Cryptocurrencies (WS2023/24) Dr. Zeta Avarikioti, Prof. Matteo Maffei

Kerma - Task 6b

1 General Notes

In your group of up to three students, you may use any programming language of your choice to implement the following task. We provide skeleton projects in Python and TypeScript (see `python-skeleton-for-task-6b.tgz` and `typescript-skeleton-for-task-6b.tgz` on TUWEL). You can use them to build upon. After the deadline, these skeleton projects will be updated and act as sample solutions for the previous tasks. These are guaranteed to pass all test cases for the previous tasks.

2 Bonus Points for this Optional Task

This is an optional additional task that may grant you bonus points. The "real" 6th task is called "Task 6a - Bitcoin Applications" (you can find it on TUWEL) and is weighted equally to the previous five Kerma tasks. This 6th Kerma task however is worth at most half of the points of a single task (see the last section for the overall grading schema). Your points obtained in this task will be added to your overall exercise points, **but do not carry over to the exam part of your grade**. You cannot receive more than 100% on the exercise part. You may use these bonus points to satisfy the necessary condition of having at least half of the points from the exercises.

3 High level Overview

After completing this task, your node should be able to:

- Mine transactions into blocks
- Collect mining fees

At the end of this task, your node should implement not only the complete protocol, but should act as a full mining node.

4 Transaction Builder

We highly recommend that you create a tool that allows you to easily create transactions. For instance, it can be a separate client application that takes as the following as inputs and produces valid transactions:

- Transaction inputs which identify unspent outputs and fund the transaction

- The private keys needed to create valid signatures for transaction inputs
- Transaction outputs, comprising the amount of coins held by the output and the public key of the new owner
- The mining fee to use

Usually, the inputs you use carry more coins than you want to transfer in your outputs, so your tool should automatically create an additional output that transfers coins in excess back to you.

5 Implementing your miner

In this task, you will implement an efficient miner that produces valid blocks extending the longest chain. You might want to

- Use lower-level programming languages (e.g., by implementing your miner in C [*] and letting it run as a subprocess of your node)
- Let multiple miners run in parallel to utilize multi-core architectures

You listen for transactions on the network to put in your mining pool, so you can include them in a block. Your miner can be very simple: It receives as input the canonical encoding of the block you want to mine. The miner will then iteratively change the nonce field until a nonce fulfilling the PoW condition is found, which it will then output. If your mempool is updated, you should update the transactions in your block as well - and restart your miner with the new block. Similarly, if your chaintip changes because a new block has been found, you should update your mempool, the transactions in the block you want to mine, as well as the pointer to the parent block in your block. If your miner found a new block, treat it as if you would have received it from the network - validate it (by construction it should be valid, but it won't hurt to re-check), store and gossip it. Recall that changing the block you want to mine between block arrivals will not increase your average block creation time because of "losing progress" - the probability that you find a satisfying nonce in the next iteration is the same for every block.

In the feedback for Task 5, every group received an ed25519 key pair (private key and public key (which we call "group address")) and a public key "target address". If your group did not submit any solution for task 5, please contact me per mail.[†] A valid block belongs to your group when it includes a coinbase transaction whose output gives money to your group address. Your overall task is to mine blocks that create and collect coins, store it in outputs belonging to your group and then create transactions that transfer these funds to the "target address". **Do not take a shortcut and directly use your target address in coinbase transactions.**

[*] https://github.com/BLAKE2/BLAKE2

[†] dominik.apel@tuwien.ac.at

6 Mining Strategy

You can choose the mining strategy you prefer: for instance, when identifying transactions to include in a block, you might want to prioritize transactions paying higher fees.

7 Grading

While we still require you to submit your implementation, **grading will not be done locally**. Right after the deadline is due, we will take a snapshot of the longest chain seen by the bootstrap node and we will analyze it. This task can give you maximum 10 points. You will be awarded points as follows:

- You will receive points depending on the number of blocks belonging to your group in the longest chain, i.e., blocks that contain a coinbase transaction with your group's public key as address. The more blocks, the more points you will receive: Let n be the number of blocks in the longest chain belonging to you. You will receive

  $$min(5, 5 - \lfloor \frac{5}{1.06^n} \rfloor)$$

  points.

- You will receive points depending on the sum of Ker held by unspent outputs **of non-coinbase transactions** belonging to your target. The more funds your target has, the more points you will receive: Let s be this sum of coins (in Ker, recall that 1 Ker is 10^{12} picaker). You will receive

  $$min(5, 5 - \lfloor \frac{5}{1.06^{\frac{s}{51}}} \rfloor)$$

  points. This means you need to create transaction(s) that transfer the coins you received by mining to the target account. It is not sufficient to just use the target public key in a coinbase transaction.

Note that only the blocks in the longest chain will be taken into consideration during grading. The best way to ensure that blocks you mine will not be pushed out of the chain is to keep mining on top of the longest chain, so a malicious agent will not be able to rob you of your points.

8 Grade Calculation

This section describes how your final grade will be calculated. The projects consist of 6 tasks, plus this extra optional one. The tasks $1, 2, 3, 4, 5, 6a$ are weighted equally, while $6b$ may grant bonus points worth half of the points of a normal task. This means that you can calculate the percentage of your overall grade using the following formula (provided you satisfy all passing

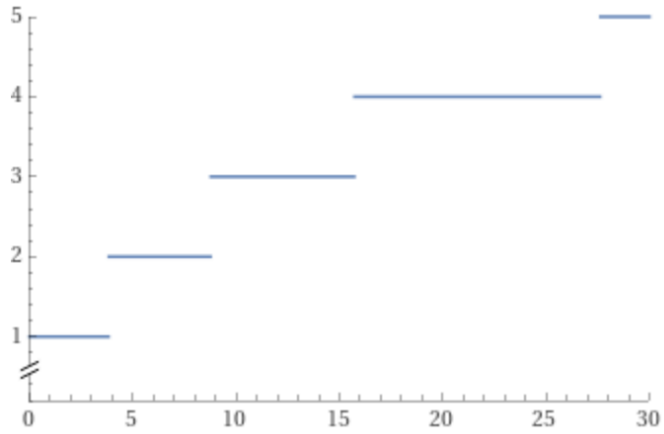

Figure 1: Points obtained per block count

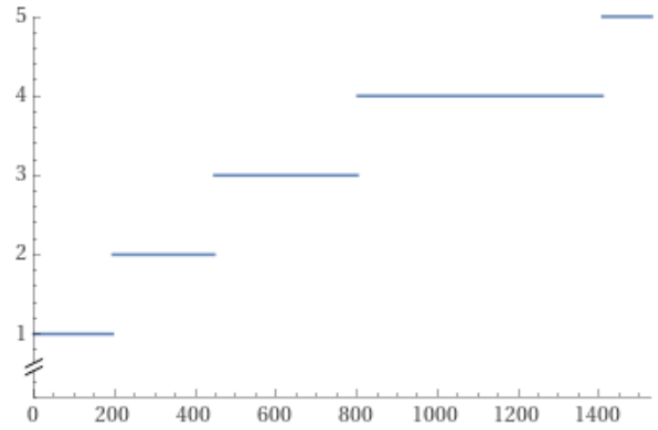

Figure 2: Points obtained per Ker belonging to your target account

criteria):
Let p_i be the points obtained in (regular) task i, m_i the maximum points of (regular) task i, e_i the points obtained in the exam i and n_i the maximum points in exam i. Let p_{6b} be the points obtained in this task and m_{6b} the maximal points of this task. Then your overall grade percentage g can be calculated by:

$$p = \frac{1}{12} \cdot (\sum_{i=1}^{6} \frac{p_i}{m_i}) + \frac{1}{24} \cdot \frac{p_{6b}}{m_{6b}}$$
$$g = \frac{1}{4} \cdot (\frac{e_1}{n_1} + \frac{e_2}{n_2}) + \min(0.5, p)$$

The passing criteria:

- $\frac{e_1}{n_1} \geq 0.5$
- $\frac{e_2}{n_2} \geq 0.5$
- $p \geq 0.25$

Depending on g, your final grade will be:

- 5 "Nicht genügend", if $g < 0.5$ or one of the required passing criteria not met
- 4 "Genügend", if $g \geq 0.5$
- 3 "Befriedigend", if $g \geq 0.63$
- 2 "Gut", if $g \geq 0.75$
- 1 "Sehr Gut", if $g \geq 0.88$

The deadline for this task is 16th January, 11.59pm. We will not accept any submissions after that. One submission per group is sufficient. Plagiarism is unacceptable. If you are caught handing in someone else's code, you will receive zero points.