COMP0143 Project

Due: 24th April 2019 by 4pm

Instructions

Answer all questions, writing your code in Python. When you are done, put all your code in a single `.py` file and put all your answers, along with an indicator of any partner you worked with, in a single PDF file. Use your candidate number and the module code as the name of the files; e.g., name your code file `KTJA9_COMP0143.py` and name your answers `KTJA9_COMP0143.pdf`.

You are allowed to discuss the questions at a high level with anyone, but you may not discuss specific answers with anyone other than your (optional) partner. You may hand in the same code as your partner, but your answers must be written separately, and both of you must hand something in. If your answer for a question is identical (modulo cosmetic tweaks) to an answer we find on the Internet or to the answer of another student, we will give no marks.

Data description

In this project, you will be given a truncated version of the Bitcoin blockchain, starting from the genesis block and ending at block height 100,017. This is real Bitcoin data, but the transaction data is simplified and some transactions have been removed or modified. Thus, it will not work to use an external version of the Bitcoin blockchain (e.g., one parsed from an online block explorer), and the code you write for the project would need to be adapted to work on the real Bitcoin blockchain.

The data for this project can be found on Moodle and consists of four CSV files, containing different types of identifiers. The on-chain identifiers would all be 256-bit hashes in the real Bitcoin data, but for the sake of this project they are numeric.

The `transactions.csv` file contains two columns:

1. `id`, which is a unique identifier for a transaction.
2. `block_id`, which is the block in which the transaction appeared.

The `inputs.csv` file contains four columns:

1. `id`, which is the identifier of an input to the transaction.
2. `tx_id`, which is the identifier of the transaction.
3. `sig_id`, which is the identifier of the public key associated with this input (i.e., the public key used in the scriptSig). This value is 0 if `tx_id` is a coin generation transaction and -1 if the transaction is using some non-standard script that does not involve a public key.
4. `output_id`, which is the identifier of the UTXO that this input is "spending." This value is -1 if `tx_id` is a coin generation transaction.

The `outputs.csv` file contains four columns:

1. `id`, which is the identifier of an output in the transaction.
2. `tx_id`, which is the identifier of the transaction.
3. `pk_id`, which is the identifier of the public key associated with this output (i.e., the public key used in the scriptPubKey). This value is -10 if the transaction is using some non-standard script that does not involve a public key.
4. `value`, which is the amount being sent to this output (in satoshis).

The `tags.csv` file contains three columns:

1. `type`, which is the type of the service (one of Exchange, DarkMarket, Vendor, or Wallet).
2. `service`, which is the name of the (fictional) service.
3. `pk_id`, which is the identifier of the public key that belongs to this service.

1 Basic statistics [20 marks]

1.1 Transactions [3 marks]

How many transactions were there in total? Of these, how many transactions had one input and two outputs? How many transactions had one input and one output?

1.2 UTXOs [5 marks]

How many UTXOs exist, as of the last block of the dataset? Which UTXO has the highest associated value?

1.3 Public keys [5 marks]

How many distinct public keys were used across all blocks in the dataset? Which public key received the highest number of bitcoins, and how many bitcoins has it received? Which public key acted as an output the most number of times, and how many times did it act as output?

1.4 Invalid transactions [7 marks]

Several of the transactions in the dataset are invalid. List the `tx_id` for *at least* five of these transactions, along with the reason why they are invalid.

2 Clustering [40 marks]

We covered one common heuristic for clustering different public keys: the "multi-input" heuristic, which says that public keys used as input to the same transaction are controlled by the same entity. Assuming there is no mixing or other obfuscation in place, write code to cluster the public keys in the dataset according to this heuristic. Use commenting to mark this code clearly in your file.

2.1 Specific cluster [6 marks]

How big is the cluster that contains public key (`pk_id`) 41442? Identify the cluster according to its keys with the lowest and highest numeric values.

2.2 Biggest cluster [12 marks]

Which cluster has the largest number of keys, and how many keys does it contain? Identify it according to its keys with the lowest and highest numeric values.

2.3 Richest cluster [12 marks]

As of the last block in the dataset, which cluster controls the most unspent bitcoins, and how many bitcoins does it control? Again, identify it according to its keys with the lowest and highest numeric values.

Which transaction is responsible for sending the largest number of bitcoins to this entity (i.e., to one or more of the keys in this cluster)?

2.4 Heuristics [10 marks]

Is this clustering heuristic accurate? Identify at least one potential source of false positives (keys that are clustered together but are not actually owned by the same entity) and one source of false negatives (keys that were not clustered together but are owned by the same entity). What strategy could you use to make your clustering heuristic more accurate?

3 Tagging and tracking [40 marks]

While clustering is already a useful way to gain some insights into the Bitcoin ecosystem, tagging provides a way to additionally learn the real identity of entities and track money as it is transferred between them. Use the tags available in `tags.csv` to associate both individual keys and larger clusters with (fictional) real entities. Use commenting to mark this code clearly in your file.

3.1 Richest service [15 marks]

Which tagged entity controls the most unspent bitcoins, and how many bitcoins does it control? Be careful to consider entities that may control multiple tagged clusters.

3.2 Interactions [15 marks]

How many transactions sent bitcoins directly from a (fictional) exchange to a (fictional) dark market? How many bitcoins in total were sent across these transactions?

3.3 Tracking techniques [10 marks]

What are the risks associated with accepting as conclusive the results of this analysis, when a transaction seems to demonstrate a clear connection between two entities, such as a user and an exchange? What about in the "two-hop" case (or, more generally, a multi-hop case) where the user sends to one key and then that key sends to an exchange? What strategy could you use to gain further confidence in the results of this analysis?