Bitcoin Network Data Tools, v0.5

Ivan Brugere
Laboratory for Computational Population Biology, University of Illinois at Chicago
Jan 05 2013

---

0. Acknowledgement:

We use tools developed by Martin Harrigan (github.com/harrigan/bitcointools), which extends Gavin Andresen's bitcointools (github.com/gavinandresen/bitcointools)

We also port and extend code by Harrigan and form a 'user network' according to the strategy in "An Analysis of Anonymity in the Bitcoin System" (Reid and Harrigan, 2011; http://arxiv.org/abs/1107.4524).

1. Overview:

Bitcoin (bitcoin.org) is a digital, cryptographically secure currency. Transactions between public-key "addresses" maintained in a distributed, verified public ledger form a transaction network that can be studied by network scientists. This code processes binary-format Bitcoin .dat files generated by the Bitcoin client (bitcoin.org, tested on v0.5.3.1 or lower) into human-readable flat-file formats, retaining all available information. Furthermore, we provide a data model to facilitate storage and querying in a relational database.

2. Bitcoin transaction overview:

The bitcoin digital currency allows users to securely prove ownership of a portion of coins that cascade through the network as a chain of re-assigned ownership *transactions* over time.

A *transaction* on the bitcoin network is a many-to-many function, executed by a user who has ownership to (potentially many) outputs of previous transactions; the user takes this owned value and writes ownership to (potentially many) output nodes (users, represented by addresses in the network).

Figure 1 (see also: “example_transactions.png”) illustrates a chain of transactions in the network.

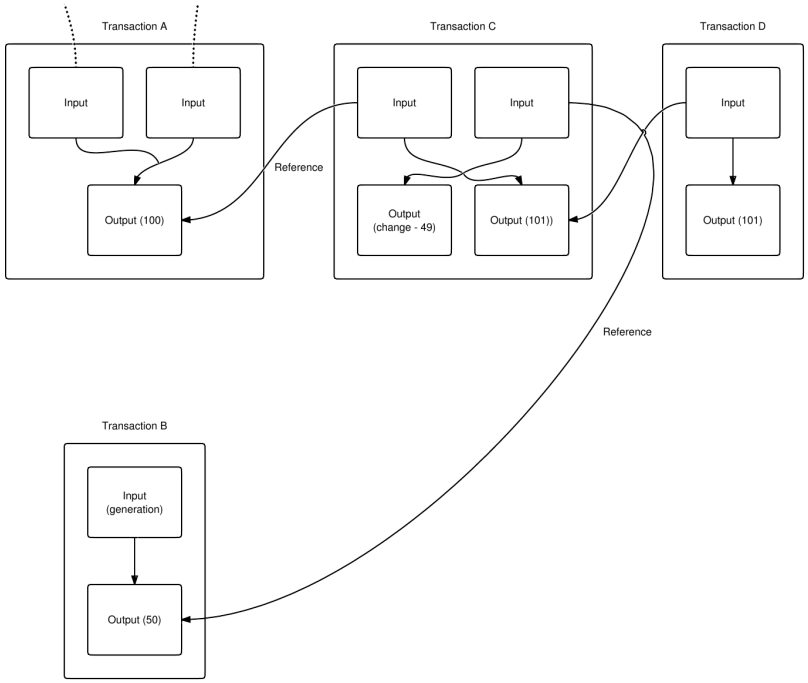

Figure 1: An example transaction chain. Transaction C is a transaction of two inputs and two outputs. A user owns the outputs from Transaction A and Transaction B, and signs this combined value (150) to two outputs. Because the network reclaims any unassigned value from the inputs, change is often assigned back to the user (source: en.bitcoin.it/wiki/File:Transaction.png).

3. File output description:

We generate the following files, with the included row specification:

pubkey_list.txt: [*public_key_string*]
transactionkey_list.txt: [*transaction_key_string*]

userkey_list.txt: [$public\_key_1$,$public\_key_2$,…,$public\_key_k$]
user_edge_inputs.txt: [$transaction\_key_{self}$,$transaction\_key_1$,…,$transaction\_key_k$]
user_edge_inputs_public_keys.txt: [$transaction\_key_{self}$,$public\_key_1$,$public\_key_k$]

user_edges.txt: [$transaction\_key_{self}$, $user\_key_{from}$, $user\_key_{to}$, *date*, *value*]

The first group of files, ("pubkey_list.txt", "transactionkey_list.txt") contains the lengthy text keys for public keys (i.e. "1EnHwdiKxvTE5AzcSnZqS52mMcHSLtCLwH") and transaction keys (i.e. "5dc77144dcf46a7f76e369d406481e857be9e95b21375935832a5bed4e23633b"). Each 'key' field is a line number (starting at 1) to index into the appropriate list file. Using this, information on a public key or specific transaction can be queried on sites such as blockchain.info.

The second group of files ("userkey_list.txt", "user_edge_inputs.txt", "user_edge_inputs_public_keys.txt") organizes the 'user' information, where a 'user' is a grouping of public keys inferred from public keys combined as inputs into a single transaction (meaning the user owns the private key to each address). This method is described in (Reid and Harrigan, 2011). Briefly, we create a graph where two public keys have an edge if they have been used as inputs in a single transaction. The connected components of this graph are 'users'. Each line in "userkey_list.txt" is one of these components (a grouping of public keys). The files "user_edge_inputs.txt" and "user_edge_inputs_public_keys.txt" record the transaction keys, and public keys used as input to "this" transaction (see Figure 1).

The third group of files ("user_edges.txt") is the primary network data file. The files "user_edges.txt", "user_edge_inputs.txt", and "user_edge_inputs_public_keys.txt" share a transaction key field which allows them to be joined on.

See the relational model diagram "bitcoin_relational.pdf" summarizing these files.

4. *Python code execution:*

The included Python scripts use only core packages and have been tested on Python 2.7, with Bitcoin dat files generated by the official client, version v0.5.3.1. The code executes on a log of 46.9M transactions (dated Jan 03 2013) with a memory footprint under 10GB, and completing in less than 4 hours, (memory is cheap, but still this could be improved by using non-core packages like NumPy).

The main shell script "process_bitcoin_network_runner.sh" can be configured with your paths and desired output filenames. The two primary paths needing set by the user are the file output path ("file_path"), and "lib_path" the path to Martin Harrigan's branch of bitcointools (https://github.com/harrigan/bitcointools), which handles the extraction of the binary files to raw text output (as of Jan 03 2013, about 7.5GB of transaction data).