1 Blockchains and proof-of-stake

1. What is a blockchain, in 3 fundamental components?
2. Discuss the relevance of using a blockchain technology in the following cases:
   (a) I need to share data that only I can modify.
   (b) I need to share data and allow modifications with a circle of friends.
   (c) I need to share professional data and allow modifications with different staff in my company.
   (d) I need to share professional data and allow modifications with my business partners.
   (e) I need to share professional data and allow modifications with all my potential customers.
   (f) I need to share professional data and allow modifications with the big players in the field and guarantee that all actions are verifiable.
   (g) I need to share professional data and allow modifications with a trusted authority.
3. Who are the most requested validators in a proof-of-stake consensus algorithm, and why?
4. What is a "nothing at stake" attack on a proof-of-stake, and propose some counter-measures.

(1) A blockchain is a combination of the following:

→ Distributed ledger

→ Consensus protocol

→ Crypto-currency

(2) a. This is not a good example of the use of a blockchain, there are no blockchain that fullfil this property

b. This may be suited for a private blockchain

c. This may be fullfilled with a private or hybrid blockchain

d. This would be perfectly suited for an hybrid blockchain or private if we can include all partners

e. This could be either hybrid or fully public

f. This could be the case of Public permissioned blockchain

g. Same case as f.

(3) In proof of stake, the most requested validators are the one with highest reputation, activity, amount of coins owned (wyh) ....

(4) This problem leverages the fact that mining a block in the "wrong" chain in case of a fork is relatively cheap. So miners may decide to mine new blocks in both chains making the system more susceptible to double spending attack. A counter measure can be penalization or incentive to validate the main chain.

2 Bitcoin

1. What is a proof of work protocol in the blockchain context?
2. The difficulty of the bitcoin proof of work mining, and thus the current target, is modified every block as a function of the 2016 previous valid blocks. To how many days does it correspond?
3. This compensation depends on the actual time required to validate those previous 2016 blocks. How is it possible to get this real time?
4. We suppose that the results of SHA-256 are uniformly random, and that the current target is: 00000000 00000000 006C2146 00000003 A6D973D5 B458E659 B5E926C8 2648746E, What is approximately the power of the world network, in number of hashed blocks per second?
5. If the real time is lower than the expected time, that is the network is more powerful than expected, the target is reduced, otherwise the target is increased. The modification is in any case directly proportional to this time ratio. How is it possible to simultaneously know the current target in a distributed manner?
6. To enable humans to better understand the notion of target, it is often represented as a **difficulty** to find a hash lower than the target. If the real time is lower than the expected time, that is the network is more powerful than expected, the difficulty is increased, otherwise the difficulty is reduced. This notion of difficulty is the ratio between a maximal target (00000000 FFFF0000 00000000 00000000 00000000 00000000 00000000 00000000) and the current target (the higher the target, the easiest it is to validate a bloc). What is approximately the current difficulty?
7. A minor finds a nonce allowing to validate a block of transactions and publishes it (this is verifiable, so this means that the nonce is public). What prevents another miner to take this nonce and to claim the rewards for himself?
8. The mining reward is halved every 210 000 mined blocks. Knowing that the first bitcoin was mined on March, 3, 2009, with a reward of 50 BTC, and that one bitcoin is 100 000 000 satoshis, when will the last bitcoins be created?
9. An $E10$ ASIC computer can test about 18 millions of millions of nonces per second. Supposing that a block is validate every 10 minutes how many $E10$ machines would be required to mine?
10. An $E10$ require performs also 11.1 billions of tests per joule of electricity. What would be the annual consumption of electricity of the machines of the previous question if they were alone to mine?

① In the context of blockchain a proof of work algorithm refers to the problem required to mine a cryptocurrency and validate the transactions in the ledger.

$$SHA256\left(SHA256(Transactions, nonce)\right) = x < Target$$

↑ we have to find this

② We have to assume 10 minutes per block and 2016 blocks before a new adjustem happens.

$$\frac{10\,minutes}{1\,block} \times 2016\,block = 20160\,minutes \rightarrow 14\,days$$

(3) We may look at the longest chain and have an idea of how many blocks have been already validated.

(4) We assume that 10 minute is required for the block so the mentioned block is the new target and it will take 10 minutes to be found.

$5 \times 8 \rightarrow 48 + 6 = 54$ values to be found

each value can range between $[0, F]$ so each of them have 16 possibilities.

16^{54} : 10 minute = x : 1 minute

$x = \frac{16^{54}}{10}$ Hashes per minute

(5) It is sufficient to to synchronize to the last element of the transaction and get always the upto date target

3 Merkle hash-tree for integrity

We want to design an integrity check of a database by Merkle hash-trees. The setup of the protocol is as follows:

- Split the database in blocks;
- Hash each block, this forms a layer of hashes;
- Combine these hashes 2 by 2 to form a new layer (for an odd number of hashes, the last one is duplicated)
- Repeat the 2 by 2 combination until there remains only a single hash; this hash, r_0, is called the Merkle root.

Figure 1 shows an exemple with 6 blocks.

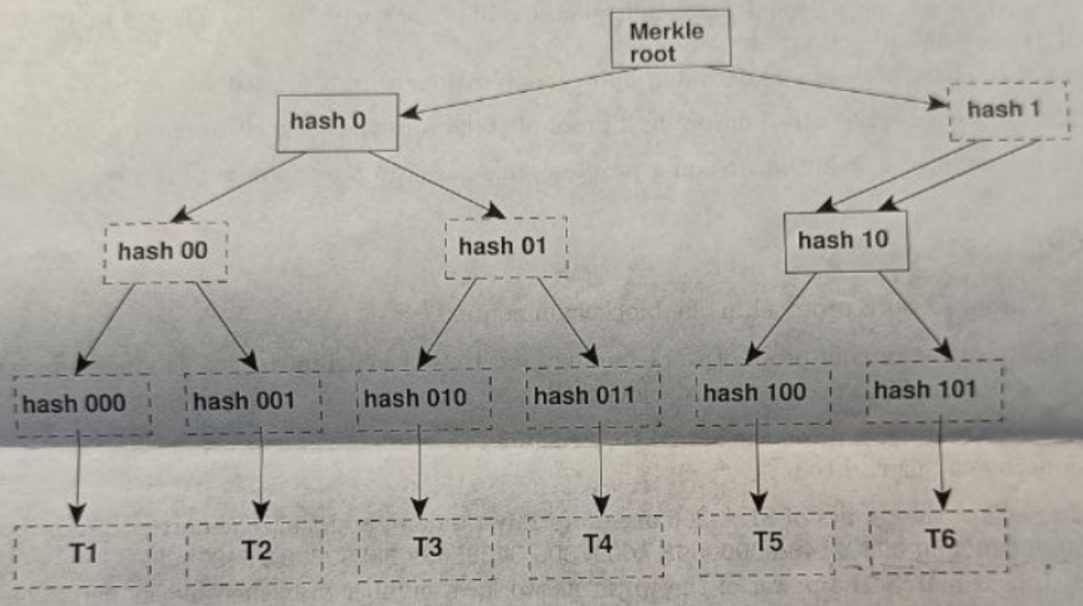

Figure 1: Merkle hash-tree with 6 blocks

Then we will imagine a cloud computing setting where a client stores only the Merkle hash root and a server stores the database.

1. What is the depth (number of layers) of a tree formed from n blocks? $\lceil \log_2(n) \rceil + 1$
2. How many calls to the hash function are required to build the tree?
3. What memory is required for an algorithm updating a hash tree with new blocks, in order to use the minimal number of extra hash operations per update?
4. Design an integrity certificate provided by the server enabling a very fast check by the client that a given value is present as a block within the database. What is its complexity in terms of Communication and client operations?
5. Is the scheme of Figure 1 resistant to second preimage if the hash function is resistant to second preimage?
6. Design a counter-measure.
7. Can this be used by the Client to ask to retreive the i-th value of the database together with an integrity proof?

① The depth is $\log_2(n)+1$

② It requires 2^n calls to the hash function (if we assume a complete tree