Quantitative Finance

Decentralised Technologies and Cryptocurrencies

29 04 2021- class test

1. Write your number below and **RENAME this file** by changing StudentNumber with your student number (**please do it NOW**).
2. You need to answer to 5 out of 6 question or 4 out of 6 question and the programming (Q6).
3. This test lasts 90 **mins + 15” for uploading the collecting results**. Please, save your word file regularly.
4. Each question Qi is worth 5 points. We expect summary, succinct but clear answers. The PROGRAMMING part is worth 10 points. Six questions OR four questions plus the program add up to 30/30 points: feel free to choose how you collect your 30 points.
5. Your file **should** **not exceed five pages** in total including both questions and answers (font and page layout cannot be changed). Answers can be written in English or Italian. The page with the code of Q7 is not counted.

I tried to answer to all questions and the programming question also. Q4 is the one I would like to discard if not needed in the valuation, because I answered less accurately.

**TEST**

**Q1:** Briefly explain the concept of *decentralisation* presented in the module and its advantages and limitations. Your explanation must be accessible to a student who does not have previous knowledge about blockchain technology. Use an example to support your explanation.

La decentralizzazione sta alla base della tecnologia blockchain e della DLT (distributed ledger technology). Il cambio di paradigma si verifica da un sistema basato sulla fiducia in cui c’è un’autorità centrale a carico della gestione e dell’autorizzazione dei processi al suo interno. Un esempio classico è una banca, dove gli utenti si fidano dell’entità centrale che coordina le transazioni e dove quest’ultima stabilisce cos’è fraudolento e cosa no. Con la DLT vi è un *ledger* (un registro) distribuito, o meglio replicato, tra tutti gli utenti che decidono di partecipare al network. Questo registro contiene tutte le transazioni e, nel caso di Bitcoin per esempio, 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. In un sistema decentralizzato non c’è un utente con più potere decisionale, tutti sono alla pari. Nel caso di Bitcoin in particolare viene premiato, e di conseguenza considerato come veritiero tramite consenso, lo sforzo computazionale impiegato per risolvere un particolare problema crittografico.

**A1**: The concept of decentralization is one of the main pros of blockchain technology and distributed ledger technology.

Decentralization is the shift from a system where you have to trust a central authority (in finance for example: a bank), to one in which there is no need for intermediary and everything is based on a peer-to-peer network.

For example, in the case of the blockchain there is not a single node always in charge of building a new block or of deciding which node will be in charge of building a new block; instead, a leader election is carried on deciding who will augment the chain.

**A1 🇮🇹**: The concept of decentralization is one of the main pros of blockchain technology and distributed ledger technology.

Una delle innovazioni più importanti sviluppate nella tecnologia della blockchain e del distributed ledger è il concetto di decentralizzazione.

La decentralizzazione è il cambiamento da un Sistema in cui tutta la fiduci è da dare a una autorità centrale, ad esempio una banca, ad uno in cui ogni transazione non è gestita da alcuna terza parte ma da una rete peer to peer.

Oltre all’esempio più noto della blockchain un altro esempio di decentralizzazione può essere trovato in sociologia, infatti un Sistema decentralizzato garantisce l'assoluta parità tra tutti coloro che ne fanno parte escludendo ogni tipo di autorità centrale, come ad esempio una democrazia.

**Q2:** What is meant by *consensus* in a distributed system and why it is difficult to achieve? Explain how the miner in charge of defining the next block is selected in the Bitcoin blockchain. How is this selection related to consensus?

Il consenso è il meccanismo tramite cui si determina lo stato corrente del sistema, in particolare lo stato della blockchain e quindi quali sono le transazioni da considerare approvate. È un concetto che risale a un problema informatico più generale, detto problema dei generali Bizantini, dove vi è da prendere una decisione e tutti devono prendere la stessa. Nel caso del problema dei Bizantini: se attaccare o ritirarsi. Il consenso si raggiunge quando almeno il 50%+1 dei nodi è d’accordo sulla versione. Nel caso della blockchain di Bitcoin la decisione da prendere è quale utente scriverà il prossimo blocco. Ciò avviene mediante la risoluzione di un problema crittografico: il primo nodo (*miner*) a risolverlo viene insignito del compito di scrivere il nuovo blocco. Questo sforzo computazionale prende il nome di Proof of Work (dimostrazione del lavoro svolto). Nel nuovo blocco, oltre a scrivere un certo numero di transazioni nel blocco per fissarle, ha diritto a scrivere anche una transazione fittizia in cui viene remunerato una certa quantità di Bitcoin. Questa quantità era inizialmente di 50 BTC, e viene dimezzata ogni 210000 blocchi; attualmente siamo a 6.25 BTC e questo fa sì che in circolazione non ci saranno mai più di 210000\*(50+25+12.5+…)=21 milioni BTC. La PoW è un metodo per permettere a tutti i nodi di verificare velocemente che il puzzle crittografico è effettivamente stato risolto senza doverlo ri-risolvere, e permette quindi di raggiungere il consenso, convergendo su una versione comune. Come conseguenza della teoria dei giochi, il fatto di ricevere questa ricompensa fa sì che tutti i nodi abbiano interesse a comportarsi in questo modo.

**A2**: Reaching the consensus is an historic problem in a distributed system. Reaching consensus means finding an agreement between different participants of a network. When there is a lack of trust, or there are Byzantine participants, isn’t a simple task.

In Bitcoin the consensus concerns which miner will oversee the creation of the next block to be added to the blockchain. To achieve this Bitcoin uses proof of work (PoW): a complex computation must be performed, and the first node able to solve the problem will have the right to augment the blockchain. Since there are many miners working at the same time, usually after one hour the consensus is reached (up to a given probability). The miner selected is rewarded for his work, so there is always an incentive to reach the consensus.

**A2 🇮🇹**: Reaching the consensus is an historic problem in a distributed system. Reaching consensus means finding an agreement between different participants of a network. When there is a lack of trust, or there are Byzantine participants, isn’t a simple task.

Il raggiungimento del consenso è il processo per cui viene trovato un accordo fra tutti i partecipanti alla rete.

In Bitcoin the consensus concerns which miner will oversee the creation of the next block to be added to the blockchain. To achieve this Bitcoin uses proof of work (PoW): a complex computation must be performed, and the first node able to solve the problem will have the right to augment the blockchain. Since there are many miners working at the same time, usually after one hour the consensus is reached (up to a given probability). The miner selected is rewarded for his work, so there is always an incentive to reach the consensus.

**Q3:** Explain briefly at least two of the three among

Merkle Tree,

Hash,

Elliptic Curves (for Digital Signature Algorithm).

The explanation should include: a brief definition; how they are used in the Bitcoin protocol.

Un Merkle Tree è una struttura ad albero costruita a livelli, dove al livello base vi è un insieme di informazioni (nel caso di Bitcoin delle transazioni) hashate, queste vengono poi concatenate a due a due e ri-hashate, e così via fino a che non si ha un solo hash, che costituisce il Merkle Root, la radice dell’albero. Nella blockchain di Bitcoin, è utilizzato nel problema crittografico da risolvere per attuare il meccanismo di Proof of Work. In particolare si prende l’hash del blocco precedente, il Merkle Root delle transazioni che si intendono scrivere nel blocco, la transazione di reward del miner e un numero detto nonce (number only used once). In realtà in questa operazione vengono anche inclusi dei timestamps e altre informazioni. Ciò che è importante è che tutte queste informazioni vengono date in input alla funzione di hash, che darà in output una stringa di lunghezza fissa. La PoW viene soddisfatta quando l’hash risultante soddisfa certe caratteristiche, in particolare che inizi con un certo numero di zeri e che sia inferiore di un certo valore target. Al fine di raggiungere queste condizioni, il nonce viene incrementato ogni volta e l’hash viene ricalcolato (questo è a tutti gli effetti il mining). Le due condizioni permettono di regolare la velocità con cui mediamente un blocco è aggiunto: questa è tenuta intorno a un blocco ogni 10 minuti.

**A3**: Merkle tree : it is a tree made of hashes of transactions, pairwise hashed and then re-hashed. As leaf nodes we have the hash of all the transaction, and then we hash them two by two until the root is reached, which is called Merkle root.

In Bitcoin the Merkle root of all transactions contained in a block, together with the hash of the previous block and the nonce (number used once) are the necessary elements in order to perform the PoW to create a new block and add it to the blockchain.

Un hash è un funzione che prende in input una stringa di qualunque dimensione (quindi potenzialmente anche documenti e foto) e restituisce in output una stringa di dimensione finita. Chiamata H(•) la funzione di hash, le proprietà che deve soddisfare sono principalmente tre:

la “collision resistance”: deve essere computazionalmente impraticabile trovare due input a e b tali che abbiano lo stesso hash H(a)=H(b).

il “hiding”: dato un output b, deve essere computazionalmente impraticabile trovare l’input a che dato alla funzione di hash restituisca proprio b.

il “puzzle friendliness”: dati k e a è computazionalmente impraticabile trovare x tale che H(k\*x)=a.

Nel sistema di Bitcoin le funzioni hash sono il cuore della Proof of Work, infatti il problema crittografico da risolvere consiste nel calcolare svariati hash dando opportunamente in input una serie di informazioni (transazioni, hash del blocco precedente, transazione di reward e altre) concatenate con un nonce (number only used once). Variando questo numero, si cerca un output complessivo che rispetti certe caratteristiche: iniziare con un certo numero di zeri e essere inferiore di un valore soglia (in modo da controllare la difficoltà del problema). Una volta trovato il nonce, queste informazioni vengono diffuse a tutta la rete (tramite un sistema ad “amici di amici” progressivamente), vengono validate facilmente dal momento che bisogna calcolare un solo hash e una volta verificato lo sforzo computazionale compiuto dal miner, tutti aggiungono questo blocco alla propria blockchain.

Hash: it is a function that takes in input a value/string of unlimited dimension and output a value/string of fixed dimension.

Cryptographic hash functions must have three properties:

1) Collision resistance: it is computationally difficult (unfeasible) to find a and b such that h(a) = h(b) 2) Hiding: given b it is computationally difficult (unfeasible) to find a such that H(a) = b

3) Puzzle friendliness: given k and a it is computationally difficult (unfeasible) to find x such that h(k\*x) = a

In Bitcoin hash functions are at the base of PoW: the miners must find the nonce that will correspond to a hash which starts with a certain number of zeros. This number of zeros is adjusted to guarantee that the task is hard, the miner must use a certain amount of computational power. Once the number is found, anybody can check that it corresponds to the declared hash.

Le curve ellittiche sono funzioni matematiche usate in crittografia per generare una coppia chiave pubblica-privata che rispetti le condizioni necessarie [INSERIRE QUI LE PROPRIETA SE NECESSARIO].

Sono della forma y^2 = x^3 + ax + b, in particolare in Bitcoin viene usata la curva y^2 = x^3 + 7, nota come secp256k1. Su queste curve è possibile costruire un particolare modo per sommare due punti del piano (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). Si parte fissando un punto P sulla curva, noto a tutti, e lo si somma a sé stesso nel modo 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 dell’utente. Inoltre, per evitare che i numeri siano troppo grandi, non si lavora su R ma su un campo finito (ovvero prendendo le classi di resto modulo p, dove p è un 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; inoltre, non c’è nessuna ragione che x sia più vicino a un certo punto piuttosto che a un altro.

ECDSA: elliptic curves are mathematical functions used in cryptography to get a public key starting from the private one, so that it is impossible to get back to the private key by knowing the public key.

In Bitcoin the elliptic curve used is y^2 = x^3 + 7; a random 256-bit integer value x is generated (private key), then it is multiplied (in elliptic curve manner) to a point P; the result is the public key X = xP. Finally, the address of a user is found by hashing X.

**Q4:** Take the elliptic curve EC y2 = x3 + 7 in with its closure at infinity. As discussed, on this curve in R or any cyclic subgroup, it is possible to define an addition P+Q, so that the curve is an Abelian group. This implies that mP+nP = (m+n)P.Where is this curve used in the Bitcoin protocol and for which purpose? Give a short explanation.

This curve is used in ECDSA, defined over a Finite Group instead of Real. What does it mean? Which consideration can you add?

**A4:** This specific elliptic curve is used in the process of generating the public key starting from the private key: a random 256-bit integer value x is generated (private key), then it is multiplied (in elliptic curve manner) to a point P; the result is the public key X = xP. Reverting the process is practically impossible.

**~~Q5~~**~~: Informally describe the P2MS (pay to multi-sig) payment in Bitcoin and its most relevant features (we have seen a classification with six different payment types). Then, consider the following P2MS (pay to multi-sig)~~ *~~output script~~* ~~(for your convenience operations are recapped below):~~

~~<2> <Alice's Public Key> <Bob's Public Key> <Tom's Public Key> <3> OP\_CHECKMULTISIG (1)~~

~~Explain how the output script (1) can be used to support a trust system for Alice to sell an item to Bob, with Tom in charge of resolving possible disputes between Alice and Bob (you can outline informally the execution of the script against one redeeming input script, if useful). Discuss limitations and possible improvements, if any.~~

~~SCRIPT OPERATIONS: <X> pushes X on the top of the stack; OP\_CHECKMULTISIG checks whether N out of the M keys in the script are valid keys for M signatures left on the stack by an input script. More specifically, in the output script (1) the value of M, i.e. 2, is initially pushed on the stack (which should already contain the signatures provided by the input script), then the keys are added, and N, i.e. 3, is also pushed on the top of the stack. Finally OP\_CHECKMULTISIG can check the multi-sig payment.~~

**~~A5~~**~~: Multi-sig is used in order to make payment/transaction in bitcoin safer with no need of trust between the counterparties, as it allows a payment to be redeemed only if more than one agrees.~~

~~The output script (1) of the question checks if at least two out of three signatures are present, and based on which signatures it finds, proceeds as it was programmed to.~~

~~For example, let’s say Alice wants to buy something from Bob but they do not trust each other.~~

~~Alice makes a multi sig payment and two out of them will have to redeem the transaction. Also Tom is third party with the power to rule in favor of either Alice or Bob~~

~~Three possibilities:~~

~~Alice and Bob agree and sign the payment to Bob.~~

~~Alice and Bob disagree and Tom signs in favour of Alice (payment to Alice).~~

~~Alice and Bob disagree and Tom signs in favour of Bob (payment to Bob).~~

~~Some limitations may be the need of a third party (Tom), or in other examples of multisig (we have seen the idea of Micropayments), when the input has to be given within a certain time limit, one of the two participants could attack the other with a DDOS attack.~~

**Q6:** Explain informally what a smart contract is and discuss one example of a smart contract with a possible implementation on bitcoin.

**A6**: Smart Contracts are programs written in code that implement a self-executing logic on a blockchain; they reside inside a transaction and are executed by every node. In Ethereum a smart contract is associated to a certain amount of gas (proportional to the complexity of the code) that must be paid to make the program run.

An example of smart contract on Bitcoin is using multisig to pay for small sums, for example drinks at a bar, which is not possible to do directly since transactions need 60 minutes in order to be confirmed. The customer and bartender can agree on transferring a certain amount into a multisig and taking track of the purchases off the blockchain. Only at the end they will write it on the blockchain.

**Q7 -PROGRAMMING** Copy the solidity program that is given below. You can use Remix to compile, run and test it:

* Change the contract behaviour so that the account can support payments only with a limited threshold. Modify the program so that the transfer is allowed only with a threshold. Do this changes, and report the modification, explaining what you did and why.
* Deploy locally (Environment Javascript VM) and make a transfer from your account to one of the demo account available in the list, take Screenshort and copy it in the answer to Q7.

**PROGRAM**

pragma solidity >=0.5.0 <0.7.0;

contract MyCoin {

// The keyword "public" makes variables accessible from other contracts

address public minter;

mapping (address => uint) public balances;

// Events allow clients to react to specific contract changes you declare

event Sent(address from, address to, uint amount);

// Constructor code is only run when the contract is created

constructor() public {

minter = msg.sender;

}

// Sends an amount of newly created coins to an address

// Can only be called by the contract creator

function mint(address receiver, uint amount) public {

require(msg.sender == minter);

require(amount < 1e60);

balances[receiver] += amount;

}

// Sends an amount of existing coins from any caller to an address

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);

}

}

**A7:**

1. To require a certain threshold in the function “send”, modify as:

function send(address receiver, uint amount) public {

require(amount <= balances[msg.sender], "Insufficient balance.");

require(amount < 1e60); #1e60 can be changed, it is the threshold

balances[msg.sender] -= amount;

balances[receiver] += amount;

emit Sent(msg.sender, receiver, amount);

}

Using “require” the function checks that the input amount is not greater than a certain threshold.

**Immagine che contiene testo

Descrizione generata automaticamente**

After deploying the contract from the first account (0x5B…), I used the mint function to give myself 1000 coins. Then by simply copying the second account address (0xAb…) I used the send function and a value of 100 to send 100 coins. Finally, I checked everything went well by using the “balances” function on the second address: the balance is indeed 100 as expected.