CS 251: Bitcoin and Crypto Currencies Fall 2015

Project 3

due: November 9, 11:59 PM via email to cs251.stanford.fall.2015@gmail.com

Introduction

At this point in the course, you should be fluent with the structure of Bitcoin transactions. This assignment will draw on your knowledge of the blockchain structure and de-anonymization techniques. You will be given a truncated data set of Bitcoin transactions starting from the genesis block and ending at height 100,001. The block reward was 5,000,000,000 satoshis during this period.

While this is almost entirely *real* Bitcoin data and the code you develop for the assignment would equally work over the entire blockchain, the data is slightly simplified and we have removed or modified some transactions. For this reason, you'll need to work with this dataset to get the correct answers—using an externally parsed version of the blockchain will lead you to incorrect results.

Getting started

1. Download the blockchain data set from the course website. The data set contain three CSV files: inputs, outputs and transactions. The schema is explained below.
2. You can analyze the data using any programming languages or data-management tools that you desire. For example, you may import the data into a SQL database or other data management system if you are most comfortable doing that, or you may write your own parsing scripts using the programming language of your choice. We will not be grading your solution on query efficiency, so don't worry if you use an interpreted language.
3. You should submit written answers to the questions below (like for a HW assignment) but you must also submit whatever code you develop.

Exercises

1. We have inserted a variety of invalid transactions in this dataset. List the transaction ID for *at least* 6 invalid transactions, along with the reason why they are invalid.
2. After removing the invalid transactions, how many UTXOs exist as of of block 100,001 (the last block of the data set)? Which UTXO has the highest associated value?
3. Cluster addresses using two common idioms of use: *joint control* (addresses used as inputs to a common transaction are controlled by the same entity) and *serial control* (the output address of a transaction with only a single output is usually controlled by the same entity owning the input addresses). Assume there is no mixing or other obfuscation in place.
    a. As of block 100,001 (the last block of the data set), find the entity controlling the most total (unspent) bitcoins. What is its lowest address (numerically) and what is the total value of all the bitcoins it controls?
    b. Give the ID of the transaction sending the greatest number of bitcoins **to** this entity.

c. Is this clustering method accurate? List at least one potential source of false positives (clustering addresses which aren't actually owned by the same entity) and one source of false negatives (failing to cluster addresses which actually are owned by the same entity) in this method. What strategies could you use to make your clustering more accurate? *There is no "correct" answer for the last part of this question. Be creative and don't be afraid to propose novel ideas). You can see an example of clustering in action on https://www.blockseer.com/*

Data schema

There are three "tables" provided in three separate CSV files. The Transactions table simply contains a unique id for each transaction and the block id that transaction appeared in. Each transaction has one or more input(s) and output(s), each given a unique id, linked to a transaction and given an address id representing the address used to authorize that input/output. Each input also refers to exactly one output_id which it is "spending".

Note that with real Bitcoin data, the ids would of course be 256-bit hashes. We replaced them with numeric ids here to keep the dataset small.

Transactions
- id
- block_id *(database id, not hash)*

Outputs
- id
- tx_id
- address_id *(address sent to, -1 if non-standard scripts used)*
- value *(in satoshis)*

Inputs
- id
- tx_id
- address_id *(address spent from, 0 if coinbase tx, -1 if non-standard scripts used)*
- output_id *(previous output being referenced, -1 if coinbase tx)*