

Law, Science and Technology
MSCA ITN EJD n. 814177



Mirko Zichichi

Smart contracts

Outline

- Introduzione agli Smart Contracts (~1.5 ore)
 - Installazione wallets
- Introduzione alla programmazione in Solidity (~1.5 ore)
 - Smart contracts lego
 - Cryptozombies
- Introduzione ai token in Ethereum (~0.5 ore)
 - ERC20
 - NFT
- Interazione dapps Ethereum (~2 ore)
 - KLEROS
 - Remix

Introduzione agli Smart Contracts

+ 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 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

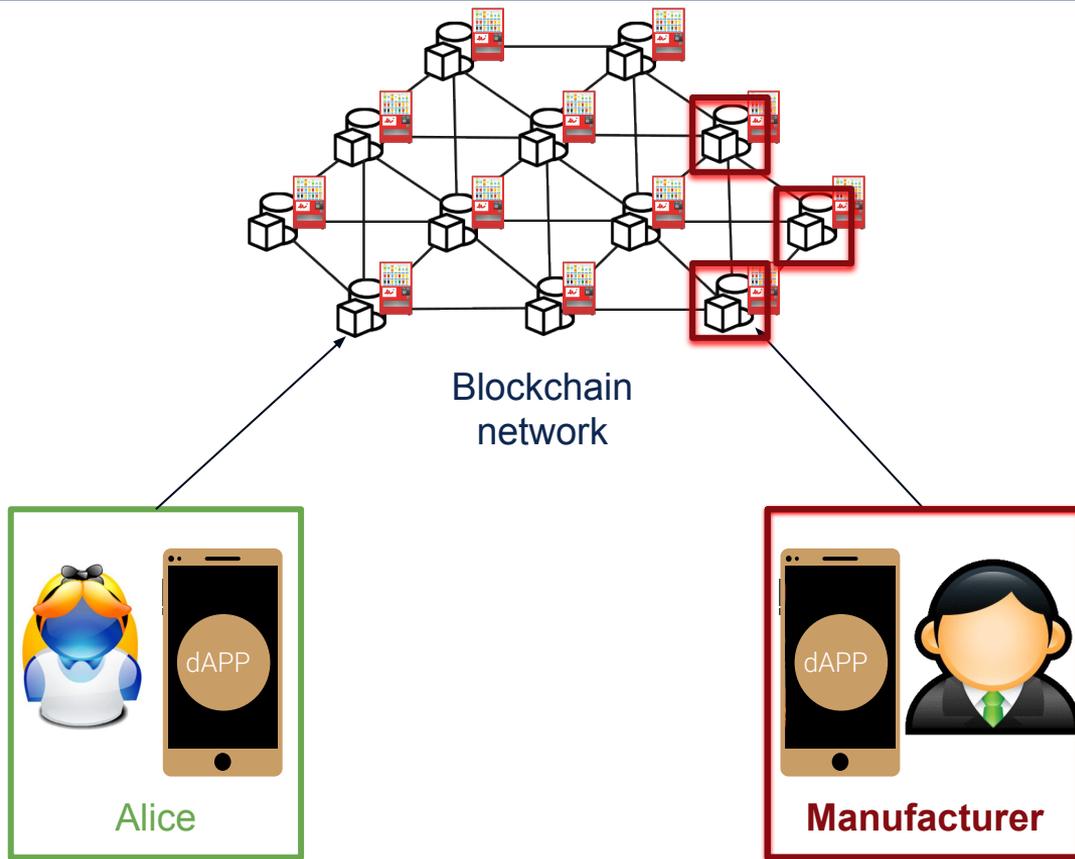


Fiducia contrattuale senza una terza parte fidata?

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.



Ethereum

- Ethereum può essere considerato come il tentativo di **decentralizzare l'accesso alle informazioni su Internet**
 - ma soprattutto come il riconoscimento dei limiti della rete **Bitcoin**, la prima ad utilizzare la blockchain.
 - L'intento è quello di creare un protocollo alternativo per lo sviluppo di **applicazioni decentralizzate** attraverso una Blockchain.
- Progetto open-source introdotto da → *Vitalik Buterin*
 - Nessuno “controlla” Ethereum, tuttavia esiste un'entità, la Ethereum Foundation, che controlla lo sviluppo del software progetto.



Ethereum

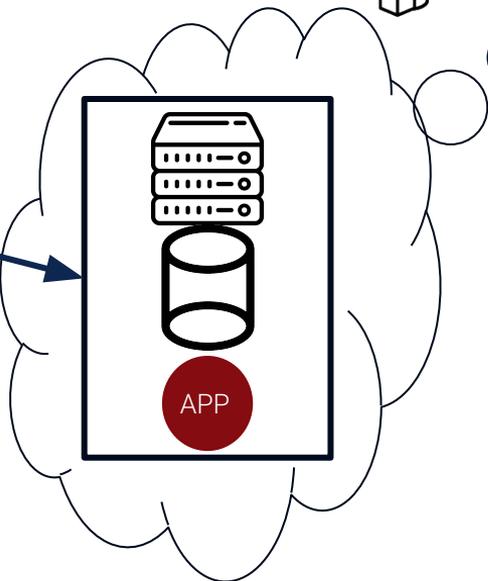
- Ethereum è un sistema composto da:
 - un network di **validatori** (nodi della blockchain)
 - un **algoritmo di consenso** (PoW, PoS)
 - un **registro condiviso** (blockchain)
 - un sistema di **indirizzi** (address e wallets)
 - una **computer decentralizzato** (macchina virtuale)
 - un insieme di **linguaggi di programmazione**
 - una **struttura economica** complessa (cryptocurrency e tokens)

Network di Validatori

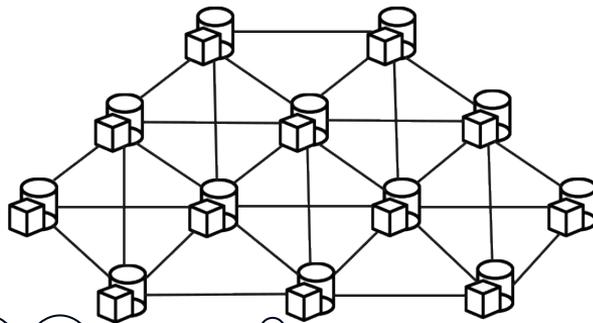


User

= **Wallet**



Peer Node = **Miner**

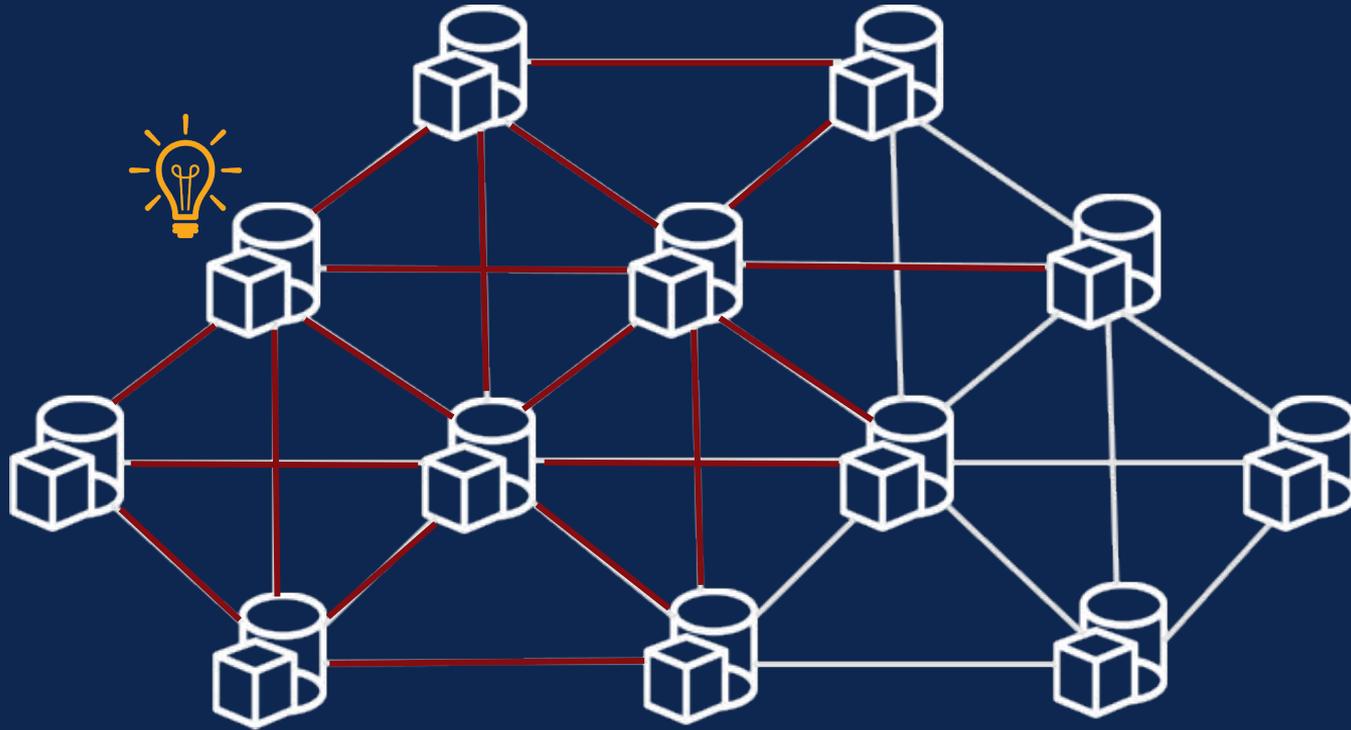


Interfaccia **nodi** blockchain



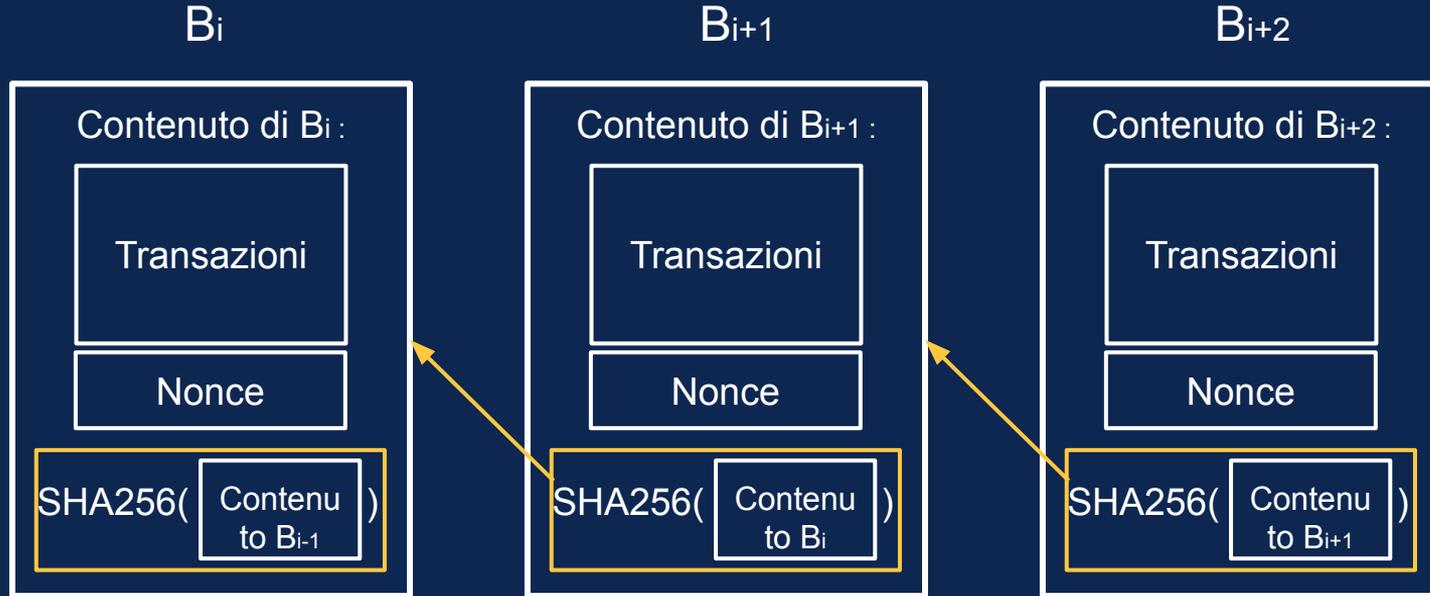
Interfaccia **utenti**

+ Algoritmo di consenso (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

+ Registro condiviso (Blockchain)





Ethereum

- Ethereum è un sistema composto da:
 - un network di **validatori** (nodi della blockchain)
 - un **algoritmo di consenso** (PoW, PoS)
 - un **registro condiviso** (blockchain)
 - un sistema di **indirizzi** (address e wallets)
 - una **computer decentralizzato** (macchina virtuale)
 - un insieme di **linguaggi di programmazione**
 - una **struttura economica** complessa (cryptocurrency e tokens)

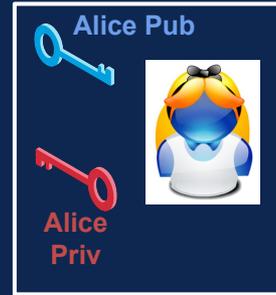




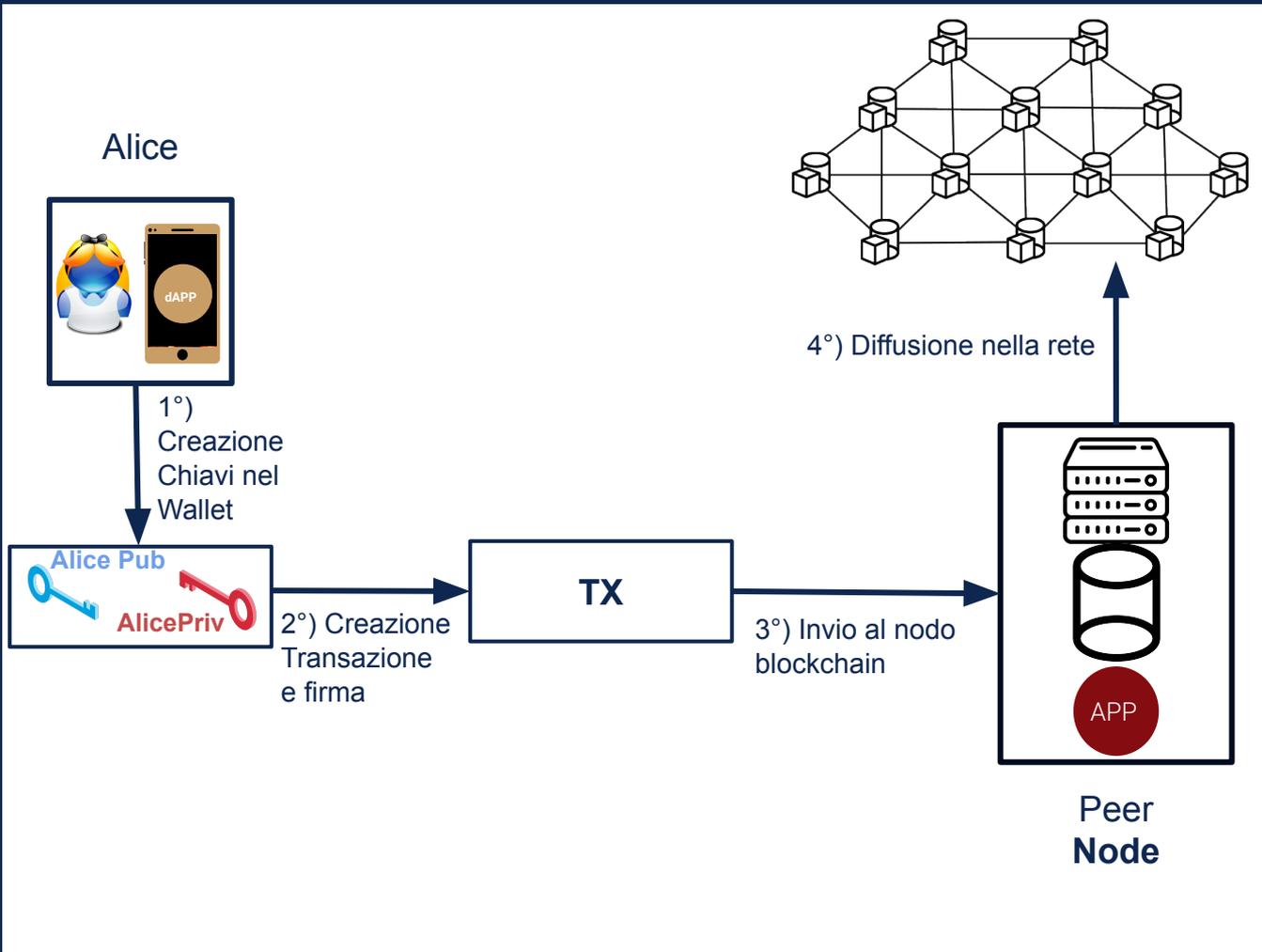
Wallet

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**



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. Installazione wallet (Metamask)**

<https://metamask.io/download.html>

- 2. Acquisire Ether (Faucet)**

<https://faucet.metamask.io/>

<https://faucet.dimensions.network/>

- 3. Leggere dalla Blockchain**

<https://etherscan.io/>

- 4. Interazione con la blockchain**

<https://intelligible-demo.herokuapp.com/>

Introduzione alla programmazione in Solidity

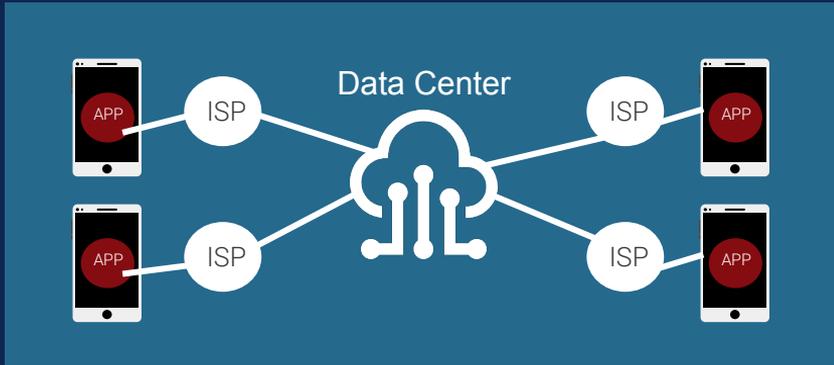


Ethereum

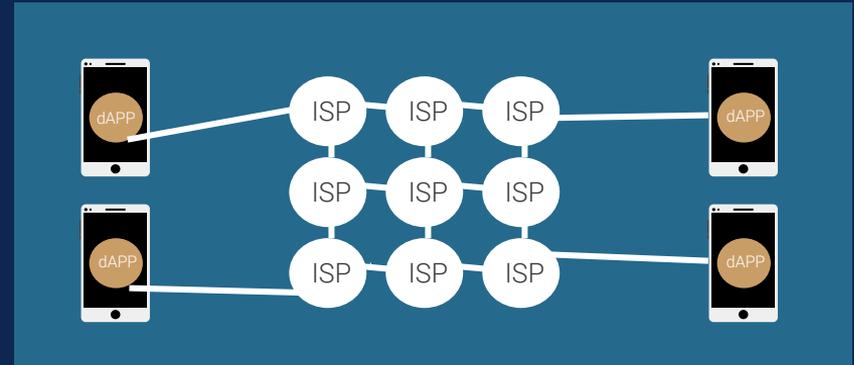
- Ethereum è un sistema composto da:
 - un network di **validatori** (nodi della blockchain)
 - un **algoritmo di consenso** (PoW, PoS)
 - un **registro condiviso** (blockchain)
 - un sistema di **indirizzi** (address e wallets)
 - una **computer decentralizzato** (macchina virtuale)
 - un insieme di **linguaggi di programmazione**
 - una **struttura economica** complessa (cryptocurrency e tokens)



+ Computazione Decentralizzata



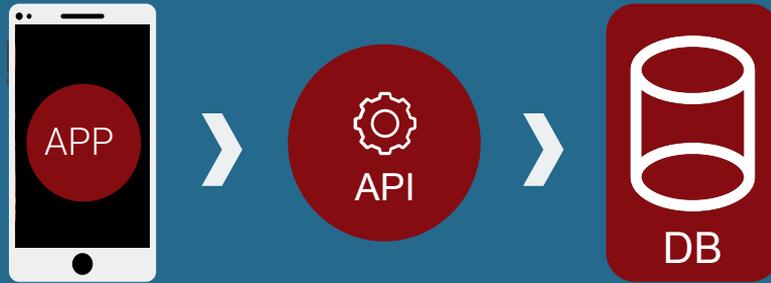
Le app tradizionali fanno richieste che vengono elaborate da uno o “pochi” server



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).

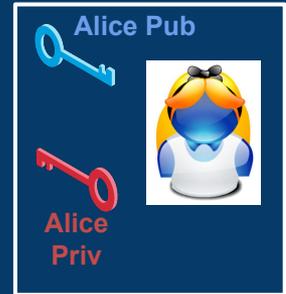
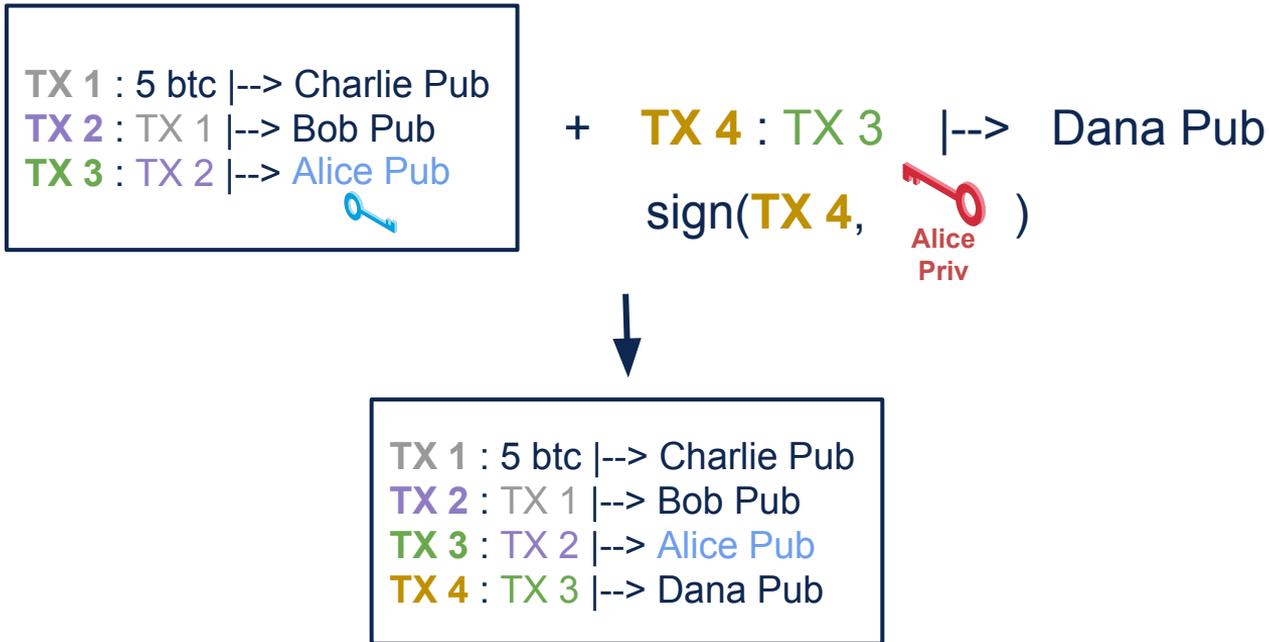


Ethereum Smart Contracts

- 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

(Transazioni blockchain)

Gli INPUT devono essere sbloccati con una firma digitale



Alice's
Wallet

Immutabilità

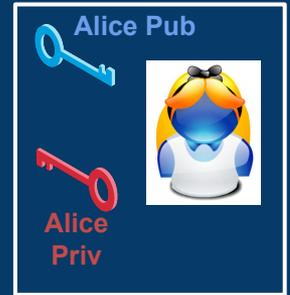
Gli smart contract inseriti in una blockchain si dice che siano immutabili di default.

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

+ TX 4 : **Bytecode** |--> Indirizzo Contratto

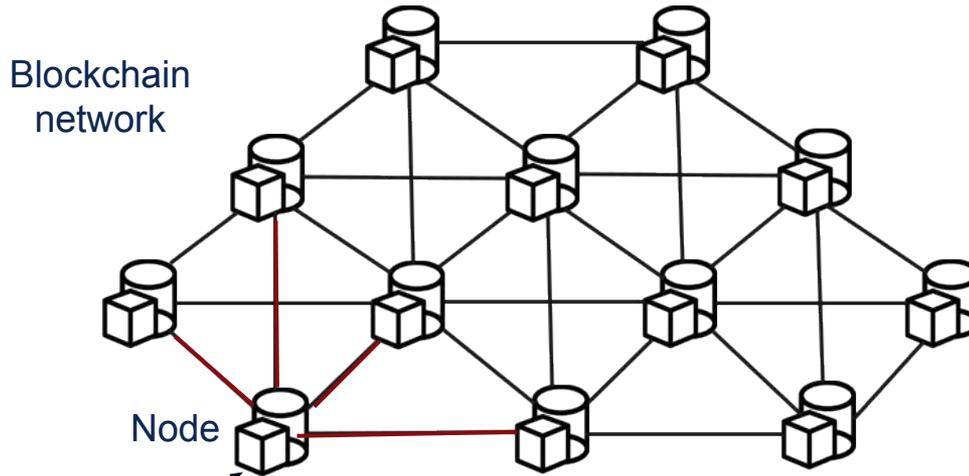
sign(TX 4,  Alice Priv)

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



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  
    }  
}
```

+ 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



Ethereum

- Ethereum è un sistema composto da:
 - un network di **validatori** (nodi della blockchain) ✓
 - un **algoritmo di consenso** (PoW, PoS) ✓
 - un **registro condiviso** (blockchain) ✓
 - un sistema di **indirizzi** (address e wallets) ✓
 - una **computer decentralizzato** (macchina virtuale) ✓
 - un insieme di **linguaggi di programmazione**
 - una **struttura economica** complessa (cryptocurrency e tokens)



Codice

1. Il codice è, nella sua forma più semplice, **un linguaggio usato per dare istruzioni** ai computer
2. Un **programma** per computer è un insieme di istruzioni scritte in codice ed eseguite da un computer.
3. Il processo di creazione di un programma si divide in:
 - a. **codice sorgente** → scrittura del codice in un linguaggio di programmazione di **"alto livello"**
 - b. **codice macchina** → la conversione finale del codice sorgente in un linguaggio di programmazione a **"basso livello"**



Linguaggi di programmazione di alto livello

Python, C++, Java, Javascript

Gli smart contract, come qualsiasi altro software, sono scritti in un linguaggio di programmazione di alto livello -> **Solidity***

Utilizzato dai programmatori per scrivere le istruzioni degli smart contracts, che poi vengono tradotte (*compile*) in linguaggio macchina -> **EVM**



Breve introduzione alla programmazione

1. Variabili

- a. Le variabili sono contenitore di dati situati in una porzione di memoria destinata a contenere valori.
- b. Sono identificate da un nome arbitrario (scelto dal programmatore).
- c. Sono suscettibili di modifica nel corso dell'esecuzione di un programma.
- d. Sono caratterizzate da un tipo.

2. Esempio

- a. **str** nome = "Moat"
- b. **int** anni = 24
- c. **float** peso = 54.3



Breve introduzione alla programmazione

3. Operatori

- a. Gli operatori sono simboli che specificano quale legge applicare a uno o più operandi, per generare un risultato.
- b. Ne esistono di vari tipi: aritmetici, booleani, assegnamento, decremento/incremento, condizionali, bitwise, di relazione.
- c. Funzionano... più o meno come ci si aspetta!

4. Esempio

- a.

```
int anni = 24
int anni_doppi = anni * 2
bool risultato = anni > anni_doppi
```
- b.

```
print(anni_doppi) >>> 48
```
- c.

```
print(risultato) >>> False
```



Breve introduzione alla programmazione

5. Strutture di controllo

- a. Le strutture di controllo sono dei costrutti sintattici che controllano il flusso di esecuzione di un programma.
- b. Specificano se, quando, in quale ordine e quante volte devono essere seguite determinate istruzioni.
- c. Le più note sono: if-then-else, while, for

6. Esempio

- a.

```
If (anni > 80) then {  
  bool vecchio = True  
} else {  
  bool vecchio = False  
}
```



Breve introduzione alla programmazione

7. Funzioni

- a. Le funzioni sono costrutti sintattici che permettono di raggruppare insieme una sequenza di istruzioni.
- b. Prendono uno o più valori in input e restituiscono uno o più valori in output.
- c. Sono identificate da un nome arbitrario scelto dal programmatore.
- d. Per essere eseguite, vanno “chiamate” o “invocate”.

8. Esempio

```
a. function somma(int a, int b) {  
    int c = a + b;  
    return c;  
}
```



Inizio parte interattiva 2

1. **Smart Contracts Lego**

<http://etherscripter.com>

2. **Lezioni crypto zombies**

<https://cryptozombies.io>

Introduzione ai token in Ethereum



Ethereum

- Ethereum è un sistema composto da:
 - un network di **validatori** (nodi della blockchain) ✓
 - un **algoritmo di consenso** (PoW, PoS) ✓
 - un **registro condiviso** (blockchain) ✓
 - un sistema di **indirizzi** (address e wallets) ✓
 - una **computer decentralizzato** (macchina virtuale) ✓
 - un insieme di **linguaggi di programmazione** ✓
 - una **struttura economica** complessa (cryptocurrency e tokens) ✓



Cryptocurrency

- Una **cryptocurrency** è un asset digitale utilizzato come mezzo di scambio di **valore**
 - impiega l'utilizzo della crittografia, i.e. blockchain, per rendere sicuri questi scambi.
- Spesso viene utilizzata all'interno della blockchain come incentivo
 - I validatori (o miners) eseguono il PoW e ricevono una certa quantità di cryptocurrency in cambio
- Esempi: Bitcoin, Ether, Monero, etc...

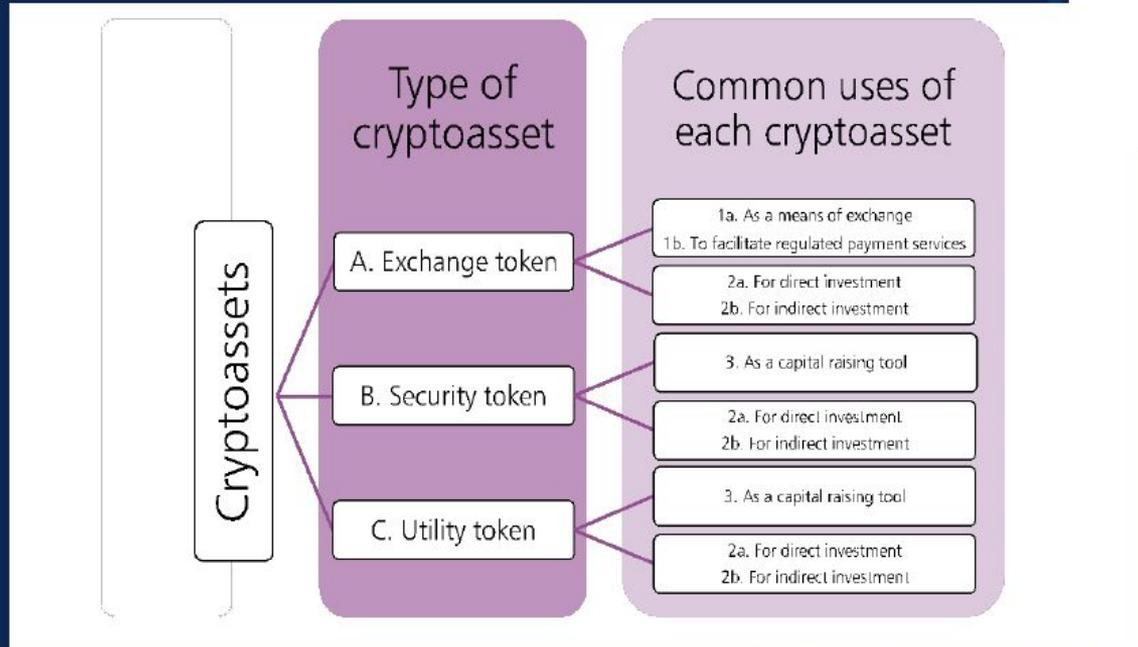


Distinzione tra coin e token

- Un coin è la **cryptocurrency** nativa di una blockchain/DLT
 - è l'asset usato dal protocollo della rete di nodi
 - di solito è solamente 1 per ogni blockchain
- Un token è un “crypto-asset” generato “on top of” la blockchain
 - rappresentazione di valore o di diritti contrattuali digitale, crittograficamente sicura, che utilizza un qualche tipo di DLT e che può essere trasferita, memorizzata o scambiata elettronicamente (FCA 2019)
 - Potenzialmente possono esserci infiniti token per ogni blockchain
 - Solitamente sono implementati usando gli smart contracts



TOKEN



Financial Conduct Authority, Bank of England. 2018

→ TOKEN IBRIDI

→ APPLICATION /
PLATFORM TOKENS

- Gli smart contract possono essere usati per creare e controllare *asset digitali* ;
- Quando un token viene creato per essere usato su una DApp, il suo scopo dipende dall'applicazione stessa



ERC-20

- L'ERC-20 introduce uno standard per i Token Fungibili.
- In altre parole, hanno la caratteristica per cui ogni Token sia esattamente lo stesso (per tipo e valore) di un altro Token.
- Per esempio, un Token ERC-20 agisce proprio come l'ETH, il che significa che 1 Token è e sarà sempre uguale a tutti gli altri Token



ERC-20

- function name() public view returns (string)
- function symbol() public view returns (string)
- function decimals() public view returns (uint8)
- function totalSupply() public view returns (uint256)
- function balanceOf(address _owner) public view returns (uint256 balance)
- function transfer(address _to, uint256 _value) public returns (bool success)
- function transferFrom(address _from, address _to, uint256 _value) public returns (bool success)
- function approve(address _spender, uint256 _value) public returns (bool success)
- function allowance(address _owner, address _spender) public view returns (uint256 remaining)
- event Transfer(address indexed _from, address indexed _to, uint256 _value)
- event Approval(address indexed _owner, address indexed _spender, uint256 _value)



ERC-20 Esempi

- <https://etherscan.io/tokens>
- <https://app.uniswap.org/#/swap?chain=ropsten>



ERC-721 -> Non Fungible Tokens

- Sono Token Non Fungibili
- Un Non-Fungible Token (NFT) viene utilizzato per identificare qualcosa o qualcuno in un modo unico.
- Questo tipo di Token è adatto per essere utilizzato su piattaforme che offrono oggetti da collezione, chiavi di accesso, biglietti della lotteria, posti a sedere numerati per concerti e partite sportive, ecc.
- Può anche essere utilizzato per rappresentare beni reali.



ERC-721

- function `balanceOf(address _owner)` external view returns (uint256);
- function `ownerOf(uint256 _tokenId)` external view returns (address);
- function `safeTransferFrom(address _from, address _to, uint256 _tokenId, bytes data)` external payable;
- function `safeTransferFrom(address _from, address _to, uint256 _tokenId)` external payable;
- function `transferFrom(address _from, address _to, uint256 _tokenId)` external payable;
- function `approve(address _approved, uint256 _tokenId)` external payable;
- function `setApprovalForAll(address _operator, bool _approved)` external;
- function `getApproved(uint256 _tokenId)` external view returns (address);
- function `isApprovedForAll(address _owner, address _operator)` external view returns (bool);
- event `Transfer(address indexed _from, address indexed _to, uint256 indexed _tokenId)`;
- event `Approval(address indexed _owner, address indexed _approved, uint256 indexed _tokenId)`;
- event `ApprovalForAll(address indexed _owner, address indexed _operator, bool _approved)`;



ERC-721 Esempi

- <https://opensea.io/>
- <https://testnets.opensea.io>
- <https://faucet.rinkeby.io/>
- <https://faucets.chain.link/>



Ethereum

- Ethereum è un sistema composto da:
 - un network di **validatori** (nodi della blockchain) ✓
 - un **algoritmo di consenso** (PoW, PoS) ✓
 - un **registro condiviso** (blockchain) ✓
 - un sistema di **indirizzi** (address e wallets) ✓
 - una **computer decentralizzato** (macchina virtuale) ✓
 - un insieme di **linguaggi di programmazione** ✓
 - una **struttura economica** complessa (cryptocurrency e tokens) ✓

Interazione dapps Ethereum



Inizio parte interattiva 3

1. **KLEROS**

- a. https://www.youtube.com/watch?v=NuSps_2wMQ4&t=7s
- b. <https://www.ibanet.org/lex-cryptographia-due-process-blockchain-based-arbitration>
- c. <https://kleros.io/>
- d. <https://kleros.gitbook.io/docs/products/court/kleros-juror-tutorial>
- e. <https://ropsten.etherscan.io/address/0x9AdCEAa6CFd7182b838Beb085e97729EB1Da681E#writeContract>
- f. <https://court.kleros.io/>
- g. <https://court.kleros.io/cases/19>

2. **Remix**

<https://remix.ethereum.org/>

3. **Utils**

<https://profitplane.com/str-to-bytes32>