CSCI-GA.3033-019: Cryptocurrencies and decentralized ledgers Fall 2017

Project 3

due: 2017-11-02 22:00 via classes.nyu.edu

Introduction

By now you should be familiar with the structure of the Bitcoin transaction graph. 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 evaluating your solution on query efficiency, so don't worry if you use an interpreted language.
3. You should submit whatever code you develop as well as a plain text README file with your written answers to the questions below as well as instructions for running your code.

Idioms of use

In this assignment you will try to determine which addresses are owned by the same real-world entity by considering common two very simple idioms of use:

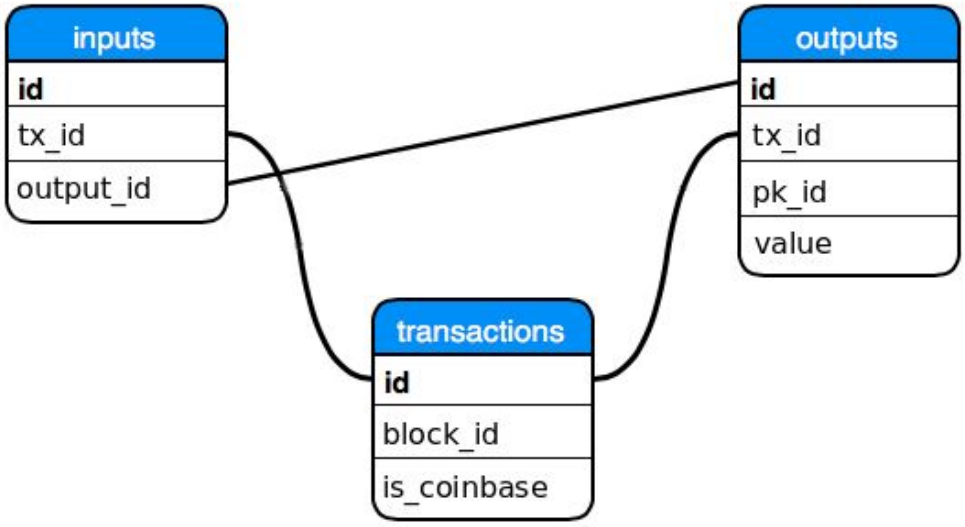
- *Joint control*: when two addresses provide common inputs to a single transaction, they are likely controlled by the same entity.
- *Serial control*: whenever a transaction has a single output, the output address is usually controlled by the same entity that owns the input address. The reasoning here is that it is rare for one entity to own exactly the right amount of bitcoin in a single UTXO that it wants to pay another entity, so typically a *change address* is used to return the leftovers to the original owner. It can be difficult to identify change addresses reliably, but if there is no change address typically the money is simply being cycled from one key to another by the same owner (or distinct UTXOs are being consolidated into a single larger UTXO).

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 completely removing the invalid blocks (remember that a single invalid transaction invalidates an entire block), how many UTXOs exist as of the last block of the data set? Which UTXO has the highest associated value?
3. Cluster addresses into entities that appear linked through either the *joint control* idiom or the *serial control* idiom.
    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 in which a different entity sends the greatest number of bitcoins **to** this entity.
    c. As mentioned, these clustering methods are just heuristics. 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 single "correct" answer for this part.*

Data schema

Three "tables" are provided in separate CSV files. The Transactions table contains a unique `id` for each transaction, the `block_id` that transaction appeared in, and whether the transaction `is_coinbase`. 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	chronological sequence ID of the block containing this transaction
is_coinbase	1 if this is a coinbase transaction, 0 otherwise
inputs	
id	unique id
tx_id	transaction this output is part of
output_id	previous output being referenced
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