CS 251: Bitcoin and Cryptocurrencies Fall 2016

Project 3

due: 2016-11-14 23:59 via email to cs251ta@cs.stanford.edu

Introduction

By now 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 (50 BTC) during this period.

While this is almost entirely *real* Bitcoin data and the code you develop for the assignment could be adapted to work on 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 contains 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. Submit your written answers and whatever code you develop by emailing a .tar file to the address at the top of this page. Please type your answers in a plain text README file, and include instructions for running your code at the top of the README.

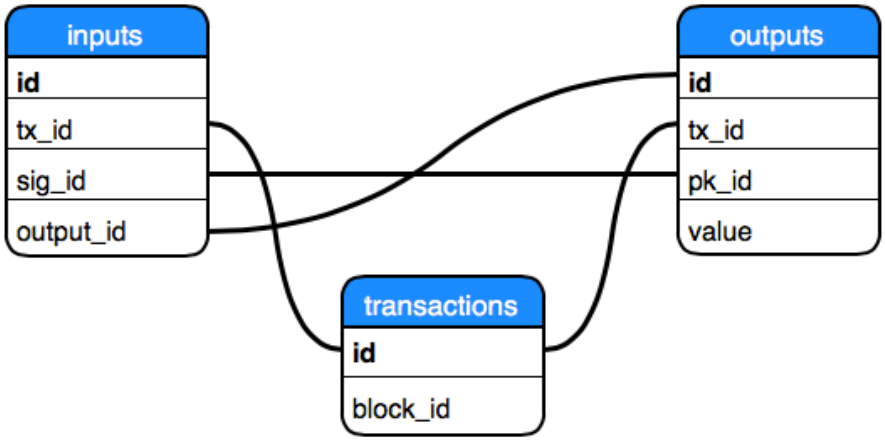
Exercises

1. Twelve of the blocks in this dataset are invalid for various reasons. List the block IDs for *at least nine* invalid blocks, along with the reason why they are invalid.
2. After removing the invalid blocks, how many UTXOs exist as of 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 input and 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 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.*

Data schema

Three "tables" are provided in separate CSV files. The Transactions table 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`. Each output has a `pk_id` denoting the public key used in the scriptPubKey and each input has a `sig_id` denoting the public key used in the scriptSig. 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. To keep the dataset small, they have been replaced with numeric ids.



transactions	
id	unique id
block_id	block containing this transaction
inputs	
id	unique id
tx_id	transaction this output is part of
sig_id	scriptSig public key id, 0 if coinbase tx, -1 if non-standard scripts used
output_id	previous output being referenced, -1 if coinbase tx
outputs	
id	unique id
tx_id	transaction this output is part of
pk_id	scriptPubKey public key id, -1 if non-standard scripts used
value	output value in satoshis