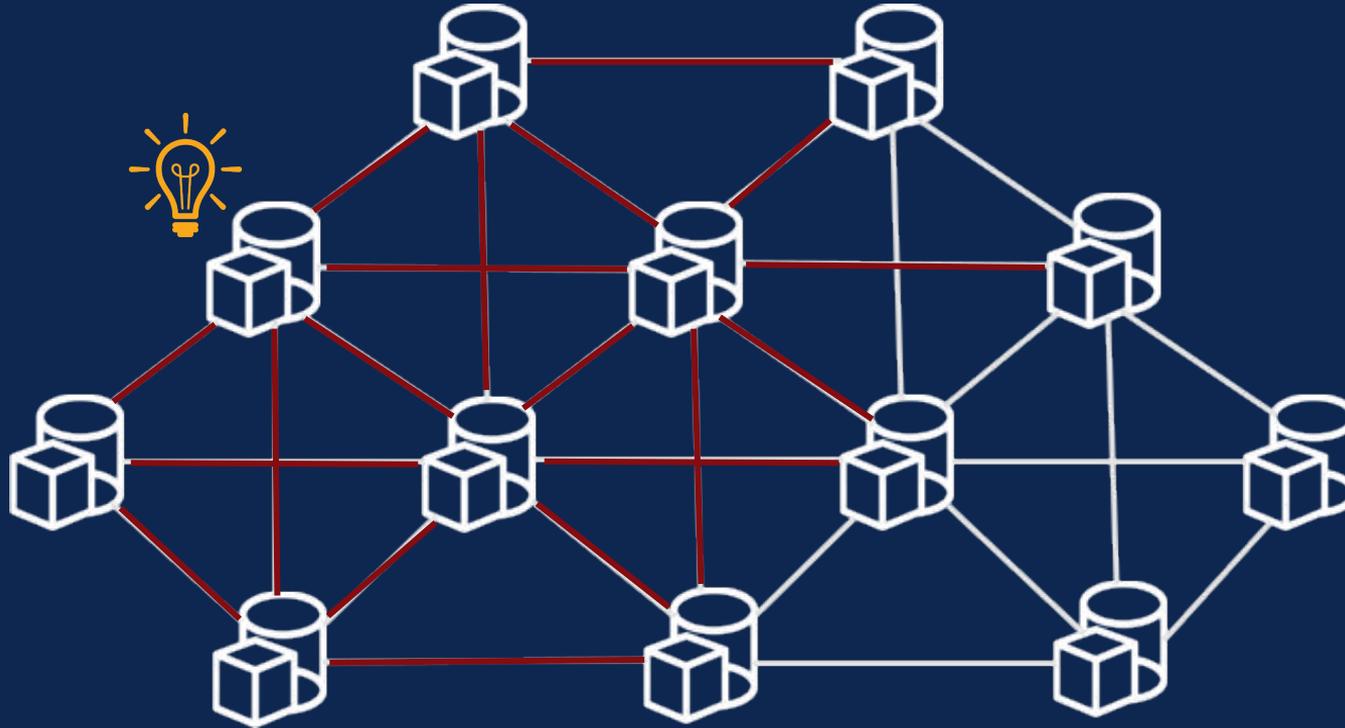


Puzzle:
questo digest deve iniziare con
9 zeri



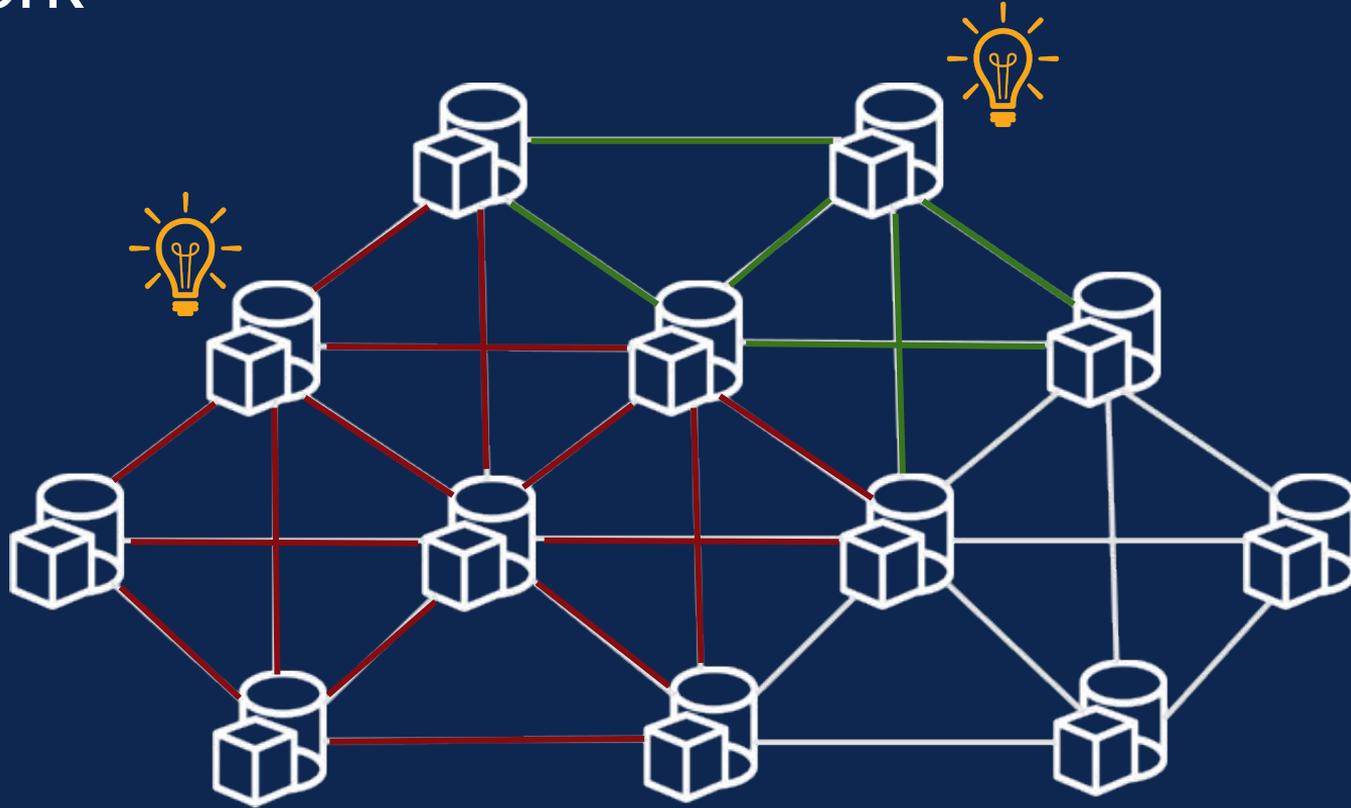
$$\text{SHA256(Contenu to } B_i) = 000000000\dots5BB589$$

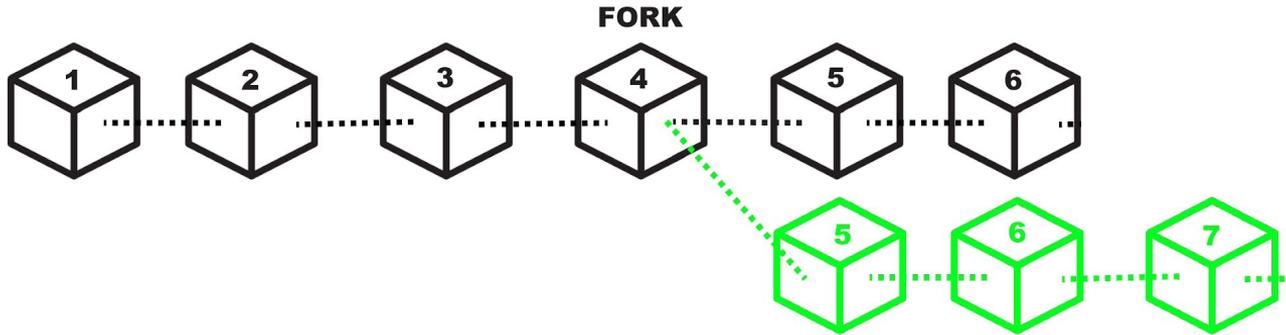
+ Proof of Work



1. Nodo risolve PoW
2. Trasmette il blocco ai vicini
3. Tutti i nodi controllano che:
 - Il PoW è valido
 - Il contenuto del blocco è valido
4. Quindi lo trasmettono ai loro vicini

+ Fork





Il **fork accidentale** si verifica quando:

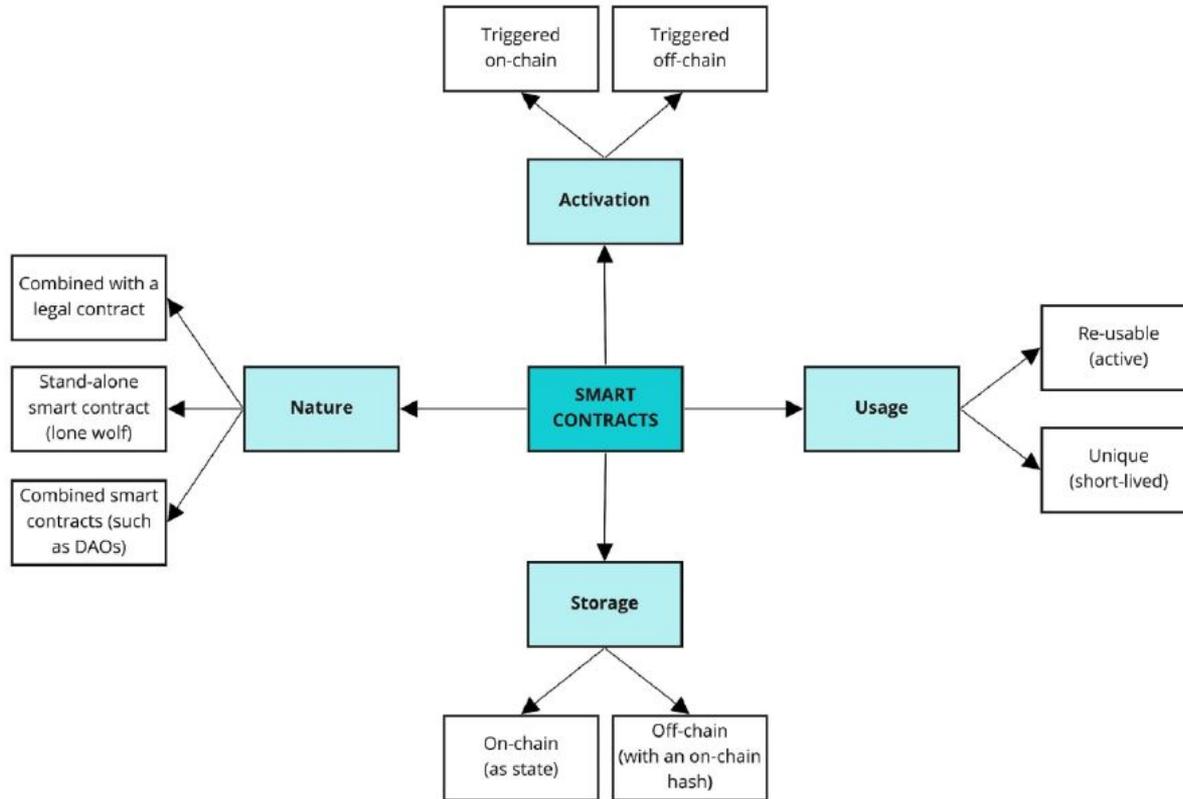
- due o più minatori trovano un blocco quasi nello stesso momento

Si risolve quando vengono aggiunti uno o più blocchi successivi e una delle catene diventa più lunga →

La rete abbandona i blocchi che non sono nella catena più lunga



3. Varietà della specie



Title: An overview of blockchain smart contracts
© Thibault Schrepel (2021)



Varietà: **Natura**

1. **Combinare uno smart contract con un "contratto legale"**
 - a. es., un contratto di affitto potrebbe essere scritto in prosa tra il proprietario di un appartamento e un inquilino, mentre lo smart contract potrebbe automatizzare il pagamento.

2. **Smart contract senza il supporto di un contratto legale**
 - a. maggior parte degli smart contract in circolazione oggi
 - b. "lupi solitari" → perché intendono essere autosufficienti.

3. **Smart contracts combinati con altri smart contracts**
 - a. creano le condizioni per la governance decentralizzata di ecosistemi
 - b. Decentralized Autonomous Organizations (**DAOs**).



Varietà: **Uso**

1. Smart contract condizionato a eventi della vita reale che **solo una delle due (o più) parti del contratto può invocare** -> *intuitu personae*
 - a. le condizioni per invocare uno smart contract sono specifiche per una singola parte

2. Smart contract invocato regolarmente, sia da una sola parte che da una **moltitudine di parti**
 - a. qualsiasi utente può invocarli
 - b. detto “attivo”



Varietà: **Attivazione**

1. Attivati **on-chain**: vengono invocati in seguito ad un evento della blockchain
 - a. es., uno smart contract può essere progettato per essere invocato solo quando il valore di uno asset presente nella blockchain supera un certo livello
2. Attivati **off-chain**: vengono invocati in seguito a un evento esterno alla blockchain
 - a. oracoli



Varietà: **Conservazione**

1. **On-chain:** il bytecode di uno smart contract è memorizzato su una transazione messa on-chain
 - a. facendo così si assicura l'immutabilità ma anche una mancanza di segretezza
2. **Off-chain:** i dati (compreso il bytecode) possono anche essere memorizzati off-chain, con solo solo l'hash che viene registrato sulla blockchain
 - a. l'immutabilità dello smart contract rimane di fatto garantita perché cambiandolo si genera automaticamente un nuovo valore di hash che non corrisponde a quello originale registrato sulla blockchain.



4. Interazioni tra varietà e con il mondo esterno



Interazioni tra varietà

Inter-blockchain

- gli smart contracts interagiscono tra loro, sia per competere che per cooperare
- diverse blockchain sono in competizione per gli smart contracts e, a seconda della tecnologia su cui sono costruiti, hanno caratteristiche uniche

Esempi:

- Gli smart contracts di *Polkadot*, *Cardano* e *EOS* sono, in media, convalidati più rapidamente di *Ethereum*
- *Tezos* permette più segretezza
- *Polkadot* utilizza dei bridge per consentire il trasferimento di token o dati da una blockchain ad un'altra

Intra-blockchain

- c'è anche concorrenza e cooperazione tra gli smart contract costruiti sulla stessa blockchain
- alcuni diventano più appetibili di altri perché sono meglio progettati, introducono nuove funzioni, o sono più supportati

Esempi:

- Uniswap 1, 2, 3
 - Gli smart contracts cooperano quando sono tecnicamente collegati tra loro.
- Ad esempio molti smart contracts trasferiscono automaticamente lo stesso tipo di ERC20 Token

+ Interazioni con il mondo esterno

Gli oracoli permettono agli smart contracts di **interagire con il mondo esterno**

- In origine, un oracolo era una persona incaricata di riferire la profezia sussurrata da fonti divine
- Per quanto riguarda la blockchain, generalmente, designa l'intermediario che riporta le informazioni dal mondo reale alla blockchain o viceversa
- In alternativa, l'oracolo può avere una funzione computazionale quando esegue calcoli off-chain



Varietà degli Oracoli: **Direzione, Raccolta Dati, Fonti**

1. Le informazioni possono prendere due **direzioni**:
 - 1.1. *outbound*, le informazioni dalla blockchain sono portate al mondo esterno
 - 1.2. *inbound*, si portano informazioni all'interno della blockchain.

2. Quando *inbound*, si distinguono diversi modi di **raccogliere informazioni**:
 - 2.1. *software*, interagisce con (esistenti) informazioni online e poi le trasmette
 - 2.2. *hardware*, trasforma le misure del mondo reale in informazioni digitali
 - 2.3. *human*, terza parte fidata che fornisce informazioni del mondo reale.

3. L'oracolo può usare una sola **fonte** o diverse di esse:
 - 3.1. *singola fonte*, ``ricentralizza'' la blockchain introducendo un **singolo punto di fallimento** e richiedendo la fiducia in un solo punto di ingresso
 - 3.2. *combinazione di diverse fonti*, è preferibile ma richiede tuttavia regole di governance ben progettate.



Varietà degli Oracoli: **Validazione, Integrazione, Uso**

4. Si deve poi **convalidare** l'informazione una volta trasmessa:
 - 4.1. *automatica*, se l'utente decide di fidarsi dell'oracolo
 - 4.2. *voto*, oggetto di un voto sottoposto agli utenti della blockchain (DAO).

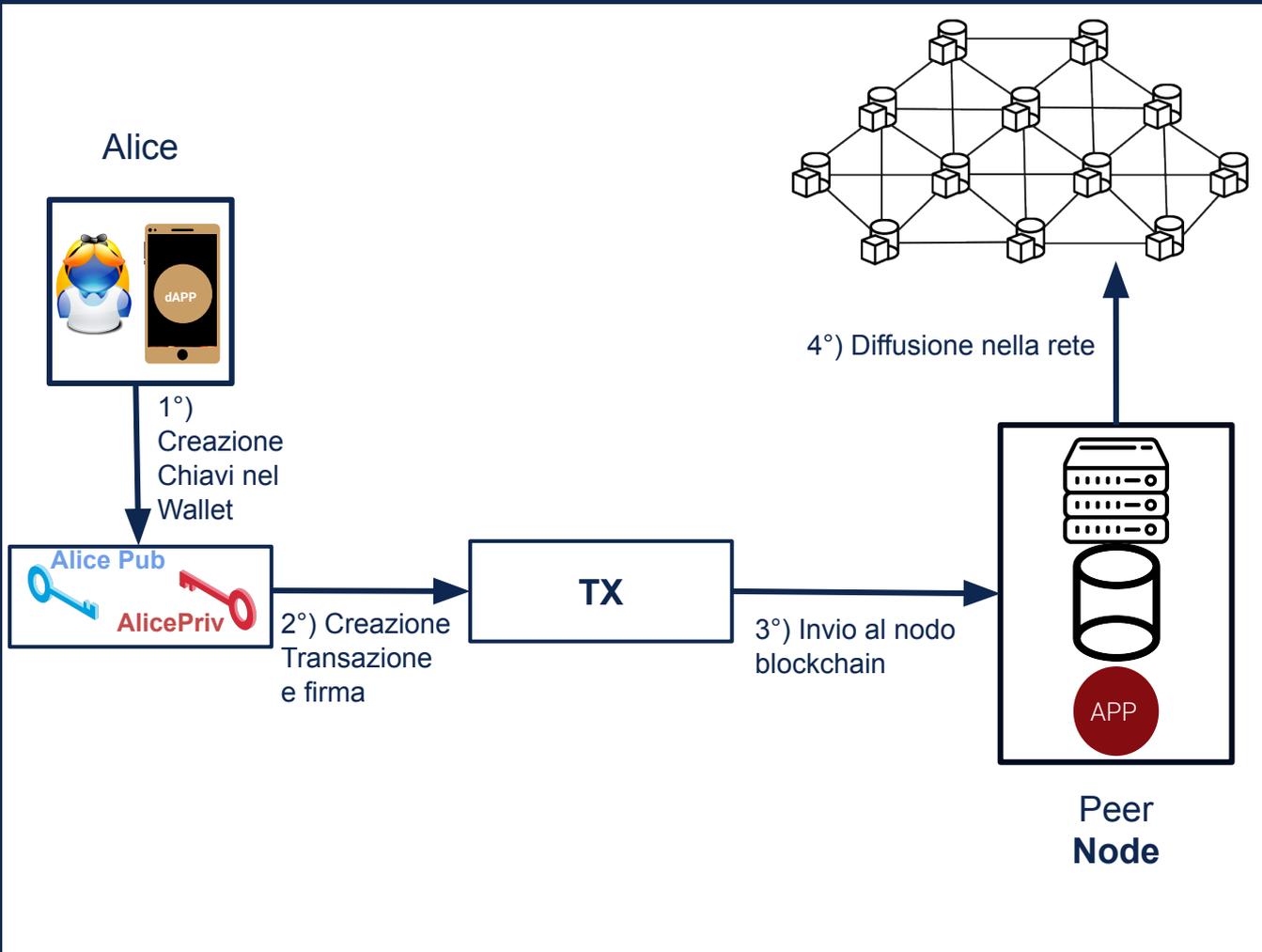
5. Le informazioni devono essere **integrate**:
 - 5.1. *senza intermediari*, direttamente distribuite alla rete blockchain
 - 5.2. *interfaccia di uno smart contract personalizzata*, e.g. dApp
 - 5.3. *modulo software per il data pre-processing*
 - 5.4. *soluzione personalizzata*, per prevenire la falsificazione e.g. fingerprint

6. Una volta che l'informazione è integrata, i suoi **usi** possono essere:
 - 6.1. *contract-specific*, impiego in un singolo smart contract
 - 6.2. *multiple smart contracts use*, come un database es. dati finanziari



Esempio pratico di uno Smart Contract

Esempio creazione transazione



Interfaccia nodi blockchain



Interfaccia utenti



1°) Creazione Chiavi nel Wallet

- Per inserire transazioni nella blockchain bisogna autenticarsi (firma digitale)
 - Se non si possiede già, serve una coppia di chiavi asimmetriche
 - Chiave Pubblica + Chiave Privata
- In Ethereum -> **Elliptic Curve Digital Signature Algorithm (ECDSA)**
 - La chiave privata consiste in un valore segreto generato da un processo di randomizzazione
 - La chiave pubblica deriva da quella privata
- L'Indirizzo (Address) di un Account si ottiene dalla chiave pubblica

1°) Creazione Chiavi nel Wallet



Private Key (256 bits):

```
'0xdc47ca238ffb638cf76658fb02a351bc7c1c6b  
bb32fa40db9bb43fee47c9dfbd'
```

↓
ECDSA secp256k1

Public Key (x,y point – 512 bits - Two 32-bit integers):

```
'b9f5d91099422bcfa991abe2866f3dc39bba8da  
50c52b77179eca74ecdaefd06cebbb2987f223c  
e8d1585d899999948b969b0039e3c36f14b297  
3cf20ed96330'
```

↓
Keccak-256
(first 40 bytes - 160 bits)

Ethereum address:

```
'0x39532829E35c3238cd0bc1613F7e586Cb106  
46CC'
```

<https://asecuritysite.com/encryption/ethadd>

+ 2°) Creazione Transazione e firma

Struttura Transazione

- **Destinatario** - l'indirizzo del destinatario (EOA oppure CA)
- **Firma** - la firma digitale del mittente
- **Valore** - importo in Ether da trasferire dal mittente al destinatario
- **Dati** - campo opzionale per includere dati arbitrari (per gli Smart Contracts)
- **gasLimit** - la quantità massima di unità di gas che può essere consumata
- **gasPrice** - la tassa che il mittente paga per unità di gas

2°) Creazione Transazione e firma

```
transaction = {  
  nonce: web3.toHex(0),  
  gasPrice: web3.toHex(200000000000),  
  gasLimit: web3.toHex(100000),  
  to: '0x687422eEA2cB73B5d3e242bA5456b782919AFc85',  
  value: web3.toHex(1000),  
  data: '0xc0de'  
}
```

rlp + hash



0x6a74f15f29c3227c5d1d2e27894da58d417a484ef53bc7aa57ee323b42ded656

sign with privateKey



```
v: '0x1c'  
r: '0x668ed6500efd75df7cb9c9b9d8152292a75453ec2d11030b0eec42f6a7ace602'  
s: '0x3efcbbf4d53e0dfa4fde5c6d9a73221418652abc66dff7fddd78b81cc28b9fbf'  
signature
```

https://miro.medium.com/max/1142/1*4Ta0bUfCEmgH4kOrNI8mKg.png



Inizio parte interattiva

- 1. Smart Contracts Lego**
<http://etherscripter.com>
- 2. Installazione wallet (Metamask)**
<https://metamask.io/download.html>
- 3. Acquire Ether (Faucet)**
<https://faucet.metamask.io/>
<https://faucet.dimensions.network/>
- 4. Interazione con smart contract (ERC-721)**
<https://intelligible-demo.herokuapp.com/>