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

# Assignment 3: Understanding Attacks and Defenses

Due at 11:59pm Thursday, January 31, 2019

## Introduction

In this assignment, you will design and implement an attack on the GoodCoin that uses a bad node (or nodes) to "rewrite history". In particular, your attack will create a new version of the blockchain that erases a transaction of interest. This attack is very similar to the selfish mining attack discussed in class. In addition to implementing this attack on the GoodCoin, you will also change the GoodCoin consensus algorithm so such an attack is no longer possible and will answer a few questions about attacks in general.

To complete this assignment, you will download an updated version of the GoodCoin from Canvas. This new version includes the notion of a playbook and simulations, which will allow you to write out the actions of a GoodCoin network step-by-step. Though this simulation is deterministic and therefore a bit unrealistic, it will allow you to better understand the interactions between nodes in the network and will allow you to create a repeatable simulation of the selfish mining attak discussed above.

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

## 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 three files for this assignment. The first will be the playbook you wrote to run the selfish mining attack. This will be named `<YOUR CNETID>.playbook`. The second file will be an updated version of blockchain.py that implements bad nodes and updates the consensus protocol to defend against this version of the selfish mining attack. This file should be renamed `<YOUR CNETID>-blockchain.py`. The third file, entitled `<YOUR CNETID>-assignment2.pdf/txt`, will contain responses in English to the questions asked during the course of this assignment. All 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. Below, the new components of the GoodCoin are explained. Note that the bulk of this assignment will be command line-based rather than web-based as before, since controlling the actions of an entire network of GoodCoin nodes is difficult to do with web-based actions alone. After all the new files are presented, the desired behavior of bad nodes in the GoodCoin is discussed.

### New Files

- `network.py`: This file defines the notion of a Node and a Network for the GoodCoin. You can use this function to define a topology for the network, to start all the nodes in the network running, and to cause particular nodes in the network to perform different actions. These actions are defined in `playbook.py`.

- `playbook.py`: This file defines the kinds of actions nodes in a GoodCoin network can take and also introduces a way of defining a series of actions for the network. Currently, nodes in the GoodCoin network can do three things: mine a block, conduct a new transaction with another node, and flip from good to bad and vice versa. A playbook represents a series of actions defined for a particular network. It is possible to play through this playbook and observe the behavior of the network as it goes. A playbook can be randomly generated (see the `random_playbook` function) or can be defined action-by-action by the user (see `build_playbook`).

- `examples.py`: **This file will be very helpful in getting started on this assignment**. This file contains an example of how to build a playbook and a network. This is meant to show you ways you could interact with the playbook on the command line (i.e. you could run each line in this file in a command line Python interpreter such as IPython and achieve the same result as running the entire file at once). We have provided this for you as a way to get you started with command line interactions with the playbook. We think it will be easier to build your attack if you have the ability to quickly test things on the command line rather than editing a file and re-running the whole file.

- `simulate.py`: **This file is where you will run your attack**. This file contains some example simulations for the GoodCoin network and also is where the function to run your selfish mining attack is defined. You will have to first modify the playbook provided to the `main` function of `simulate.py` in order to have the attack in place. But then, by running `python3 simulate.py`, you can run the attack.

### Bad Nodes

Bad nodes in the GoodCoin behave as selfish miners. All nodes start out as good, but can be flipped through an action in the playbook (see the `playbook.py` file). Once flipped to "bad", a GoodCoin node will display the following properties.

- A bad node will no longer participate in the consensus protocol with good nodes. This means that any chain updates propagated through mining by good nodes will be ignored by the bad node (and the bad node will maintain its own version of the utxo pool and current transactions).

- A bad node will not notify good nodes of any new blocks it mines. A bad node will tell other bad nodes about its new version of the blockchain, but **only if those bad nodes are its peers** in the GoodCoin network.

Once a bad node flips from being bad to being good again, it will maintain its own isolated verison (though not completely isolated, because it may share its blockchain with any bad nodes

that are its peers) of the blockchain until another block is mined. Then, when a new block is mined, this formerly bad node will participate in the consensus protocol. Recall that the consensus protocol requires that all nodes accept the longest chain with the longest common prefix (where "common" is defined as two nodes seeing this particular prefix). This means that if two bad nodes present the same version of the blockchain with a longer chain and a longer common prefix than any version propogated by good nodes, that version will be accepted by all nodes in the network as the definitive version. You will use this fact to generate your selfish mining attack.

## Tasks

You will complete three tasks in this assignment. These are described below.

### Implement Bad Nodes (20 points)

In this part of the assignment, you will implement how "bad nodes" will behave in the GoodCoin blockchain. The desired behavior is described above. Your job is to modify `blockchain.py` to implement this behavior. It should be sufficient to modify the `resolve_conflicts` function to do so. **To repeat: you only need to modify the `resolve_conflicts` function in order to institute bad node behavior as defined above.** Do not modify other parts of the code. In particular, you may find the ability to query for a node's "goodness status" through a call to `requests.get(f'http://{node}/good')` or `flip_node` in either `blockchain.py` or `server.py` helpful.

Note that bad nodes can only communicate with each other if they are peers in the GoodCoin network, so you do not need to implement extra communication functions to allow non-neighbor bad nodes to communicate. For the purposes of your attack, you can assume you are able to choose which node(s) are bad and that you can choose nodes which are neighbors (should you need more than one node for your attack).

### Simulate Attack (30 points)

In this part of the assignment, you will simulate a selfish mining attack that erases a transaction of interest from the GoodCoin blockchain.

As part of this assignment, you have provided with a topology and a basic playbook that sets the stage for your attack. Both these things can be found in the `main` function of `simulate.py`. In the base attack playbook, you will find actions that instantiate a chain and conduct a transaction of interest.

Your job is to undo that transaction using the selfish mining attack. You will do this by adding actions to this playbook that will result in all nodes having a version of the playbook that reflects a GoodCoin chain in which this transaction did not take place. You will eventually submit an updated version of the playbook that builds on the actions defined there to perform this attack. As described above, you will rename this as `<YOUR CNETID>.playbook`. Please use the `Playbook.save_playbook()` function to save the playbook as a Python pickle file to make life easier for your TAs.

## Consensus Update (30 points)

In the previous part of the assignment, you created an attack that takes advantage of weaknesses in the GoodCoin consensus algorithm to get good nodes to accept a "malicious" verison of the blockchain.

In your writeup, describe what elements of the current GoodCoin consensus algorithm make this attack possible. Further describe what modifications could be made to the consensus algorithm to defend against this attack. Finally, implement the changes you suggest in the GoodCoin consensus algorithm. You will make those changes in the `consensus` function of `blockchain.py`. As described above, submit this file as `<YOUR CNETID>-blockchain.py`.

## Reflections (20 points)

Now, you will answer some reflection questions about your work in this assignment. Please provide a 3-6 sentence response to each question.

1. Could the attack you implemented in part 1 of this assignment happen in real life, specifically in the Bitcoin blockchain? Why or why not?

2. Describe a current remaining issue you see with the GoodCoin consensus protocol, even after your update in part 2 of this assignment. Describe an attack that could take advantage of this vulnerability.

## Helpful Hints

- If you try to start up a network and get errors telling you that a socket is already in use, you may have forgotten to shut down servers previously running on that socket. You can shut these down by killing the processes running them. Run `ps -fA | grep python` on your command line. Note the process ids for the processes of interest and run `kill x` (where x is replaced with the process id) for each process you want to kill.

- More helpful hints will be posted to Piazza as common errors are discovered.