ECE 579: Blockchain and Cryptocurrencies 2020

Assignment 3

In this assignment you need to implement Proof of Work functionality for a given blockchain application using Python. The implementation should adjust the difficulty of the PoW to match a given time interval, e.g. if the time interval is given as 10 seconds, the creation of a new block should be around 10 seconds. We provide most of the Python code and commented it. You need to complete 4 functions in order PoW code to work. The code will create empty blocks (in the code it is 32 blocks, you can decrease it for tests) and measure the time it creates a single block on average. Later, it uses the computation time and the targeted block creation time to adjust the difficulty.

If you want the average block creation time to be 2 seconds, even if you code it correctly, you will probably achieve a number between 1.5-3 seconds. It is perfectly normal given that we have a small runtime. As long as the timing is close enough, your code is probably correct. For instance; in our test for 10 second interval, we saw timings between 8 to 15 seconds.

Once the code runs correctly, it will output a json file and stores the blockchain. Please copy the terminal output to a txt file. Please submit the txt file along with the json file and your code. Write a small report that explains what you did for each function.

Details about the code:

Blocks: The fields for the blocks is as follows:

```
block={
                    'previous_hash': 00000000000000,
                    'index': len(self.chain),
                    'transactions': [],
                    'bits': 0x1EFFFFFF,
                    'nonce': 0,
                    'time': str(datetime.datetime.now()),
                }
```

Bits: It is 32-bit input to produce a 256-bit hash target. The formula to convert the bits to target to is given as

$$target = bits[5:0] \cdot 2^{8\cdot(bits[7:6]-3)}$$

In the code you are going to create genesis block using the genesis_block which creates the first block with a given difficulty. Rest of the cases you are going to produce empty blocks and set ‘bits’, ‘time’, ‘nonce’, ‘index’ and ‘previous_hash’ fields.

You need to fill the following 4 functions:

Mine: You need to get the bits from the block and compute the target. By changing the nonce, you need to find a hash that is smaller than the target.

Get_Target_From_Bits: Take the bits from the block and compute the 256-bit target.

$$target = bits[5:0] \cdot 2^{8\cdot(bits[7:6]-3)}$$

Get_Bits_From_Target: Compute the bits (32-bit representation) using 256-bit target as an input. You need to inverse the operation of Get_Target_From_Bits.

Change_Target: Take the previous 32-bit bits, time of first block, time of last block and targeted time interval. If the blocks are checked every 32 blocks, the first and last block time belongs to the 32 blocks that are created for that round.