Chair of Software Engineering for Business Information Systems
Department of Informatics
Technical University of Munich

Eexam
Place student sticker here

Note:

- During the attendance check a sticker containing a unique code will be put on this exam.
- This code contains a unique number that associates this exam with your registration number.
- This number is printed both next to the code and to the signature field in the attendance check list.

Blockchain-based Systems Engineering

Exam:	IN2359 / Endterm Online	**Date:**	Wednesday 4th August, 2021
Examiner:	Prof. Dr. Florian Matthes	**Time:**	08:00 – 09:30

Working instructions

- This exam consists of **18 pages** with a total of **8 problems**.
  Please make sure now that you received a complete copy of the exam.
- The total amount of achievable credits in this exam is 90 credits.
- Detaching pages from the exam is prohibited.
- Allowed resources: none
- Subproblems marked by * can be solved without results of previous subproblems.
- Do not write with red or green colors nor use pencils.
- Physically turn off all electronic devices, put them into your bag and close the bag.

Problem 1 Cryptographic Basics (10 credits)

Alice and Bob want to play rock-paper-scissors over a peer-to-peer connection. To prevent cheating, they want to use their knowledge of cryptography to devise a commitment scheme based on hashing. To start out, they consider possible hash functions.

0 □ □ 1 □

a)* At one point, Bob proposes the following function.

$$h(x) = x + 17$$

Explain why this function is not a hash function.

0 □ □ 1 □ □ 2 □

b)* Next, they fix Bob's mistake and consider the following simple hash function:

$$g(x) = (x + 17) \mod 1024$$

Still, they deem it unsuitable. Recall the key properties of **cryptographic** hash functions from the lecture. Name a property this function violates and briefly explain why.

Given some time, Alice and Bob come up with some arbitrary cryptographic hash function *h* and the following scheme. One round looks like this:

1. Both secretly choose one option: rock, paper, or scissors.
2. Both compute the hash $h_i = h(choice_i)$ and send it to each other.
3. When they reveal their choice to the other, the other can verify that the commitment was made before the reveal by hashing the revealed choice and comparing to the previously received hash.

0 □ □ 1 □

c)* Where is the flaw in this scheme?

0 □ □ 1 □

d) Propose a way to fix this scheme.

Since Alice and Bob are busy students, they decide to save time and expand their scheme to support playing multiple games per round of their scheme. Naively sending *n* commitments at once leads to a linear increase in required hashes and thus an increase in network traffic. Well versed in cryptography, they want to decrease the required network traffic by using Merkle trees.

e)* Given the use of Merkle trees, what is the minimum number of hashes Alice has to send to Bob to commit to {rock, paper, scissors} for a round consisting of 16 games.

□ 0
□ 1

f)* Alice and Bob agree to play a round consisting of 3 games. Draw the Merkle tree over Alice's three hashes $h(c_1) = 1$, $h(c_2) = 3$, and $h(c_3) = 7$. For this construction assume

$$h(x) = x \mod 8$$

and use addition to combine hashes (instead of concatenation).

□ 0
□ 1
□ 2
□ 3
□ 4

Problem 2 Bitcoin and Blocks (6 credits)

Bitcoin was the first cryptocurrency and is still one of the most popular ones. Figure 2.1 shows the structure of a Bitcoin block.

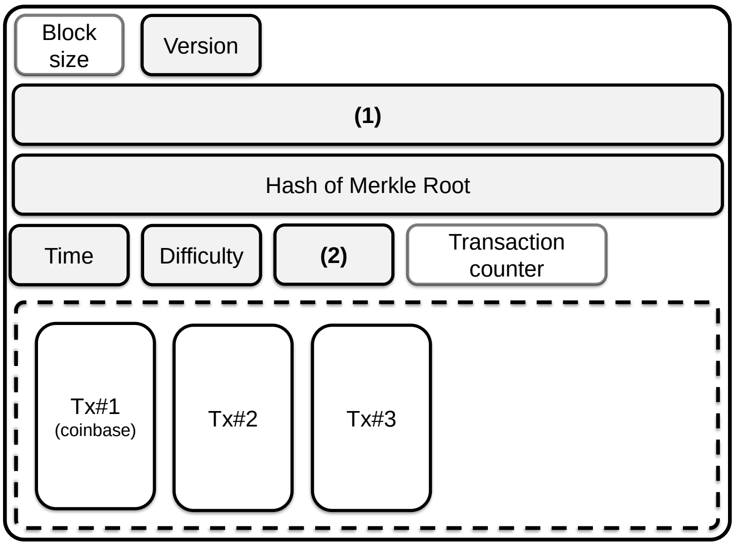

Figure 2.1: Structure of a Bitcoin block.

0 □ □ 1 □ □ 2 □

a)* Name the two missing fields in the block.

0 □ □ 1 □

b)* The hash of the block header includes all data from `Version` until `(2)`. Explain how transactions are included in the header.

0 □ □ 1 □ □ 2 □

c)* The figure shows that the first transaction in a block is called `coinbase` transaction. Name two functions it can fulfill.

0 □ □ 1 □

d)* The header features a timestamp (`Time`), which is one of the values required to govern bitcoins mining difficulty (also found in the header). How are these two values related?

Problem 3 **Bitcoin Transactions (12 credits)**

a)* The Bitcoin blockchain is a ledger recording transactions between entities. What identifies an entity **on-chain**?

□ 0
□ 1

For the following questions, look at the transactions shown in figure 3.1. The left side shows input, while the right side shows outputs. Suppose, all of them have been included in a Bitcoins block and are part of the currently agreed upon state.

Tx1	
Ø	25 → Alice
(Txin)	*(Txout)*

Tx2	
#1[0]	4 → Bob 10 → Alice 5 → Dave

Tx3	
#2[0]	2 → Carol 2 → Dave

Tx4	
#2[2]	3 → Alice 1 → Dave

Figure 3.1: Four Bitcoin transactions.

b)* Which entity has created transaction output `#3[1]`?

□ 0
□ 1

c)* Which entity is allowed to spend the transaction output `#2[0]`?

□ 0
□ 1

d)* Calculate the balances for each entity after Tx4.

□ 0
□ 1
□ 2

e)* For Tx4, the sum of all inputs is larger than the sum of all outputs. What happens to the excess currency?

□ 0
□ 1

UTXOs are locked using Bitcoin script, making sure only the intended recipient gets to spend them. Probably the simplest type of script is pay-to-public-key (P2PK). Over time, this type has been replaced by an updated version.

0 □ □ 1 □ □ 2 □

f)* What is the successor to P2PK transactions called? Also name one reason it is superior.

0 □ □ 1 □ □ 2 □

g)* Apart from P2PK (and its successor) there are other well known transaction types. Name one and briefly explain it.

Bitcoin has always had a scaling problem. To decrease transaction sizes and fit more transactions into each block, the Segregated Witness update (SegWit) was introduced in 2017. SegWit separates signature data from the transactions and appends it at the end of the block.

0 □ □ 1 □ □ 2 □

h)* Name the two types of scripts that are concatenated for a transaction verification. Also state which one of them is shortened by SegWit.

Problem 4 Bitcoin Network and Consensus (13 credits)

a)* What is the name of Bitcoin’s consensus mechanism?

□ 0 □ 1

b)* The Bitcoin network consists of many nodes all over the world. Name two types of nodes in the bitcoin network and state if each of them performs validation.

□ 0 □ 1 □ 2

c)* For a long time now, miners group up in so called mining pools. Name

- one reason why miners join mining pools and
- one disadvantage of mining pools for the network as a whole.

□ 0 □ 1 □ 2

d)* What is (so far) the most important metric to consider when choosing hardware as a miner, apart from hashrate?

□ 0 □ 1

e)* As of now (July 2021), Bitcoin is experiencing an unprecedented decrease in **mining difficulty**. Before the decrease, block times went up to $>$ 15 minutes for several days due to China cracking down on their ban on crypto and destroying mining operations [1]. Then, at the next difficulty adjustment, mining difficulty dropped dramatically. What are the effects that this has on the network in the **short term** ($\leq$ 2 Weeks) directly after the difficutly decrease.

□ 0 □ 1 □ 2

f)* What is the minimum percentage of nodes required to attack the Bitcoin blockchain to reliably rewrite history?

□ 0 □ 1

[1] China’s war on bitcoin just hit a new level with its latest crypto crackdown, MacKenzie Sigalos, CNBC, retrieved from `https://www.cnbc.com/2021/07/06/china-cracks-down-on-crypto-related-services-in-ongoing-war-on-bitcoin.html` on 28th July 2021

The next big Bitcoin update is called Taproot and scheduled for November. It introduces a new signature scheme, the Schnorr signature. These signatures allow signature aggregation, improving performance and privacy aspects. Its designers have purposefully crafted this fork such that all blocks considered valid after Taproot are also considered valid by older versions.

0 □
1 □

g)* What kind of fork is Taproot?

0 □
1 □

h)* Taproot's designers have ensured that Taproot has at least 90% support within the Bitcoin community. How is this support for a potential fork measured?

0 □
1 □
2 □

i) Based on the information within this exercise and your previous answers, will Taproot cause a chain split? Explain your answer.

Problem 5 **Ethereum (20 credits)**

A miner created a block containing a transaction. There are further transactions in this block, however they are irrelevant to this task. The transaction is depicted in table 5.1.

Position	Sender	Recipient	Nonce	Value	STARTGAS	gas used	GAS PRICE	data
5	`0xaaa...`	`0xbbb...`	157	0 ETH	500,000	467,352	70 Gwei	`0xb76eff...`

Table 5.1: Block containing a transaction.

Furthermore, this transaction leads to several messages displayed in table 5.2.

Position	Sender	Recipient	Value	STARTGAS	gas used	data
0	`0xbbb...`	`0xccc...`	0 ETH	60,000	38,543	`0x`
1	`0xccc...`	`0xddd...`	0 ETH	90,000	80,776	`0xb76ea962ff...`
2	`0xbbb...`	`0xaaa...`	0 ETH	100,000	21,000	`0x`
3	`0xbbb...`	`null`	0 ETH	93,000	92,830	`0x6080604052...`

Table 5.2: Resulting messages of transaction 5.

a)* State the fundamental design difference regarding payment structure in comparison to Bitcoin stating the names of both design approaches.

□ 0 □ 1 □ 2

b) What is the problem resulting from Ethereum's design approach (referring to a)) that necessitates the introduction of a `Nonce`? And how does the `Nonce` mitigate the problem?

□ 0 □ 1 □ 2

c)* Notice that the column `Nonce` is not mentioned in the messages table. Explain why.

□ 0 □ 1

d)* Present how to calculate the transaction fees paid for the transaction in Ether. You do not have to actually calculate the result. A correct mathematical term using the concrete numbers and (if applicable) units is sufficient.

□ 0 □ 1

0 □ □ □ 1

e)* Who paid for the execution of message #1?

0 □ □ 1 □ □ 2 □ □ 3 □

f)* Name both types of Ethereum accounts. For each type state one account address that certainly is of that type and give a **brief** explanation.

0 □ □ 1 □ □ 2 □ □ 3 □ □ 4 □ □ 5 □

g)* Alice (0xAlice...) wants to transfer three tokens (managed in contract 0xyyy...) to Bob (0xBob...). Name the following parameters for a transaction in which Alice transfers three tokens to Bob. **If you cannot provide exact values, give realistic estimates. Provide units if necessary.** For your convenience, we provide the ERC20 interface in listing 1 [2]. For the data field, please do **not** input hexadecimal values, but name the function and arguments to call.

(Sender):

(Recipient):

(Value):

(STARTGAS):

(gas used):

(GAS PRICE):

(data):

```
contract ERC20Interface {
    function totalSupply() public constant returns (uint);
    function balanceOf(address tokenOwner) public constant returns (uint balance);
    function allowance(address tokenOwner, address spender) public constant returns (uint remaining);
    function transfer(address to, uint tokens) public returns (bool success);
    function approve(address spender, uint tokens) public returns (bool success);
    function transferFrom(address from, address to, uint tokens) public returns (bool success);

    event Transfer(address indexed from, address indexed to, uint tokens);
    event Approval(address indexed tokenOwner, address indexed spender, uint tokens);
}
```

Listing 1: ERC20 Smart Contract Interface in Solidity

[2] ERC Token Standard #20 Interface, https://eips.ethereum.org/EIPS/eip-20, retrieved from https://de.bitcoinwiki.org/wiki/ERC20 on 12th July 2021

h)* Domain experts approach you: They want to tokenize key items in their business. Argue about whether ERC721 or ERC20 tokens are better suited for their use case. Domain expert A plans to issue company shares as tokens and expert B wants to expand their art gallery's portfolio via tokenized art.

□ 0
□
□ 1
□
□ 2
□
□ 3

(A):

(B):

Currently, Ethereum relies on proof-of-work (PoW) for consensus. There is a strong movement to gradually transition to proof-of-stake (PoS) in the near future. PoS replaces miners with validators, that propose blocks (without mining) and validate blocks they receive from other validators. To participate, a validator has to stake (i.e. lock up) some Ethereum. Then they get rewarded for correct validator behavior and lose their stake otherwise.

i)* Name two advantages, that PoS will most likely have compared to PoW.

□ 0
□
□ 1
□
□ 2

Problem 6 Ethereum Smart Contracts - Solidity (15 credits)

Not content with their previous version of rock-paper-scissors, Alice and Bob want to up the stakes and create a Dapp version. They decide to use an Ethereum smart contract as backend for their game. These are the specifications for that smart contract:

- Alice will deploy the smart contract and designate Bob as `playerTwo`.
- Only Alice and Bob are allowed to play.
- One game consists of three steps:
  - Both call `commit()` to commit to a choice out of rock (0), paper (1), or scissors (2). Whoever goes first sets the bet size depending on how much Ether they send with their transaction. Whoever is second must match that bet exactly.
  - Then, both call `reveal()` to reveal their committed choice.
  - Finally, one of them (usually the winner) calls `distributeWinnings()` to trigger payout and reset the game.
- The minimum bet size is 1 Ether.
- Naturally, committing and revealing can only be performed once per person per game.

As Bob and Alice are friends, they agree to keep the contract simple and **not** account for anyone of them refusing to play and locking up the other's bet.
You are allowed to use all functionality that Solidity provides. As you can see on the next page, the smart contract assumes a current version of Solidity, but **older syntax will be accepted**. This Solidity version includes SafeMath and thus you do **not** need to account for over-/underflow of variables.
Some of the following code snippets may be useful:

- Sender of the transaction: `msg.sender`
- Amount sent with the transaction: `msg.value`
- Enforcing conditions: `require(...)`
- Transfer assets: `recipient.transfer(...)`
- Unit literals: ether, finney, wei
- Casting arbitrary data to uint: `uint(...)`
- Casting an address `a` to a payable address: `payable(a)`
- Empty address: `address(0)`
- Returns Ether balance of the contract: `address(this).balance`

Convert the instructions on the previous page into code.

```
pragma solidity >=0.8.4;

contract RockPaperScissors{
    address payable owner;
    address payable playerTwo;

    mapping(address => bytes32) commitments;
    mapping(address => bool) isPlaced;
    uint bet;

    mapping(address => uint) revealedCommitments;
    mapping(address => bool) isRevealed;

    mapping(uint => mapping(uint => uint)) private gameResults;

    // restricts access to players only
```

a)

```
    modifier onlyPlayer() {
```

□ 0 □ □ 1 □ □ 2

```
    }

    // 0, 1, 2 are rock, paper, scissors
```

b)

```
    constructor (address _playerTwo) {
```

□ 0 □ □ 1 □ □ 2

```
        gameResults[0][0] = 0;
        gameResults[1][1] = 0;
        gameResults[2][2] = 0;
        gameResults[0][1] = 2;
        gameResults[1][2] = 2;
        gameResults[2][0] = 2;
        gameResults[0][2] = 1;
        gameResults[1][0] = 1;
        gameResults[2][1] = 1;
    }

    // allows to commit to one choice per game and processes the bet
```

c)

```
    function commit(bytes32 _hashedChoice) public payable onlyPlayer {
```

□ 0 □ □ 1 □ □ 2 □ □ 3 □ □ 4 □ □ 5 □ □ 6 □ □ 7

```
    }

    // allows players to reveal their commitment and checks that it is correctly revealed
    function reveal(uint _choice, int _nonce) public onlyPlayer {
        require(isPlaced[owner]);
        require(isPlaced[playerTwo]);
        require(isRevealed[msg.sender] == false);
        _choice = _choice % 3;
```

d)

```
        bytes32 claimHash = keccak256(abi.encodePacked(_choice, _nonce));
```

□ 0 □ □ 1 □ □ 2 □ □ 3

```
    }
```

0 □
□
1 □

e)

```
    // at the end of the game, this method pays the winner
    function distributeWinnings() public onlyPlayer {
```

```
        uint result = gameResults[revealedCommitments[owner]][revealedCommitments[playerTwo]];
        if(result == 0){
            owner.transfer(bet);
            playerTwo.transfer(bet);
        } else if(result == 1){
            owner.transfer(2*bet);
        }else{
            playerTwo.transfer(2*bet);
        }
        isPlaced[owner]=false;
        isPlaced[playerTwo]=false;
        bet = 0;
        isRevealed[owner]=false;
        isRevealed[playerTwo]=false;
    }
}
```

Problem 7 Hyperledger Fabric (7 credits)

a)* In contrast to previously discussed ledgers such as Bitcoin or Ethereum, Fabric restricts access to its chain. What type of blockchain is Fabric and which entity governs access?

□ 0
□
□ 1

b)* What is a channel in the context of Fabric and what is one advantage they provide?

□ 0
□
□ 1

c)* Which are the three types of nodes in a Fabric network? Also briefly describe their roles.

□ 0
□
□ 1
□
□ 2
□
□ 3

d) Which type of node in a Fabric network could best be compared to a Bitcoin miner? Briefly explain your answer.

□ 0
□
□ 1
□
□ 2

Problem 8 Corda (7 credits)

0 □ □ 1 □

a)* Corda has the concept of flows. What do nodes use flows for and what do flows enable?

0 □ □ 1 □

b)* What are CorDapps and how are they related to flows?

0 □ □ 1 □

c)* In the context of Corda, what is a state and where is state stored?

0 □ □ 1 □ □ 2 □ □ 3 □ □ 4 □

d)* Name two consensus types in Corda? Also explain what kinds of checks each consensus requires a transaction to pass.

Additional space for solutions–clearly mark the (sub)problem your answers are related to and strike out invalid solutions.