CLASS TEST
DECENTRALISED TECHNOLOGIES AND CRYPTOCURRENCIES

MAY 23RD 2022

Instructions

- Test can be taken in English or Italian
- Fill in below with your name, surname, and student number
- Answer concisely to questions. Do not use fonts smaller that 11 pt. Do not exceed 4 pages for your answers
- This test lasts 60 min
- Please, save your word file regularly
- The exam test will be collected at the end of the test. Update the file name with your student number

Questions

1. Give a definition of decentralised ledger. Explain how it is used in blockchain protocols and why it is different from a standard database.

2. List and explain three (3) properties of the distributed ledger, making an example of application for each of them.

3. Explain what are Elliptic Curves and how they are used in Bitcoin.

4. Explain the following concepts: smart contract, Dapp, and wallet. Then describe a use case that uses at least two of them.

5. In Remix, modify the code below so that the sender function will be able to transfer tokens to three accounts at the same time instead of one, splitting the amount into three. Please implement required changes, compile, deploy and test it with Remix. Report a screenshot of the new function. Explain also your modification to the code.

CODE

```
pragma solidity ^0.5.0;

contract Coin {
    // The keyword "public" makes those variables
    // easily readable from outside.
    address public minter;
   mapping (address => uint) public balances;

    // Events allow light clients to react to
    // changes efficiently.
    event Sent(address from, address to, uint amount);

    // This is the constructor whose code is
    // run only when the contract is created.
    constructor() public {
        minter = msg.sender;
    }

    function mint(address receiver, uint amount) public {
        require(msg.sender == minter);
        require(amount < 1e60);
        balances[receiver] += amount;
    }

    function send(address receiver, uint amount) public {
        require(amount <= balances[msg.sender], "Insufficient balance.");
        balances[msg.sender] -= amount;
        balances[receiver] += amount;
        emit Sent(msg.sender, receiver, amount);
    }
}
```

NAME AND SURNAME: TEO BUCCI
STUDENT NUMBER: 993991

ANSWERS

1) Un decentralized ledger è un database decentralizzato. Un database classico è associato a un'autorità centrale che ne gestisce e autorizza i processi, in modo che se un utente utilizza tale servizio, è obbligato a fidarsi dell'autorità. Inoltre, è protetto e fisicamente locato in luoghi di responsabilità di questo ente (ad esempio una banca) ed è modificabile solo da esso. Un decentralized ledger, invece, è distribuito, o meglio replicato, tra tutti gli utenti (nodi) che decidono di partecipare al sistema. Nel caso delle blockchain, il ledger è costituito da una catena di blocchi, che tipicamente contengono transazioni. Questa catena è di solitamente di pubblico accesso (ma esistono anche delle blockchain private, la cui definizione è in parte ancora oggetto di discussione). Il ledger costituisce a tutti gli effetti la valuta che ogni utente possiede, dato che da esso si può ricostruire tutto lo storico dall'apertura del network fino al momento presente.

2) Le proprietà della DLT sono:

- Immutabilità: non deve essere possibile modificare lo storico delle transazioni. Nella blockchain di Bitcoin questo è reso impossibile tramite il sistema di consenso che premia lo sforzo computazionale impiegato per risolvere un particolare problema crittografico. Per riscrivere blocchi passati, ovvero transazioni precedenti, sarebbe necessario possedere più del 50% della potenza computazionale dell'intero network.
- Accessibilità: da un punto di vista sociale, il sistema di Bitcoin non è controllato da un utente, ma da tutta la community che ne fa parte. Tutti i nodi possono leggerne le informazioni e verificare la loro correttezza.
- Essere strutturalmente decentralizzato: la blockchain di Bitcoin, nel senso vero e proprio dei file (circa 350 GB attualmente), si trova replicata fra tutti i nodi.

3) Le curve ellittiche sono entità matematiche usate in crittografia per generare una coppia chiave pubblica-privata che rispetti le condizioni necessarie:

- la chiave privata si usa generalmente per crittare un messaggio ed è conosciuta solo dal proprietario;
- la chiave pubblica è associata a quella privata ed è di dominio pubblico, e può decrittare messaggi crittati con quella privata;
- Se si ha un messaggio crittato con una chiave, si può decrittare solo conoscendo l'altra chiave.

Le curve ellittiche (elliptic curves = EC) sono dei gruppi, ovvero soddisfano:

- Chiusura: a e b sono elementi del gruppo allora a+b è un elemento del gruppo;
- Associatività: (a+b)+c=a+(b+c);
- Esistenza dell'elemento inverso: per ogni a esiste b tale che a+b=0;
- Esistenza dell'elemento identità: 0+a=a+0=a.

Il punto focale è la particolare operazione di somma che viene definita. Considerando il piano, le curve ellittiche si presentano nella forma y^2 = x^3 + ax + b, in particolare in Bitcoin viene usata la curva y^2 = x^3 + 7, nota come secp256k1. La somma è definita come segue: si prendono due punti della curva, si traccia la retta passante per i due, e il "punto somma" è il terzo punto in cui la retta interseca la curva, che sarà unico per le proprietà della curva; nel caso in cui si somma un punto a sé stesso si considera la retta tangente. Nel caso di Bitcoin si parte fissando un punto P sulla curva, noto a tutti, e lo si somma a sé stesso nel senso delle curve ellittiche un numero x di volte. x è la chiave segreta, che non va resa nota a nessuno, un intero a 256 bit scelto a caso. Come risultato si troverà xP = X, questa quantità è la chiave pubblica, che va resa nota a tutti. Prendendo l'hash di X si trova l'indirizzo Bitcoin dell'utente. Inoltre, per evitare che i numeri siano troppo grandi e difficili da memorizzare, non si lavora su R ma su un campo finito (ovvero prendendo le classi di resto modulo p, dove p è un numero primo opportunamente scelto, p=2^256-2^32-977). Le proprietà della curva fanno sì che noto X=xP, non sia possibile ricostruire algoritmicamente x, se non provando tutte le 2^256 possibilità, rendendolo praticamente impossibile; inoltre, non c'è nessuna ragione per cui x sia più vicino a un certo punto piuttosto che a un altro.

4) Smart contract: si tratta di un programma che viene eseguito sulla blockchain, è a tutti gli effetti un account che possiede del denaro e può fare/ricevere transazioni. Non è tuttavia controllato da un utente in modo manuale, ma viene scritto un particolare codice che fa sì che si comporti automaticamente (ad esempio compia una certa transazione verso una certa persona) quando viene attivato da una certa condizione, per esempio la ricezione un particolare file. Gli smart contract sono principalmente legati alla blockchain di Ethereum, che è stata sviluppata proprio come sistema di cloud decentralizzato per eseguirci questi programmi.

Dapp: le decentralized applications sono applicazioni che vengono eseguite sulla blockchain. Dall'esterno sono come qualunque altro tipo di app, ma vengono hostate sulla blockchain. Alla loro base si trova il sistema degli smart contract. Ereditano quindi gli stessi vantaggi di sicurezza, pur essendo più difficili da sviluppare e mantenere.

Wallet: un wallet nel senso delle criptovalute è uno strumento per gestire l'accesso alle proprie criptovalute. In particolare, il wallet contiene le informazioni relative alle chiavi pubbliche e private, pertanto non contiene fisicamente la valuta, ma è un modo per accedervi e compiere transazioni. Ne esistono di versioni sia hardware che software.

Un esempio di utilizzo di una Dapp con smart contract potrebbe essere un servizio di assicurazione interfacciabile tramite Dapp, dove viene gestito l'invio di documenti che provano la necessità di un risarcimento. Per esempio, a seguito di un sinistro stradale si potrebbe caricare il verbale che lo certifica e verrebbe attivato lo smart contract per sbloccare la transazione di risarcimento.

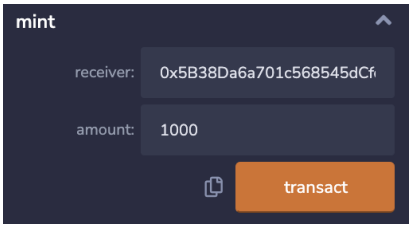
5)
Modifico la funzione send come segue:

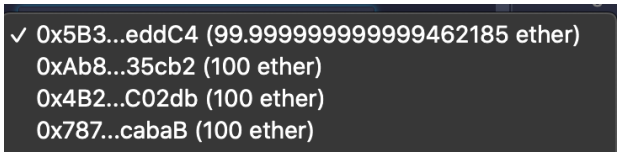
```
function send(address receiver1, address receiver2, address receiver3, uint amount) public {
        require(amount <= balances[msg.sender], "Insufficient balance.");
        balances[msg.sender] -= amount;
```

```
        balances[receiver1] += amount/3;
        balances[receiver2] += amount/3;
        balances[receiver3] += amount/3;
        emit Sent(msg.sender, receiver1, amount/3);
    }
```

In questo modo riceve in input 3 indirizzi, sottrae l'amount dal sender e incrementa il balance di ciascuno dei riceventi di amount/3 (importante non moltiplicare per 0.33 per evitare conflitto di moltiplicazione di un intero per un float).

Dopo aver deployato lo smart contract, uso la funzione mint con l'indirizzo del sender (0x5B3…) dalla sua interfaccia per farsi dare 1000 coin.





Poi usando gli indirizzi di 3 riceventi (0xAb8…,0x4B2…,0x787…) mando 30 coin usando la funzione send modificata.

Infine, con la funzione balances, che tutti possono usare, controllo per esempio il primo ricevente, che ha correttamente ricevuto 10 coin. Controllo anche il sender e verifico che ha 970 coin.