University of Chicago
ECON2340/CMSC23280: Cryptocurrencies, Winter 2019
David Cash, Harald Uhlig, and Ben Zhao

Assignment 2: Understanding Consensus

Due at 11:59pm Wednesday, January 23rd

Introduction

In this assignment, you will implement crucial pieces of verification and consensus for the GoodCoin. In particular, you will write functions to verify transaction signatures, update the proof of work, and reach consensus on a blockchain version among a distributed set of nodes.

Skeleton code for these functions has been provided. Based on the instructions below and the material taught in class, you will implement these functions. Your work will be critical to ensuring that the GoodCoin is indeed a valid cryptocurrency!

To complete this assignment, you will download an updated version of the GoodCoin from Canvas. This new version now allows users of the GoodCoin to sign transactions, which will prevent arbitrary coin theft, and establishes a protocol for communication between nodes. This communication protocol will allow more robust distributed consensus to take place.

As mentioned previously, the GoodCoin has been written fresh for this class, so please alert the TAs if you find significant bugs. Before getting to the questions, we discuss the rules for this assignment and how you can access the code. We welcome your constructive feedback on how to make this code more user-friendly and a better teaching tool.

Rules

Collaboration policy. Please respect the following collaboration policy: You may discuss problems with up to 3 other students in the class, *but you must write up your own responses. You should never see your collaborators' writing.* At the beginning of your submission write-up, you must indicate the names of your (1, 2, or 3) collaborators, if any. You may switch groups between assignments but not within the same assignment.

Sources. Cite any sources you use. You may Google liberally to learn basic Python.

Piazza. We encourage you to post questions on Piazza. If you have a question that you believe will reveal secrets you have discovered while working on the assignment, post privately to just the instructors. If you have a question that you believe will be of general interest or clarifies the assignment, please post publicly. If you are uncertain, post privately; we will make public posts that we believe are of general interest.

Grading. Responses will be graded for correctness and clarity.

Assignment Tech Set-Up and Overview

All the questions for this assignment can be answered by using the GoodCoin blockchain application, exploring the GoodCoin code, and drawing upon material from class lectures.

Setting up Python

The GoodCoin is written in Python 3, so you will need to have Python 3 installed on your computer in order to use it. Most computers are shipped with a pre-installed version of at least Python 2, but if you need to get Python 3 set up, the guide found at `https://realpython.com/installing-python/` will be helpful.

Accessing the GoodCoin

Download the goodcoin.tar file from Canvas, place it in a convenient directory, and untar it (run "tar -xvf goodcoin.tar" if you are using a Linux command line). NOTE: for this assignment, the GoodCoin has been significantly updated, so do not skip this step of re-downloading the code.

Running the GoodCoin

The README.org file included with the GoodCoin code should have enough information to get you started. You may need to "pip install" (`https://ehmatthes.github.io/pcc/chapter_12/installing_pip.html`) several of the Python packages required for the application if you haven't done so previously.

There are new python packages in this iteration of the GoodCoin. To install them, cd into the goodcoin folder and run `python3 setup.py install`. This will ensure you have all the necessary packages installed in order to run the GoodCoin.

What and How to Submit

You will submit two files for this assignment. The first will be your updated version of the blockchain.py file (where the functions to be implemented can be found). This file should be renamed `<YOUR CNETID>-blockchain.py`. The second file, entitled `<YOUR CNETID>-assignment1.pdf/txt`, will contain responses in English to the questions asked below. Both files will be uploaded to Canvas.

Instructions

After downloading the code and installing the required packages, spend time playing around with the GoodCoin, especially the elements new to this version. Explore the code, mine blocks, conduct transactions, and see what you can do. Once you have spent sufficient time understanding how the signature- and consensus-enabled GoodCoin works, you will be ready to start the assignment. Note that the relevant code for this assignment lives in blockchain.py. The contents of server.py and messenger.py could be helpful in understanding the mechanics of the distributed consensus protocol, but you will not need to modify them in order to complete the assignment. The other files in the repository (besides the README.org file) are necessary to the functioning of the application but not neccessary to your understanding of the GoodCoin and consequently can be left alone.

Tasks

You will complete three tasks in this assignment. These are described below. For each task, you will write code (in blockchain.py) and answer related questions. The format of your eventual submission is described above.

Signature Validation (20 points)

This new version of the GoodCoin now features transaction signatures! In this part of the assignment, you will implement a function to check whether or not the signature over a transaction is valid. Recall that signing a transaction (as the `sign_tx` function in blockchain.py illustrates) involves decoding the public key that is embedded in the addresses of the transaction inputs, providing a secret/private key that will sign the transaction, and then generating the signature. In the GoodCoin, signatures are implemented using the python-ecdsa library. More information about this library can be found online at https://github.com/warner/python-ecdsa.

For this part of the assignment, you must implement the details of the `verify_signature` function in blockchain.py. This will involve extracting the public key and the signature from the transaction of interest (some decoding or transformation of these elements may be necessary after you extract them) and then performing the verification according to the API of the python-ecdsa library. All the necessary libraries and functions to do this already exist in blockchain.py – you simply need to bring them together. Make sure to test your implementation on real transactions to ensure it works.

Proof of Work (30 points)

In this section, you will modify the proof of work for the GoodCoin. This question is more open-ended. Your task is to modify the `valid_proof` (and potentially the `proof_of_work`) function(s) to create new criteria for what constitutes a valid proof of work. You may be as creative as you would like (within reason – what you write must work in the GoodCoin ecosystem). Options include, among others, using a new hash function or implementing a new puzzle for miners to solve. The only two stipulations are:

1. The proof of work can be adjusted to accomodate more nodes mining in the network (i.e. like how in the current version of the GoodCoin, more zeros can be required in a valid proof in order to increase proof of work difficulty).

2. The proof must be at least as difficult as the current, naive proof version in the GoodCoin (i.e. SHA256 with four leading zeros).

You may find it helpful to explore the proofs of work used in other blockchain systems, such as Dash, ZCash, and Monero. Be sure to cite any sources used in your work.

You may NOT simply modify the required leading characters in the `valid_proof` function. Submissions that leave everything else untouched but change the "0000" to "abcd" or something similar will receive no points.

In addition to the modified functions indicated above, please provide a short explanation in your writeup describing why you made your chosen modifications and listing at least one benefit and one drawback of using your new proof of work rather than the one originally provided, especially in the context of a many-miner situation. Additionally, please clearly note any external libraries your new proof-of-work relies upon in order to make grading easier for the TAs.

Consensus (50 points)

Finally, the GoodCoin community has tasked you with implementing an additional criterion to help nodes decide which version of the blockchain is valid. Currently, GoodCoin nodes consider the longest version of the blockchain to be valid and will replace their own version of the chain with any longer chain they encounter. However, the GoodCoin community has (rightly) recognized that this is insufficient. You have been tasked with adding an additional criterion to the `consensus` function in blockchain.py.

This criterion will choose as the valid chain the longest chain with the longest common prefix to its current chain. For example, let node A's chain be $b1 \leftarrow b2 \leftarrow b3$ (where each b represents a block), node B's be $b1 \leftarrow b2 \leftarrow b3$, and node C's be $b1 \leftarrow b3 \leftarrow b4 \leftarrow b5$. When nodes A, B, and C try to reach agreement about which chain is correct, node C's chain will be ignored (under the new criterion) in favor of A's and B's because node A and B have a longer common prefix.

However, if node C's chain is $b1 \leftarrow b2 \leftarrow b3 \leftarrow b4 \leftarrow b5$, then because it is both the longest and has the longest common prefix it will be recognized as the authoritative chain by nodes A and B.

Building on the existing code in `consensus`, implement this additional criterion and use both criteria to reach chain consensus. Additionally, include in your writeup a short description of how you tackled this problem and any difficulties you encountered along the way.