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 27th July, 2020
Examiner:	Prof. Dr. Florian Matthes	**Time:**	14:30 – 16:00

	P 1	P 2	P 3	P 4	P 5	P 6
I						

Working instructions

- This exam consists of **16 pages** with a total of **6 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 Cryptographic Basics (10 credits)

In March 2019, the German government published a call for participating in a consultation process for the German Blockchain strategy. In this document, the German government states that "*[...] there is not just 'the one', but different blockchains. Nevertheless, some basic principles of blockchain technology can be described: encryption [...]*"[1].

0 □ □ 1 □ □ 2 □

a)* Explain why encryption of essential data (blocks, transactions, ...) is contrary to the notion of public Blockchains.

0 □ □ 1 □ □ 2 □

b)* The German government missuses the term **encryption** and should rather use the term **asymmetric cryptography**. Asymmetric cryptography enables a scheme that is essential to Blockchains. Name the scheme and for what purpose it is used.

0 □ □ 1 □

c)* The security of asymmetric cryptography depends on the ability to keep a string to yourself. Name one (1) way to securely store this string.

Assume following **non-cryptographic** hash function[2]: $h(x) = x \mod 5$.

0 □ □ 1 □ □ 2 □ □ 3 □

d)* Name the two basic properties of hash functions as introduced in the lecture and explain why this hash function adheres to them.

0 □ □ 1 □ □ 2 □

e)* As $h(x)$ is a **non-cryptographic** hash function, it is also has no pre-image or collision resistance property. Give **numeric examples** (x and h(x)) which refute following properties:

	x_1	$h(x_1)$	x_2	$h(x_2)$
Second-preimage resistance	7	2		
Collision resistance				

[1] *Online-Konsultation zur Erarbeitung der Blockchain-Strategie der Bundesregierung*, modified to fit problem description, `https://www.bmwi.de/Redaktion/DE/Downloads/B/blockchain-strategie.pdf`, accessed 18th July 2020

[2] Modulo (mod) is a mathematical function that returns the remainder of a division of two integers.

Problem 2 **Bitcoin (29 credits)**

a)* Name two reasons for a transaction not beeing valid.

□ 0 □ 1 □ 2

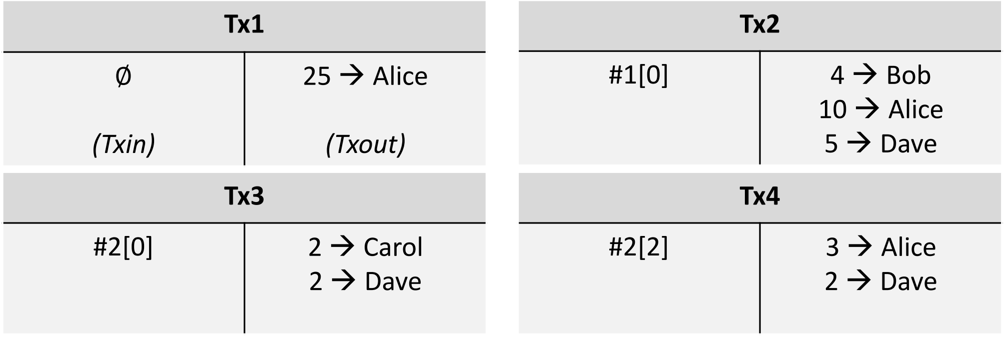
b)* You are tasked with building an e-commerce shop that accepts Bitcoin and sells physical goods with medium to high value that must be shipped. You want to build the shop from scratch without relying on third parties for payment. Zero-confirmation-transactions are transactions that are accepted as payment when the transaction has been broadcasted to the network but has not been confirmed yet.

Do you accept zero-confirmation-transactions? State your reasons. (1p)

What different node-types are there in Bitcoin that could be used for this use-case in the backend? Explain them and discuss which you would use. (3p)

□ 0 □ 1 □ 2 □ 3 □ 4

Take a look at following transactions. All of them have been included in a Bitcoin block.



Tx1	
∅	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 2 → Dave

Figure 2.1: Four Bitcoin transactions.

c)* Which entity has created transaction output `#2[0]`?

□ 0 □ 1

0 □ 1 □

d)* Which entity is allowed to spend the transaction output `#3[0]`?

0 □ 1 □ 2 □

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

Alice	Bob	Carol	Dave

Full nodes in Bitcoin need to keep track of the so called UTXO-set (set of unspent transaction outputs). As this set grows, the storage requirements for running a node increase. As these nodes only have to keep track of the UTXO-set (and not inputs), Bob proposes a simple, but effective way to decrease the size of P2PKH (Pay-to-Public Key hash) transactions.

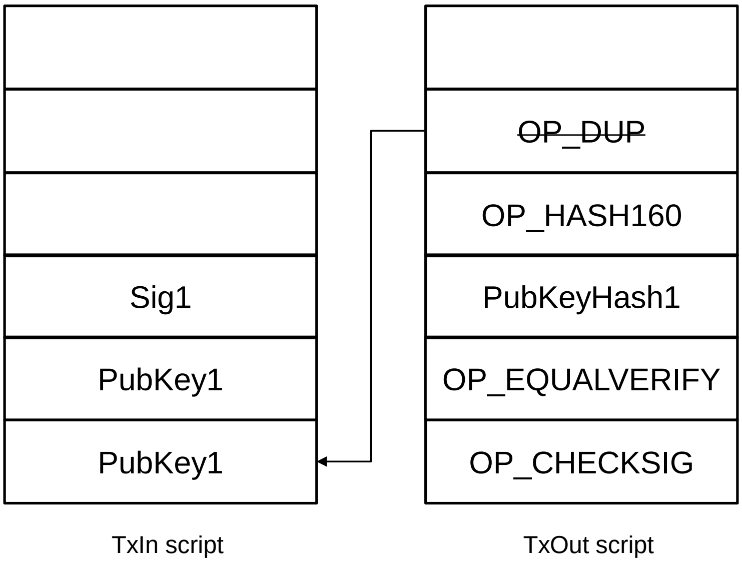


Figure 2.2: Instead of storing an `OP_DUP` in the UTXO, we provide the public key twice.

Instead of leaving the `OP_DUP` in the TxOutScript, he requires the user who wants to spend the output to provide the respective PublicKey twice. This requires less storage for the UTXO and works just fine.

f)* Explain why Bob's idea is bad and further describe a scenario in which an attacker could steal the funds secured within the UTXO.

□ 0 □ 1 □ 2 □ 3 □ 4

g)* What is the most energy efficient hardware to mine Bitcoin according to the lecture?

□ ASIC

□ FPGA

□ CPU

□ GPU

h)* How many hashes did the Bitcoin network generate per second on average throughout 2019 to today?

□ 10 - 200 Exa-hashes

□ 70 - 250 Kilo-hashes

□ 50 - 500 Terra-hashes

□ 25 - 250 Giga-hashes

i)* After the last halving (May 2020, mining reward decreased from 12.5 BTC to 6.25 BTC), the Bitcoin mempool was filled with many transactions and the average time a transaction needed to be confirmed was heavily increased. Also the blocktime was much higher than the targeted blocktime in the protocol.

What could have been the reason for this? Argue conclusively. (3p)

How was this problem finally resolved? (1p)

□ 0 □ 1 □ 2 □ 3 □ 4

j)* The supply of Bitcoin is capped to 21,000,000 bitcoins. Alice argues that the Bitcoin network requires an update to continue working after all Bitcoins are mined and proposes to remove halving to allow a continued increase of the bitcoin supply. Mention one downside to this approach and name an alternative approach.

□ 0 □ 1 □ 2

0 □ □ 1 □ □ 2 □ □ 3 □

k)* Give an example for changes to the Bitcoin core client that a) causes a soft fork, b) causes a hard fork and c) that has no impact to the consensus layer.

0 □ □ 1 □ □ 2 □ □ 3 □ □ 4 □

l)* Since the creation of Bitcoin there have been multiple chain splits. A few of the most famous ones resulted in Bitcoin Cash, Bitcoin Gold and Bitcoin SV.

How does a chain split happen? (2p)

Explain the term replay attack and explain how it can be prevented. (2p)

Problem 3 Ethereum (18 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 3.1.

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 3.1: Block containing a transaction.

Furthermore, this transaction leads to several messages displayed in table 3.2.

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 3.2: Resulting messages of transaction 5.

a)* The first thing to notice is that the columns `Nonce` and `GAS PRICE` are not mentioned in the messages table. Explain why both of these parameters are not required in messages.

□ 0 □ 1 □ 2

(Nonce):

(GAS PRICE):

b)* Decide for the following addresses the type of account and give a **short** explanation. State if the type is undecidable.

□ 0 □ 1 □ 2 □ 3 □ 4

(0xaaa...):

(0xbbb...):

(0xddd...):

c)* The account `0xccc` is a Smart Contract and is called in message 0 in table 3.2. Within this contract, two address-variables are accessed: `msg.sender` and `tx.origin`. Resolve these two variables to their respective address (0x...) in message 0 executed in 0xccc.

□ 0 □ 1 □ 2

(`msg.sender`):

(`tx.origin`):

Alice (0xAlice...) wants to transfer two tokens (managed in contract 0xyyy...) to Bob (0xBob...).

0 □ □ 1 □ □ 2 □ □ 3 □ □ 4 □ □ 5 □

d)* Name following parameters for a transaction in which Alice transfers two coins 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[3]. 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

0 □ □ 1 □ □ 2 □

e)* One special type of ERC20 token is the Ether-backed ERC20 token. A token represents exactly 1 Ether; it can be created or claimed by either depositing or withdrawing Ether from this token contract. Name one (1) advantage and one (1) disadvantage of such Ether-backed token (contract) in comparison to regular Ether.

0 □ □ 1 □ □ 2 □ □ 3 □

f) 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 wants to tokenize real estate (houses, ...) and domain expert B is interested in tokenizing energy (kW/h).

(A):

(B):

[3] ERC Token Standard #20 Interface, `https://eips.ethereum.org/EIPS/eip-20`, retrieved from `https://de.bitcoinwiki.org/wiki/ERC20` on 19th July 2020

For your convenience, this page is left intentionally blank.

Problem 4 Solidity (13 credits)

An auction house of rare items wants to provide more transparency to their bidders. Using the Ethereum public blockchain as a means for transactions and decentralised logic, anyone could audit that the bidders and their bids correspond to the real-world events. The auction house has hired you to develop a set of solidity functions that comply with the three programming idioms you have learned during the lecture: Access restriction, secure Ether transfer, and safe arithmetic.

Complete the modifier by implementing the access restriction idiom, give it a name, and include it in the function definitions you see fit. Additionally, complete the constructor. Lastly, implement only the function bodies of `bid()` and `withdraw()` of the smart contract on the next page. These two functions implement the secure Ether transfer idiom. The next specifications must be followed:

- Any account can take part in the auction.
- The same account can bid multiple times by calling the `bid()` function. Ensure that the basic requirement to execute a bid is met. This function should update the previous bidder's `accountBalances` and the `auctioned_item`. Remember to use the safe arithmetic idiom.
- A user can withdraw their current balance with the `withdraw()` function. Ensure that the requirement to execute this function is met, that `accountBalances` is updated, and that the withdrawal is executed.
- Only the contract owner can execute `setAuctionedItem()` and `finishAuction()`.
- Ensure that the bidders can only withdraw the correct amount of Ether after being outbid.

You are allowed to use all functionality that Solidity provides. 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(...)`
- Empty address: `address(0)`
- Returns Ether balance of the contract: `address(this).balance`
- Prevent over-/underflow of integers: `uint var; var.add(..); var.sub(..); var.mul(..); var.div(..)`

Convert the instructions on the previous page into code.

□ 0
□ 1
□ 2
□ 3
□ 4
□ 5
□ 6
□ 7
□ 8
□ 9
□ 10
□ 11
□ 12
□ 13

```
pragma solidity >=0.4.21 <0.7.0;

import "./SafeMath.sol";

contract Auction {

    using SafeMath for uint256;

    address payable owner;

    struct item {
      string item_name;
      uint price;
      address payable highest_bidder;
    }

    item public auctioned_item;

    mapping(address => uint) public accountBalances;

    modifier
    {




    }

    constructor () public
    {



    }


    function setAuctionedItem(string memory _item, uint _price) public
    {

       auctioned_item = item(_item, _price, address(0));
    }

    function bid() public payable
    {












   }
```

```
    function withdraw() public
    {









    }

    function finishAuction () public
    {

       owner.transfer(auctioned_item.price);
    }

}
```

Problem 5 **Hyperledger (10 credits)**

a)* When is consensus considered reached in Hyperledger Fabric?

□ 0 □ 1 □ 2

b)* Check whether the following statements are true or false. If they are false correct them.

□ 0 □ 1 □ 2 □ 3 □ 4 □ 5 □ 6 □ 7 □ 8

Statement	**True**	**False**	**Reason (if false)**
Hyperledger Fabric uses a validate-order-execute architecture			
Each peer only shares the ledger with nodes that are in the same channel			
Participants are incentivized with cryptocurrency for participating in Hyperledger Fabric			
Hyperledger Fabric has been deprecated			
The ordering service packages transactions into blocks			
Everyone can join a Hyperledger Fabric network and participate with a peer node without invitation			
Peers can have two different roles in Hyperledger Fabric: endorsing peer or validating peer			
The MSP is responsible for access control to the network			
Native chaincode can be written in Go, NodeJS and Java			
A ledger in Hyperledger Fabric consists of a world state and a blockchain			

Problem 6 Corda (10 credits)

0 □ □ 1 □

a)* Why is Corda not considered a Blockchain, but rather a Distributed Ledger Technology? Name and describe one reason.

0 □ □ 1 □ □ 2 □ □ 3 □

b)* Name and briefly describe the three services offered by a Corda Network.

0 □ □ 1 □ □ 2 □

c)* When is a transaction in Corda considered valid?

0 □ □ 1 □

d) How is a ledger update coordinated in Corda? Name and briefly describe the concept.

0 □ □ 1 □ □ 2 □ □ 3 □

e)* Define the term CorDapp. Name four out of five components of a CorDapp.

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