University of Stavanger

Faculty of Science and Technology

EXAM IN SUBJECT:	**DAT650 BLOCKCHAIN TECHNOLOGY**
DATE:	**EKSAMEN, 29TH NOVEMBER 2022**
DURATION:	**4 HOURS**
ALLOWED REMEDIES:	**NONE**
THE EXAM CONSISTS OF:	**13 EXERCISES ON 6 PAGES**
CONTACT DURING EXAM:	**LEANDER JEHL, TLF. 94120764**
REMARKS:	None
ATTACHMENTS:	

Question 1: Trees (13%)

(a) (4%) Sketch a Merkle tree with 7 data elements. What data needs to be included in an inclusion proof for the third data element?

> **Solution:** Need hashes h_4, $h_{1,2}$, and $h_{5,6,7,8}$. Additional position information, left, right, left.

(b) (3%) If Alice did send some bitcoin to Bob, how can she convince Bob that the payment happened? How can she do this efficiently?

> **Solution:** Alice needs to show Bob a merkle proof showing the transaction is part of a block and block headers for 5 blocks on top.

(c) (6%) Alice wants to *read* the value of a *owner* variable in a smart-contract deployed on Ethereum. Alice does not run her own Ethereum node. How can she get an answer she trusts. How can this be done efficiently?

> **Solution:** Contracts in Ethereum contain the storage root, i.e. the root of a Merkle tree showing the contracts storage, i.e. all variables. Alice could receive a Merkle proof showing the value of the owner variable. Further, the state of a contract in Ethereum is included in the Storage tree. The root of the tree is included in every block. Alice could receive an inclusions proof for the state of the given smart contract. Finally, Alice should receive the block header, including the storage root, and possibly multiple headers on top, to show confirmation.

Question 2: Unspent transaction output (UTXO) (11%)

(a) (6%) What are the different elements of a transaction in Bitcoin? How is a transaction validated?

Solution: Transactions include inputs and outputs. Validation checks that

- Inputs reference previously unspent outputs
- Inputs fulfill condition on outputs (i.e. signature with correct key)
- Sum of outputs is less or equal to inputs.

(b) (2%) How does Bitcoin prevent the following attack:

- Alice wants to send bitcoin to Bob. She issues a transaction for this transfer. Charly intercepts this transaction before it reaches the Bitcoin network. He replaces Bobs public key with his own, submits the new transaction, and thus receives the payment from Alice.

Solution: Alice needs to sign the transaction. The signature includes also the public key of Bob. With the public key replaced, the transaction will no longer be valid, unless Alice creates a new and different signature.

(c) (3%) Name 3 reasons, why technology like Bitcoin is not suited to be used in everyday payments.

Solution: Throughput, Confirmation time, Cost

Question 3: Proof of work (19%)

(a) (6%) We had the following definition for a proof of work function:

Def: *For an integer d, the proof-of-work (PoW) function with difficulty d takes a data item and returns a nonce (random bits) and a hash value:*

$$(h_{PoW}, nonce) = f_{PoW}(Data)$$

The proof of work is valid, *if a) h_{PoW} is the hash of the data, concatenated with the nonce*

$$h_{PoW} \overset{?}{=} H(Data||nonce))$$

and b) the first d bits of h_{PoW} are 0.

What are the short-commings of this definition and how can it be adjusted?

Solution: The above definition does not allow fine tuning the difficulty. Only allows to make PoW twice or half as difficult.

A better definition for a difficulty is to take d a hexadecimal number and require $h_{PoW} < d$.

(b) (4%) Assume Alice finds a nonce that allows her to solve the PoW puzzle and create a new block for the Bitcoin blockchain. Alice sends this block to the bitcoin network.

Assume further that Bob is the first node in the network to receive Alices block. Can Bob take the Nonce from Alices block and cash in the block reward himself?

> **Solution:** No. To receive the block rewards, Bob needs to include his address in the coinbase transaction. Thus his block will be different from Alices block and he will most likely need a different nonce.

(c) (4%) Assume Alice runs a bitcoin miner. After searching for a solution for 10 minutes, Bob finds a block. If Alice stops her initial mining, and instead mines on top of the block that Bob has found, does she lose an advantage? Explain.

(d) (1%) a proof of work function needs three properties, *Fast Verification*, *Adjustable difficulty*, and *Progress freedom.* Which of the three properties of a PoW function is relevant for the example in Part c.

(e) (4%) We say that PoW is fair if in the longest chain, the amount of blocks from every miner is proportional to his mining power/hash rate.

- Under which conditions, is a PoW scheme like in Bitcoin not fair? Give an example.
- Explain one technique to alleviate this problem.

Question 4: Attacks (3%)

(a) (3%) How can a miner benefit from a selfish-mining attack? Will he mine more blocks?

> **Solution:** The attacker will not mine more blocks, but he causes other blocks to be discarded. This can increase the fraction of blocks in the longest chain, created by the attacker.

Question 5: Proof of Stake (6%)

In PPCoin (Peercoin) a miner identified by *addr*, that has deposited $\texttt{coin}(addr)$ can supply the current block, if

$$H(\texttt{prevBlockHash}||addr||\texttt{timeinseconds}) < d_0 \cdot \texttt{coin}(addr)$$

- Here d_0 is a base difficulty. The probability that a miner with a specific address *addr* can mine the next block is proportional to $\texttt{coin}(addr)$.
- `timeinseconds` shows time in seconds. Thus, a miner gets a change to submit a solution every second.

(a) (3%) Give an example, where a PoS-miner can benefit from an attack similar to proof of work.

(b) (3%) Explain the nothing-at-stake problem. How can that problem be solved?

Question 6: GHOST and Longest chain rule (4%)

Figure 1 shows an example of a chain with many forks.

(a) (2%) Which block should be extended according to the longest chain rule?

(b) (2%) Which block should be extended according to the GHOST rule?

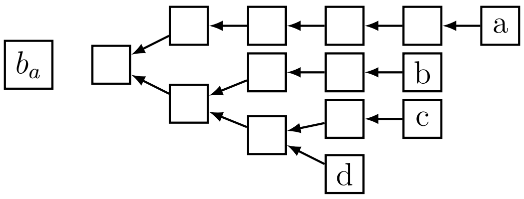

Figure 1: Example chain with many forks.

Question 7: Bitcoin-NG (9%)

Bitcoin-NG defines both key-blocks and micro-blocks. Fees from Micro-blocks are divided 40/60 between the creator of the last and next key block.

(a) (7%) Compared to Bitcoin, which of the following properties can Bitcoin-NG improve. If an improvement is possible, explain how?

- Confirmation time
- Transaction throughput
- Security

(b) (2%) What attack would be possible if the creator of the next key block would not get a part of the fees, e.g. a 100/00 divide?

Solution: A miner could, instead of extending the micro-blocks, extend the last key block. This allows him publish his own micro-blocks, including the transactions.

Question 8: Hybrid blockchains (6%)

Hybrid blockchains combine PoS or PoW with a consensus algorithm like Hotstuff.

(a) (2%) What is the advantage of combining PoW with a Hotstuff like algorithm, like in ByzCoin?

(b) (4%) What additional advantages arise when combining PoS with a Hotstuff like algorithm?

Question 9: Hotstuff (4%)

What are the advantages and disadvantages of 3-chain Hotstuff, compared to 2-chain?

Question 10: Off-Chain technology (9%)

Off-chain technology includes Payment-Channels, Payment-Channel networks and Commit Chains.

(a) (5%) What are the advantages and disadvantages or limitations of payment channels, compared to on chain transactions?

Solution: Advantages: Fewer fees, fast confirmation. Disadvantages: Need to lock fees, need to stay online, only single recipient.

(b) (4%) What are the advantages and disadvantages of payment channels, compared to commit chains?

Solution: Channels: Fast confirmation+ Commit Chains: No registration cost+

Question 11: Ethereum (6%)

How are transaction fees set in Ethereum, and how can a user control, predict, or limit the fees? How can the user set fees to ensure inclusion in the blockchain?

Solution: Based on what code the transaction executes, a cost in Gas is calculated. This can be predicted by pre-executing the transaction, but may change for the actual execution. The fee in Ether is then computed as the gas cost * gas price given by the user. The gas price needs to be set large enough, otherwise, the transaction will not be included in the blockchain. A large gas price makes the transaction being included faster. The user can also give a maximum gas cost. The transaction will not use more gas than this maximum but stop and revert if it reaches this point.

Question 12: Solidity (12%)

The contract given in Algorithm 1 is vulnerable to re-entrancy.

(a) (4%) Describe how the vulnerability can be exploited by an attacker.

(b) (4%) Rewrite the `deposit` function to remove the vulnerability. You should still use `call` to send money.

(c) (4%) What is the advantage of using `call`, rather than `transfer` when sending money. Why is the first vulnerable and the second is not?

Question 13: SmartContract Frontend (Lab 4) (4%)

A Web frontend for an Ethereum SmartContract, like the one developed in Lab 4 requires to know the interface and address of SmartContracts. How do you supply this information in code?

Algorithm 1 Reentrancy

```
pragma solidity =0.5.11
contract SimpleBank {

    mapping(address => uint) balances;

    // deposit money.
    function deposit() public payable {
        require(msg.value > 0);
        uint balances[msg.sender] = balances[msg.sender] + msg.value;
    }

    function withdraw(uint value) public {
        require(balances[msg.sender] > value);
        (bool success) = msg.sender.call.value(value);
        if (success) {
            balances[msg.sender] = balances[msg.sender]-value;
        }
    }
}
```