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	**Date:**	Monday 12th August, 2019
Examiner:	Prof. Dr. Florian Matthes	**Time:**	13:30 – 15:00

	P 1	P 2	P 3	P 4	P 5
I					

Working instructions

- This exam consists of **16 pages** with a total of **5 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:
  - one **analog dictionary** English ↔ native language
- Subproblems marked by * can be solved without results of previous subproblems.
- **Answers are only accepted if the solution approach is documented.** Give a reason for each answer unless explicitly stated otherwise in the respective subproblem.
- 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.

Left room from _____ to _____ / Early submission at _____

Problem 1 Cryptography (10 credits)

0 1

a)* Cryptographic hash functions are a a crucial building block of Blockchains. Describe the property **2nd preimage resistance** of cryptographic hash functions as explained in the lecture.

Alice and Bob are both running nodes in the Bitcoin blockchain. While Alice runs a full node, Bob only has a light-node, which does not store all Bitcoin blocks. Alice's node is connected to other nodes in the network and Bob's node is only connected to Alice's node.

0 1 2 3

b)* Bob runs an online store and accepts Bitcoin as payment. If a customer pays with Bitcoin, Bob is interested if this transaction is mined. He wants to know from Alice's full node if this transaction actually has been included in the most recent block. He decides to incorporate this transaction into a single Bloom filter and sends it to Alice. Alice receives this newly created Bloom filter from Bob. Explain shortly what Alice has to do to validate if this transaction is included in the most recent block.

0 1 2

c)* Without further information, is Alice able to see how many transactions Bob has inserted in the Bloom filter? Please provide a short explanation.

Alice now wants to prove to Bob that his transaction was indeed included into the latest block. **For the sake of simplicity, we only consider the Merkle-tree of all transactions**. Look at the provided Merkle tree in figure 1.1, it contains four transactions.

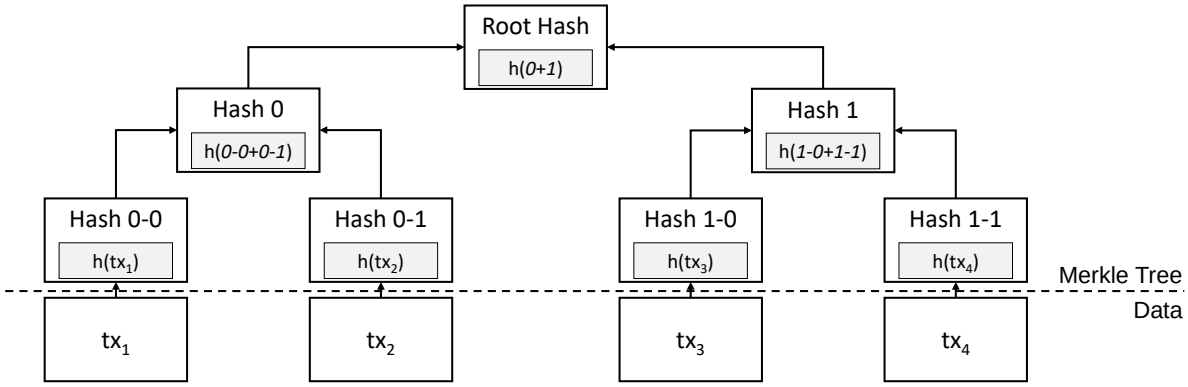

Figure 1.1: Merkle Tree

d)* Alice has the complete Merkle tree stored, Bob only knows the **Merkle tree root hash**, **his transaction, tx_3** and the **position of his transaction within the tree**. Name the data elements Alice has to send to Bob for him to be able to verify that his transaction is indeed included in the Root hash of this Merkle tree. Please ensure that only data that is absolutely necessary for verification is transmitted.

0
1
2

e) Describe each step Bob needs to do to verify that his transaction tx_3 is indeed included in the Root Hash.

0
1
2

Problem 2 Bitcoin (30 credits)

Bitcoin was introduced in 2008 as a purely decentralized P2P-cash system by Satoshi Nakamoto. The network relies on a Blockchain which keeps track of all transactions in the system.
In figure 2.1 the anatomy of a Bitcoin Block is depicted.

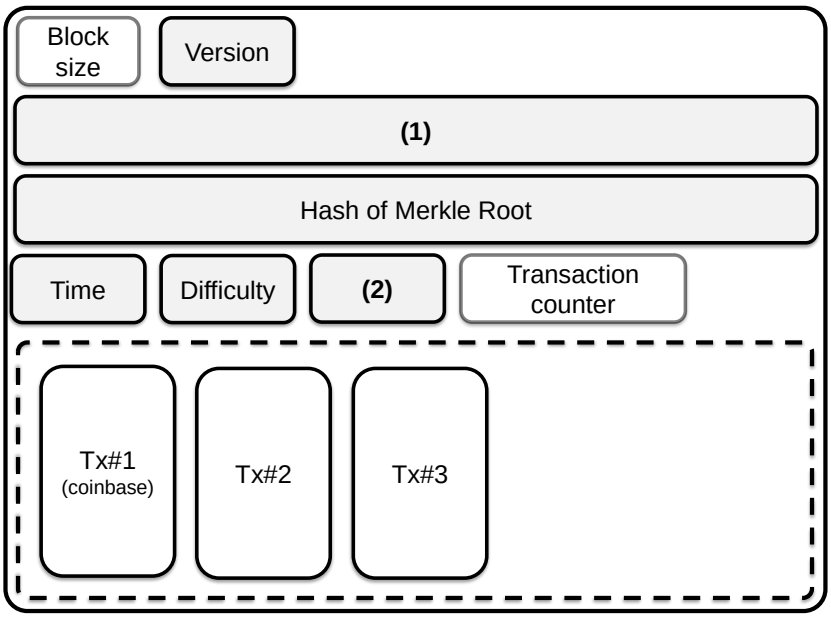

Figure 2.1: Anatomy of a Bitcoin block.

0 1 2 a)* Name the two missing fields in the block.

(1):

(2):

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 c) The difficulty is also part of the block header. Explain why the difficulty of the hash puzzle changes regularly.

The Blockchain uses a so-called transaction-based ledger. A transaction consists out of transaction inputs (Txin) and transaction outputs (Txout). The contents of both Txin and Txout are described in figure 2.2.

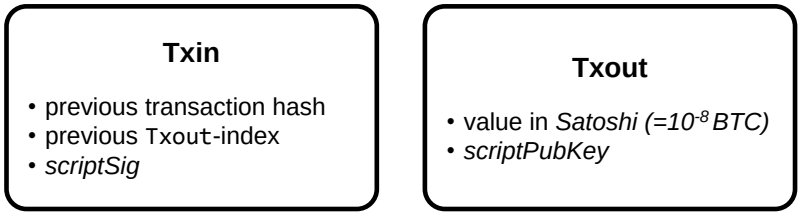

Figure 2.2: Transaction Input and Transaction Output

d)* First of all, what is the role of the Txout? Additionally describe what is stored in the *ScriptPubKey*.

0 1 2

e)* The Txin refers to a Txout contained in another transaction. Shortly explain the purpose of the transaction input and explain why the amount of Bitcoin **is not stored** within the Txin.

0 1 2

f)* Explain when a transaction output can be considered as an **Unspent Transaction Output (UTXO)**. In that context, what is the memory pool for?

0 1 2

g)* In some transactions, the *ScriptSig*-field of the Txin is empty and does not reference to a previous transaction output. How are such transactions called? What is their purpose?

0 1 2

Figure 2.3 contains four Bitcoin transactions. The left side refers to transaction inputs, the right side to transaction outputs.

Tx1	
Ø *(Txin)*	25 → Alice *(Txout)*

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

Tx3	
#2[0]	2 → Carol 6 → Dave

Tx4	
#2[2]	3 → Alice 2 → Dave

Figure 2.3: Four Bitcoin transactions.

0 1 2

h)* Which transaction out of transactions **2 to 4** is wrong? Explain.

0 1

i)* Who has signed Transaction 4?

0 1

j)* In transaction 2, the sum of all inputs is larger than the sum of all outputs. What happens to the remaining coins?

0 1 2

k)* Create a transaction in which Alice pays 12 Bitcoin to Tom. Ensure that Alice does not lose aditional money.

Figure 2.4 displays seven nodes interconnected with each other.

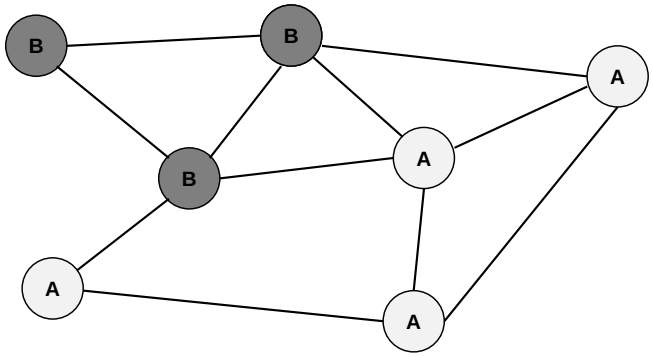

Figure 2.4: Nodes in a Bitcoin network

Two competing longest Blockchains exist within the network. They share the same history, but the highest block differs. An example is given in figure 2.5.

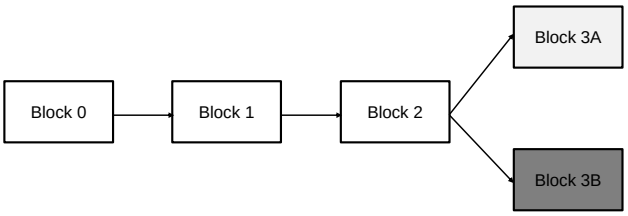

Figure 2.5: Network with two competing longest Blockchains

l) Explain how such scenarios occur. How is such a scenario resolved?

0
1
2

m) Assume block 3B is accepted by the whole network and becomes part of the longest chain. In this case, how is the block 3A called? What happens to the transactions in block 3A?

0
1
2

Consider the transaction output displayed in figure 2.6. This type of script is called P2PKH (Pay-to-public-key-hash). However, the OP_Code `OP_DUP` has been removed from the script.

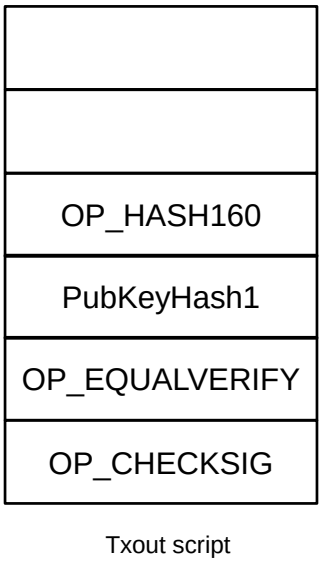

Figure 2.6: Transaction output script.

0 1 2 3 4 5 6

n)* Write a Transaction input script that, concatenated with the output script, correctly executes. For your convenience, we provide you with a table of sufficient OP_CODES for this exercise in figure 2.7. Please do not use other OP_CODES, the given ones are sufficient. If you want to use the <data> op-code, you can access additional information in figure 2.8. **Please ensure the correct ordering of your transaction input script. The Txin-script is executed from top to bottom.**

Your Txin script:

Opcode	Input	Output	Description
OP_DUP	<data>	<data><data>	Duplicates the top item on the stack
OP_NIP	<data>	-	Removes the second-to-top stack item.
OP_SWAP	<data1><data2>	<data2><data1>	The top two items on the stack are swapped.
OP_HASH160	<data>	H(<data>)	Removes the top item, hashes it, places it on top of the stack
<data>	-	<data>	Pushes <data> on top of the stack
OP_EQUALVERIFY	<data1><data2>	- or false	Checks if <data1> and <data2> are equal. If equal removes both of them, otherwise fails.
OP_CHECKSIG	<pubkey> <data>	false or true	Takes two items of the stack and validates whether the signature was created with the private key corresponding to the pubkey.

Figure 2.7: Available OP-Codes

User	Public key	Hash of Public key	Signature
Alice	PubKey1	PubKeyHash1	Sig1
Bob	PubKey2	PubKeyHash2	Sig2
Carol	PubKey3	PubKeyHash3	Sig3

Figure 2.8: User-specific data

o)* Name one other type of script for the transaction output and explain its purpose.

Problem 3 **Ethereum (18 credits)**

A miner created a block containing the following transactions:

Position	Sender	Recipient	Nonce	(1)	(2)	(3)	(4)	(5)
0	`0xaaaa...`	`0x5555...`	2,803	3 ETH	60,000	38,543	42 Gwei	`0x`
1	`0xcccc...`	`0xdddd...`	2,405	0 ETH	35,000	28,776	33 Gwei	`0xb76ea962ff...`
2	`0xdddd...`	`0x4444...`	76	6 ETH	100,000	21,000	20 Gwei	`0x`
3	`0x1111...`	`null`	12	0 ETH	93,000	92,830	10 Gwei	`0x6080604052...`
4	`0x4444...`	`0x3333...`	4	100 ETH	78,323	78,323	9 Gwei	`0xa9059cbb00...`

For brevity, some of the values in this table are abbreviated. Note: 1 ETH = 10^9 Gwei.

0 1 2

a)* Some of the headers in the table are missing. Write the five missing transaction field headers of the table above in the box below.

(1):

(2):

(3):

(4):

(5):

0 1 2

b)* What types of accounts exist in Ethereum and what are their differences?

0 1 2

c) What type is the account at address `0xdddd...`, which is involved in the transactions at position 1 and 2? Justify your answer.

0 1

d)* One of the transactions in this block failed because it ran out of gas. Which one? Why doesn't the sender get a refund of the gas they spent?

e)* How can you calculate the total transaction fees **in ETH** the miner got for this block? **Do not** calculate the exact value, just describe how to calculate it.

0 1 2

f) After which rationale did the miner choose the transactions for the block? Why is this choice a common procedure?

0 1 2

g)* Why does a block only list transactions, but not messages?

0 1

h)* Name and describe three use cases of smart contracts.

0 1 2 3

i)* Describe the process of deploying a smart contract to the blockchain.

0 1 2

Problem 4 Solidity (22 credits)

a)* What is a design pattern? Name one reason why patterns are specifically important when programming smart contracts.

b)* Describe a randomness pattern and state how the block hash is a solution for the problem.

c)* You want to program your own lottery in which the winner receives all stakes. You recall the randomness pattern from the BBSE lecture and decide to use the block hash to select the winner. Complete the `onlyOwner` modifier, the constructor and the three function bodies of the smart contract on the next page to implement the lottery with the following specifications:

- Any account can take part in the lottery by sending at least 1 ETH to the `payIn()` function.
- The same account can participate multiple times by calling the function again.
- The winner of the lottery is determined with the `selectWinner()` function. This function can only be executed once. It uses the block hash of the previous block to get a pseudo-random number and determine the winning account.
- The `onlyOwner` modifier ensures that only the creator of the contract can call the `selectWinner()` function.
- Finally, the winner can withdraw their prize by calling the `withdraw()` function.

You are allowed to use all functionality that Solidity provides. Some of the following code snippets may be useful:

- Block hash of a block (returns `bytes32`): `blockhash(...)`
- Current block number: `block.number`
- Sender of the transaction: `msg.sender`
- Amount sent with the transaction: `msg.value`
- Enforcing conditions: `require(...)`
- Append to array x: `x.push(...)`
- Length of array x: `x.length`
- Transfer assets: `recipient.transfer(...)`
- x mod y: `x % y`
- Unit literals: `ether`, `finney`, `wei`
- Casting arbitrary data to uint: `uint(...)`
- Empty address: `address(0)`

```
pragma solidity  ^0.5.1;

contract Lottery {

  address owner;
  uint pot;
  address[] players;
  address winner;



  modifier onlyOwner() {









  }

  constructor () public {









  }

  function payIn() public payable {

```

```
  }

  function selectWinner() public onlyOwner {

```

```
  }

  function withdraw() public {

```

```
  }
}
```

0 1

d)* Why is the approach of using the block hash as a source for randomness a potential security risk in this lottery?

0 1 2 3

e)* Name and describe two other solutions to create random numbers in Solidity.

Problem 5 **Hyperledger (10 credits)**

a)* Briefly describe two benefits and two disadvantages of a private blockchain system like Hyperledger Fabric compared to public Ethereum.

0 1 2 3 4

b)* Solidity is a programming language designed for Blockchain-based systems, e.g., it does not provide functions to create arbitrary random numbers. However, Hyperledger Fabric uses standard Go and Java to develop smart contracts, i.e., chaincode. Therefore, developers can easily develop methods to create arbitrary random numbers. Briefly describe how Fabric ensures that there is no indeterministic behavior in the system.

0 1 2 3 4

c)* How likely is it that 3 endorsing peers come to the same read-write set when a random number generator that outputs either 0 or 1, with a likelihood of 0.5 each, is used to set a state variable in the chaincode?

0 1 2

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