Cryptocurrencies (WS 2018) Univ-Prof. Dr. Matteo Maffei

Final Exam

Date: 24/01/2019 TU Wien

Name: ***Matriculation Number:***

- **DO NOT OPEN** the exam until instructed to do so. Read all the instructions first.
- After you have opened the exam booklet, it is your obligation to check whether it is complete. The exam booklet must contain 10 pages.
- The exam is closed-book, closed-notes. No auxiliary means are allowed. At your desk, you may only have writing utensils, beverages, food, ID cards, and an English dictionary. Bags and jackets have to be left at the walls of the lecture room, mobile phones and computers need to be switched off.
- The exam takes 90 minutes. You can get at most 50 points. The number of points you can get for an exercise thus gives you a hint about how much time you should spend on that exercise. You can count 1 minute per point and then you have 40 minutes left to check your answers again.
- **Write legibly and be concise.** It is in your best interest that we understand your answers. You will be graded not only on the correctness of your answer, but also on the clarity with which you express it.
- Use an unerasable pen for filling in your answers.
- If you need to go to the bathroom during the exam, please turn in your exam booklet. Only one person may go to the bathroom at a time.
- Good luck!

Problem	1	2	3	4	**Total**
Score					
Points	16	10	12	12	50

Problem 1 (Multiple Choice) is missing due to legal reasons.

Problem 2: Bidirectional payment channels (10 points)

The pseudocode[1] shown at the bottom of this page represents a smart contract (e.g., state channel in Ethereum) to handle bidirectional payment channels between two users. Assume that Alice (with key pair sk_A, pk_A) and Bob (with key pair sk_B, pk_B) have already opened a payment channel using this smart contract and both Alice and Bob have locked x_A and x_B coins, correspondingly. That is, they have collaborated to create an initial state *sInitial* (shown below) and have called the operation *openChannel* on input *sInitial*.

```
1  state sInitial {
2   balanceAlice = x_A
    balanceBob = x_B
4   signatureAlice = σ_A      σ_A := Sign(sk_A, balanceAlice || balanceBob)
5   signatureBob = σ_B        σ_B := Sign(sk_B, balanceAlice || balanceBob)
6  }

1  #Definition of off-chain state
2  struct state {
3    balanceAlice
4    balanceBob
5    signatureAlice
6    signatureBob
7  }
8
9  #Global variables
10 coinsLockedAlice = 0
11 coinsLockedBob = 0
12 channelOpen = False
13 timeout = 31/01/2019
14
15 def openChannel(state):
16   if channelOpen == True then abort()
17   if  Verify(pk_A, state, state.signatureAlice) == False OR  Verify(pk_B,
       state, state.signatureBob) == False then abort()
18   coinsLockedAlice = state.balanceAlice
19   coinsLockedBob = state.balanceBob
20   channelOpen = True
21
22 def closeChannel(state):
23   if channelOpen == False then abort()
24   if  Verify(pk_A, state, state.signatureAlice) == False OR  Verify(pk_B,
       state, state.signatureBob) == False then abort()
25   if (state.balanceAlice + state.balanceBob) != (coinsLockedAlice+ coinsLockedBob) then
       abort() // != denotes "not equal"
26   if expired(timeout) = True then
27     send(Alice, coinsLockedAlice)
28     send(Bob, coinsLockedBob)
29   else then
30     send(Alice, state.balanceAlice)
31     send(Bob, state.balanceBob)
32   channelOpen = False
```

Exercises

1. **(4 points)** Show an attack where Alice can receive a good from Bob worth 1 coin and yet get back all her x_A coins before the channel's timeout expires. For this, you should describe the following:
   (a) The state that Alice has to send to Bob so that Bob accepts it as valid off-chain channel update. You must specify the value of each state's field and how Alice can calculate it.
   (b) The state that Alice uses to get all her x_A coins back before the channel's timeout expires. You must specify the value of each state's field and how Alice can calculate it.
2. **(6 points)**Propose a fix to solve this attack. To this end, you should describe:
   (a) What modifications are required in the code of the smart contract
   (b) Explain how the proposed modifications prevents the above attack

[1]Please note that the pseudocode does not follow the syntax of Solidity.

Problem 3: Execution paradigms & transaction lifetime (12 points)

Consider the lifetime of a transaction as the set of entities in a blockchain technology that must process such transaction between the moment in which the client creates it until the transaction finally appears in the blockchain. In the lecture, we have seen two different execution paradigms during the lifetime of a transaction, namely, *order-execute* paradigm (e.g., as used in Bitcoin) and *execute-order-validate* (e.g., as used in Hyperledger Fabric).

For this problem, assume an architecture composed of the following entities: two clients (C_1 and C_2), two peers/miners (P_1 and P_2) and one ordering service (O_1). Further assume that C_1 has 10 coins and wants to send a transaction tx_1 that sends all 10 coins to C_2. Further assume that C_2 has 0 coins and he wants to submit a transaction tx_2 that sends 10 coins back to C_1. Finally, assume that both clients send their transactions simultaneously at time t_0. Further assume that every block contains only a single transaction.

Exercises

1. **(2 points)** Fill the time diagram below specifying one of the possible lifetimes of both transactions tx_1 and tx_2 in each execution paradigm. For simplicity, assume that there is no latency, that is, if entity E_1 sends the transaction at time t_i it arrives to the next entity E_2 at the next time t_{i+1}. For example, in both execution paradigms, the first steps in the lifecycle of tx_1 consists of C_1 creating the transaction (t_0) and then sending the transaction to one of the peers (e.g., P_1) at t_1. This has been shown already in the time diagram. Use the symbol B to denote the time slot at which a block is published by a peer. Note that there can be idle times in a transaction lifetime (e.g., a peer does not create a block immediately).

		t_0	t_1	t_2	t_3	t_4	t_5	t_6	t_7	t_8	t_9
Order-execute	tx_1	C_1	P_1								
	tx_2	C_2									
Execute-order-validate	tx_1	C_1	P_1								
	tx_2	C_2									

2. **(2 points)** Describe what is the final balance (i.e., number of coins) owned by each client at the end of your suggested transaction lifetimes for *order-execute* paradigm. Argue why is the case.

3. **(2 points)** Describe what is the final balance (i.e., number of coins) owned by each client at the end of your suggested transaction lifetimes for *execute-order-validate* paradigm. Argue why is the case.

4. **(6 points)** Although it does not exist in current blockchain technologies, for this exercise assume that there exists a function random() that returns a number chosen uniformly at random. Further assume a transaction tx^* that calls the function *foo* shown below and answer the following questions:

   (a) Assume the *order-execute* paradigm. Explain what is the problem if tx^* is added to the blockchain.

   (b) Assume the *execute-order-validate* paradigm. Would the same problem exist? Argument your answer.

```
1 def foo() {
2   r := random() //random returns a number chosen uniformly at random
3
4   if r mod 2 == 0 then
5     send(Alice, 1 coin)
6   else then
7     send(Bob, 1 coin)
8 }
```

Problem 4: S&P in payment-channel networks (12 points)

As seen in the lecture, current payment-channel networks use the hash time-lock contract (HTLC). In particular, we denote by $HTLC(Alice, Bob, x, y, t)$ a HTLC between Alice and Bob. HTLC is used to build multi-hop payments between any two users connected though a a path of opened payment channels with enough balance. We have further studied the security and privacy notions of interest for this type of networks. In particular, relationship anonymity (one of the privacy notions) is intuitively defined as follows:

Relationship Anonymity: Given two simultaneous successful payment operations of the form

$$\{\mathsf{payment}(path := (c_{\langle s_i, u_1\rangle}, c_{\langle u_1, u_2\rangle}, \ldots, c_{\langle u_{n-1}, u_n\rangle}, c_{\langle u_n, r_i\rangle}), value := v\}_{i\in[0,1]}$$

, if there exists at least one honest intermediate user u_j with $j \in [1, n]$, then corrupted intermediate users cannot determine the pair (s_i, r_i) for a given $\mathsf{payment}_i$ with probability better than 1/2.

ZK-HTLC: For the purpose of this problem, assume that there exists a zero-knowledge proof ZK-HTLC $(H(x), H(x+r))$ that intuitively proves that the pre-image of $H(x+r)$ is the same as the preimage of $H(x)$ only offset by a value r (here H denotes a hash function). In a bit more detail:

- Public values: $H(x)$, $H(x+r)$, r
- Private value: x
- Statement: Let a be the pre-image of $H(x)$. Let b be the pre-image of $H(x+r)$. Then, $\exists x \mid b-a = r$.

(**Hint**: *You require ZK-HTLC only for exercise 4*)

Exercises:

1. **(2 points)** Describe how HTLC is used in a multi-hop payment between sender s, receiver r connected through a path of opened payment channels of the form $s \to u_1 \to u_2 \to u_3 \to r$, where u_1, u_2 and u_3 are three intermediate users. For that, you should describe:
   - The inputs of the HTLC contract at each payment channel
   - How the users controlling each payment channel can get those inputs
   - In which order the different HTLC contracts are applied to *successfully* perform the multi-hop payment

   Please, note that you should consider both, the lock and unlock phases for the multi-hop payment.

2. **(2 points)** Assume that the adversary controls u_1 and u_3. Describe how the *wormhole attack* works. In particular, you should describe:
   (a) The inputs of the HTLC contract at each payment channel, how the users controlling each payment channel can get those inputs and in which order the different HTLC contracts are applied to perform the attack. You can refer to your answer in exercise 1 for repeated steps.
   (b) What is the economic benefit of the adversary doing this attack
   (c) Why the victim user does not realize that is under attack

3. **(2 points)** Assume that the adversary controls u_1 and u_3. Describe an attack against relationship anonymity. In particular, you should describe:
   (a) The inputs of the HTLC contract at each payment channel, how the users controlling each payment channel can get those inputs and in which order the different HTLC contracts are applied to perform the attack. You can refer to your answer in exercise 1 for repeated steps.
   (b) What is the probability of the adversary to determine the pair (s_i, r_i) for a given $\mathsf{payment}_i$. Argument your answer.

4. **(6 points)** Describe how to use ZK-HTLC in conjunction with HTLC to secure a payment between sender s, receiver r connected through a single intermediary u (i.e., in a path of the form $s \to u \to r$). You should describe:

   (a) The inputs of the HTLC contract at each payment channel

   (b) How the users controlling each payment channel can get those inputs

   (c) In which order the different HTLC contracts are applied to *successfully* perform the multi-hop payment

   Additionally, intuitively argue why this construction leads to multi-hop payments that achieve the relationship anonymity property.