Cryptocurrency transaction mapping

Isak Häggström
970825-6475
e-post: isaha512@student.liu.se

Mátyás Barócsai
980311-3159
epost: matba803@student.liu.se

Supervisor: Niklas Carlsson, niklas.carlsson@liu.se
Project Report for Information Security Course
Linköpings universitetet, Sweden

May 24, 2021

Abstract

Cryptocurrency has become a very popular in todays society offering a digital, decentralized, and anonymous currency. Due to its anonymity, bitcoin together with other cryptocurrencies has increasingly been used in criminal and fraudulent activities such as scams, blackmailing and black-market purchases. This report explores ways to visualize cryptocurrency transactions, and tries to answer the question on what information can be extracted from a cryptocurrency address. The result is Cryptocurrency Transacton Mapping Tool (CTMT). A tool that uses the Blockcypher API to gather information on transactions and vizualise these transactions in different graphs.

1 Introduction

A cryptocurrency such as Bitcoin is a Virtual or digital currency protected by cryptography. Most of the current cryptocurrencies are decentralized ,peer-to-peer (P2P), networks. These P2P networks manages a distributed ledger called blockchain, which is the technology that allows for the different cryptocurrencies to be transferred around the world without a need for a mediator or a central server and the possibility for anonymity.

A blockchain is essentially a digital ledger of transactions. This ledger is duplicated and distributed across the entire network of computer systems on the blockchain. Each part of the chain contains a number of transactions, and for every transaction that occurs on the specific blockchain, a record of that transaction is added to every participant's ledger.[1]

1.1 Purpose

The purpose of the project is to get a better understanding of how bitcoin and other cryptocurrencies can be used in various scamming operations.

1.2 Goal

The goal of this project in the course TDDD17 is to collect, filter and graph information about bitcoin transactions. And by doing this provide a tool to help visualize how the money moves from the victim to the scammers bitcoin wallets and further down.

1.3 Problem statement

The following problems will be discussed and analysed in this project:

- How do we create a visualization tool to follow bitcoin transactions?
- Which type of information is possible to extract from a bitcoin address?

2 Background

This section will discuss the methods used in the project, both theoretical and practical.

2.1 Past work

This project is built upon work done by students in this course (TDDD17) last year (2020). Their project goals was to gather information from Bitcoin addresses and categories these depending on what information the user is interested in. This project will use part of that information when choosing the addresses to be graphed.

2.2 Methods

This section will introduce the methods and tools that are going to be used to implement and analyse the cryptocurrency transaction mapping tool (CTMT).

2.2.1 Blockcypher API

The Blockcypher API is a free to use API for collecting information about and/or interacting with blockchains [2]. In this project the Blockcypher API will be used to collect the transaction history of bitcoin addresses which have been linked to potential criminal activities. The Blockcypher API supports multiple language SDK's but in this project the API will be used in Python only. The API itself will be installed and used as described in Blockcypher's official python library manual [3].

Blockcypher API does enforce certain constraints when using the free tier. The only constraint that will affect our implementation is a rate-limit, which ensures that only 200 API request can be made per hour by a free tier user such as ourselves. [2]

2.2.2 Python

In this project the Python programming language will be used, due to it being an easy and intuitive language excellent for smaller project such as this one. Furthermore Python has a large number of third-party modules which can be used to quickly implement code. [5]

2.2.3 Graphviz

Graphviz is an open-source python module that in this project is used to create graph objects specified in DOT Language, which is a graph description language. The tool was initated by AT&T Labs. [4]

2.2.4 JSON

JSON (JavaScript Object Notation) is language independent data format with the purpose of being lightweight and easy to read for both humans and machines. JSON is completely language independent even if it uses conventions from the C-family of languages, these languages includes C, C++, C#, Java, JavaScript, Perl, Python etc. These characteristics make JSON popular to use for communication between programming languages. [6].

3 Solution and Analysis

In this section the solution and result will be presented and analyzed.

3.1 Cryptocurrency Transaction Mapping Tool (CTMT)

To answer the problem statement and be able to present paper-ready figures a python tool was programmed. The tool's purpose is to make it easier to collect, filter and generate a diverse set of graphs each representing the transaction history of a given Bitcoin wallet. The tool is made up of three script modules each with its purpose and functionality. The modules are executed via the main program separately or in pipe-lined fashion as seen in figure 1.

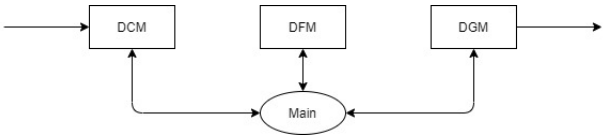

Figure 1: Structure of the python tool

3.1.1 Data Collection Module (DCM)

The Data Collection Module (DCM) is responsible for retrieving the transaction history for a specific address given by the user. It stores the information in a json format inside a specified file.

The Data Collection Module starts by asking the user for a specific amount of transactions that should be retrived. Then the Blockcypher API is used to request the information on the transactions. A loop is needed here because the API only allows to gather information on 50 transactions for each request. There is also a limit on how many request per hour can be done, 200. After this number is reached the program stops for an hour and then restarts at the same spot. When the number of transaction that the user is interested in is reached the loop stops and the information is written to a specified file. Figure 2 shows the two different API requests made depending on if we have reached the request limit or not.

```
if n_tx - count < TXN_LIMIT and not fRun:
    details = get_address_full(address=BTC_ID, before_bh=before,
    txn_limit = n_tx - count , inout_limit=SUB_TXN_LIMIT )
    total_txs.append(details)
    break
else:
    details = get_address_full(address=BTC_ID, before_bh=before,
    txn_limit=TXN_LIMIT, inout_limit=SUB_TXN_LIMIT )
```

Figure 2: A code snippet of the requests to the Blockcypher API.

3.1.2 Data Filtering Module (DFM)

The purpose of the Data Filtering Module (DFM) is to filter and limit the number of transactions which will be graphed. As the data collection module's only functionality is to collect and store the transaction history of a given Bitcoin wallet, it becomes clear that redundant or irrelevant information about transactions will be gathered by the module. Limiting or filtering the gathered information greatly reduces graphing times for the Data Graphing Module (DGM) and makes the resulting graphs not only more readable but better looking.

The filtering technique is chosen by the user when the module is executed by the main program. There are multiple filter types each resulting in a different graph. Currently there are three filter types programmed from which the user can choose their desired one.

The ”raw filter” filters only redundant transaction details but does not limit the number of transactions graphed. This filter type results in graphs that are unnecessarily large and complex but can give the viewer an understanding of the complexity of a bitcoin transaction.

The ”blockchain filter” filters the data such that the resulting graph shows only the blockchain transactions and not the individual bitcoin addresses and their transactions. The ”interval filter” filters the data into intervals, the result is a graph that will show the different intervals of BTC amount in which transaction have taken place . The filtering is applied to the collected data by reading the JSON file in which the DCM has stored the raw transaction history. The filtered data will then be written to another JSON file from which the DGM can graph the resulting graph.

3.1.3 Data Graphing Module (DGM)

The purpose of the Data Graphing Module is to create and open an intuitive, aesthetic and easily understood graph based on the data which the DFM has filtered and stored in a JSON file. The graphing itself is done with the help of Graphviz (see section 2.2.3 for more information).

The file that should be made into a graph is chosen by the user if the Data Graphing Module is the only module being run, or is automatically picked if the whole program is running. To draw using Graphviz each object in the input is read and added as nodes to the graph. The result is then saved and showed to the user.

Graphing times are dependant mostly on the amount of data which is processed. An address with longer transaction history will result in longer execution of the module. The type of filtering which is used will also have an affect on the execution times. The fastest graphing time is achieved with the interval filter as this is the filter type which has the least amount of nodes. The raw filter results in the longest graphing times taking up to 5 minutes to graph 100 transactions.

3.2 Graphs

The two tables below shows the number of transactions from a specified address per Bitcoin interval. The graphical representation of these tables can be seen in figure 3 (Appendix A).

Table 1		
Interval (BTC)	Nmbr of txs	Direction
0 - 0.0015	60	Input
0.0015 - 0.003	21	Input
0.003 - 0.0045	22	Input
0.0045 - 0.006	12	Input
0.006 - 0.0075	4	Input
0.0075 - 0.009	5	Input
0.009 - 0.0105	4	Input
0.0105 - 0.0120	8	Input
0.0120 - 0.0135	0	Input
0.0135 - 0.0150	2	Input
0.0150 - 0.0165	0	Input
0.0165 - 0.018	2	Input
0.018 - 0.0195	0	Input
0.0195 - 0.021	2	Input
total number of txs	142	

Table 2		
Interval (BTC)	Nmbr of txs	Direction
0 - 0.0015	11	Output
0.0015 - 0.003	8	Output
0.003 - 0.0045	7	Output
0.0045 - 0.006	8	Output
0.006 - 0.0075	1	Output
0.0075 - 0.009	4	Output
0.009 - 0.0105	2	Output
0.0105 - 0.0120	8	Output
0.0120 - 0.0135	0	Output
0.0135 - 0.0150	1	Output
0.0150 - 0.0165	0	Output
0.0165 - 0.018	1	Output
0.018 - 0.0195	0	Output
0.0195 - 0.021	2	Output
total number of txs	53	

4 Discussion

This section consists of an analysis and evaluation of the results from this project. This includes a discussion of the method of work, graphical representations and future work based upon our results.

Starting up the project and gather information on transactions using the Blockcypher API was not too difficult. One problem however that occured with Blockcypher API were the amount of requests needed to gather the full history of transactions. This due to the fact that each request could only gather information on 50 transactions and there was a limit of 200 requests per hour.

Another thing that proved to be difficult was the amount of data that every transaction consisted of. As well as Filtering through that data to pick the things that were needed to do the task. To solve this problem the Data Filtering Module was introduced, this gives the user options on what information should be filtered from the full transaction history. The different filters implemented are described in section 3.1.2.

4.1 Graphical results

In Appendix A the graphical representations are displayed. Figure 3, shows the graphical result for a specific address using the interval filtering. This filter gives a good overview on which bitcoin amounts are the most transferred for that specific address. In section 3.2 this graph is represented as tabular data, it is clearly seen in these tables that the majority of the transactions both in- and outgoing are in the lower intervals. One thing that the graphical representation of the interval filtering lack is the addresses listed in the graphs. A thing that could be done about this is listing the addresses for every interval on a separate paper. Putting the addresses directly in the graphs, just makes things cluttery.

In Appendix A, figure 4, a graphical example of the Blockchain filter can be seen. This filter is presented such that every transaction is a node and the amount sent to the specified address is written on the edges. This type of filter is good if the user is interested in seeing the specific amounts received and sent from a specified address for

a few transactions. What is lacking is the same thing as for the interval filter and this could be solved in a similar way as mentioned before.

Appendix A also includes a graphical example of no filter being applied, which can be seen in figure 5. What is clear is that to get a readable view on an ”address to address” level, the number of transactions can not reach a level higher than just a few transactions. The reason for this is simply that every Bitcoin transaction can consist of different amounts of addresses, in the transactions for this project the amount could reach 200. This means that the size of the graph grows really fast when the number of transactions gets higher.

4.2 Future work

It can be said with a clear conscience that this project can be improved on in multiple ways. During the project we encountered various problems and hurdles that led us to rethink our strategy of implementing the transaction mapping tool in the most efficient way.

Future work could be done on the Data Collection Module, 3.1.1, by solving the issue with the 200 request/hour limitation which the Blockcypher API enforces on its free-tier users. This issue strongly limits the amount of data which can be gathered and analysed, and solving it should therefore be of a higher priority for those who may want to analyze addresses with longer transaction history.

In this project much work and effort was put on creating a code base on which it was easy to implement more filter types. Future work could implement more filter types allowing more information to be gathered, perhaps shedding some further light on how criminal and fraudulent activities are conducted with the help of cryptocurrencies.

The modularity of the code base also allows for changing parts of the program. These new parts can be in other programming languages due to the fact that the json protocol is used to communicate between the modules.

Alternatively work could be done on the Data Graphing Module, 3.1.3. Creating intuitive and easily understood graphs proved to be more difficult than we in the beginning anticipated, as it became clear that the overabundance of data collected made it harder to create small graphs that still accurately reflected the data they were representing. Future work could perhaps find new and better ways of representing the huge amounts of data that is collected. Graphs that accurately depict the transaction history but is still understandable to the viewer.

5 Conclusions

In this section conclusions will be drawn from the project and discussed their impact and relevance to the subject.

5.1 How do we create a visualization tool to follow bitcoin transactions?

There is not a ”one” way of creating a visualization tool that follows bitcoin transactions. There are various ways of creating such a tool, each with its own benefits and weaknesses. In this project we have created a tool with the help of third-party libraries and API’s. Although not necessary for creating a visualization tool, third-party libraries and API’s greatly reduce the work amount and completion time for a tool. Furthermore a dynamic filter system is perhaps not vital for the task but is strongly recommended in order to create more diverse visualizations. With a dynamic filtering system which is not ”hard-coded” the same data can be presented in multiple ways, allowing for more analysis which in turn leads to better understanding.

5.2 Which type of information is possible to extract from a bitcoin address?

The type and amount of information which is possible to extract from a bitcoin address is dependant on how much work and effort one wants to put on analyzing the address. Trivial information such as address name, amount of BTC in account and number of transactions made is obviously not very hard to extract with the help of a third-party API like Blockcypher API, which was used in this

project. The transaction history is possibly the most interesting and valuable information which can be gathered from a bitcoin address, as it includes all the transaction details, outgoing as well as incoming money to the account. By analyzing and categorising the transaction history it is possible to extract even more data from the bitcoin address. It is possible to see in which BTC range the majority of transactions sent money or when in time most transactions were made. It is possible to see if there are any reoccuring transactions made to specific addresses and if the BTC amount is reoccuring also.

References

[1] Sheldon Mark D Binance Academy Jenkins J Gregory. *Bitcoin and Blockchain: Audit Implications of the Killer Bs*. URL: https://search.ebscohost.com/login.aspx?direct=true&db=buh&AN=149147359&site=eds-live&scope=site. (visited on 06/04/2020).

[2] Blockcypher.com. *BlockCypher's API documentation*. URL: https://www.blockcypher.com/dev/bitcoin/#introduction (visited on 05/05/2020).

[3] Blockcypher.com. *Official Python library for Blockcypher web services*. URL: https://github.com/blockcypher/blockcypher-python (visited on 05/05/2020).

[4] https://graphviz.org/. *Welcome to Graphviz*. URL: https://graphviz.org/ (visited on 03/05/2020).

[5] https://www.python.org/. *Python 3.9.5 documentation*. URL: https://docs.python.org/3/ (visited on 02/05/2020).

[6] json.org. *ECMA-404 The JSON Data Interchange Standard*. URL: https://www.json.org/json-en.html (visited on 06/05/2020).

Appendix A

This appendix is a list of different graphical representation of a bitcoin transaction.

Figure 3: Interval filtering on a specified address.

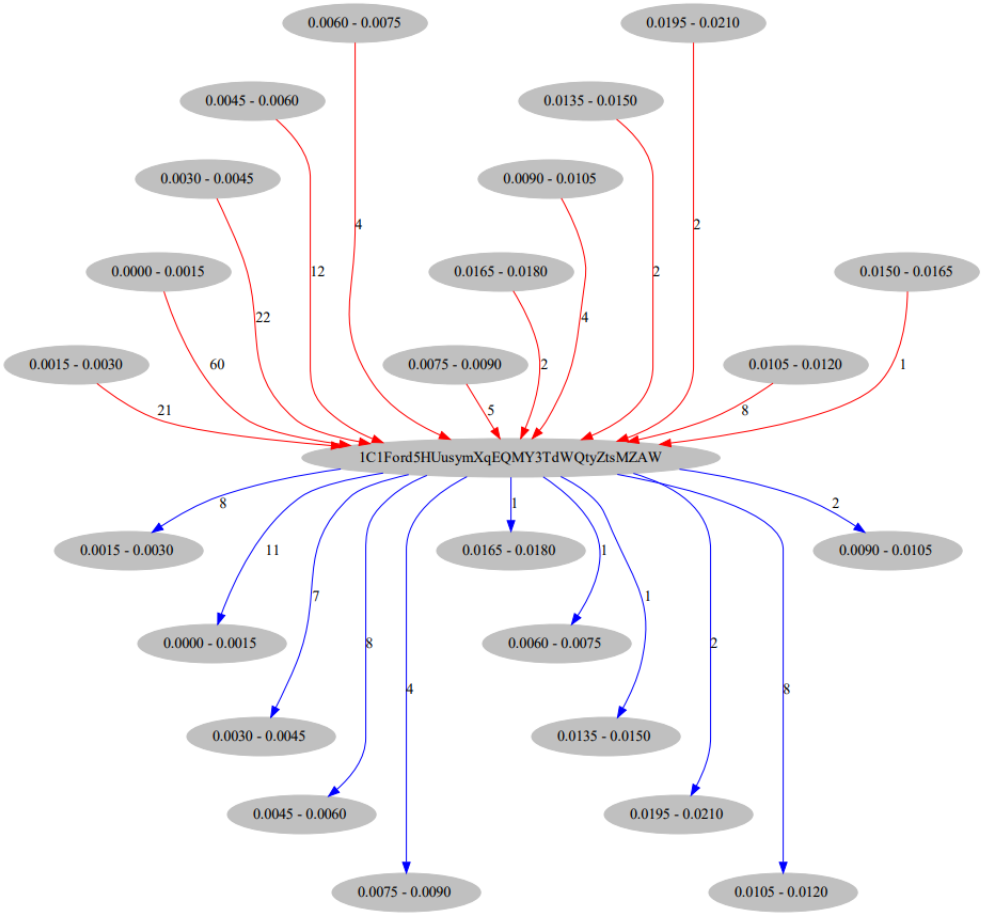

Figure 4: Blockchain filtering with 50 blockchains.

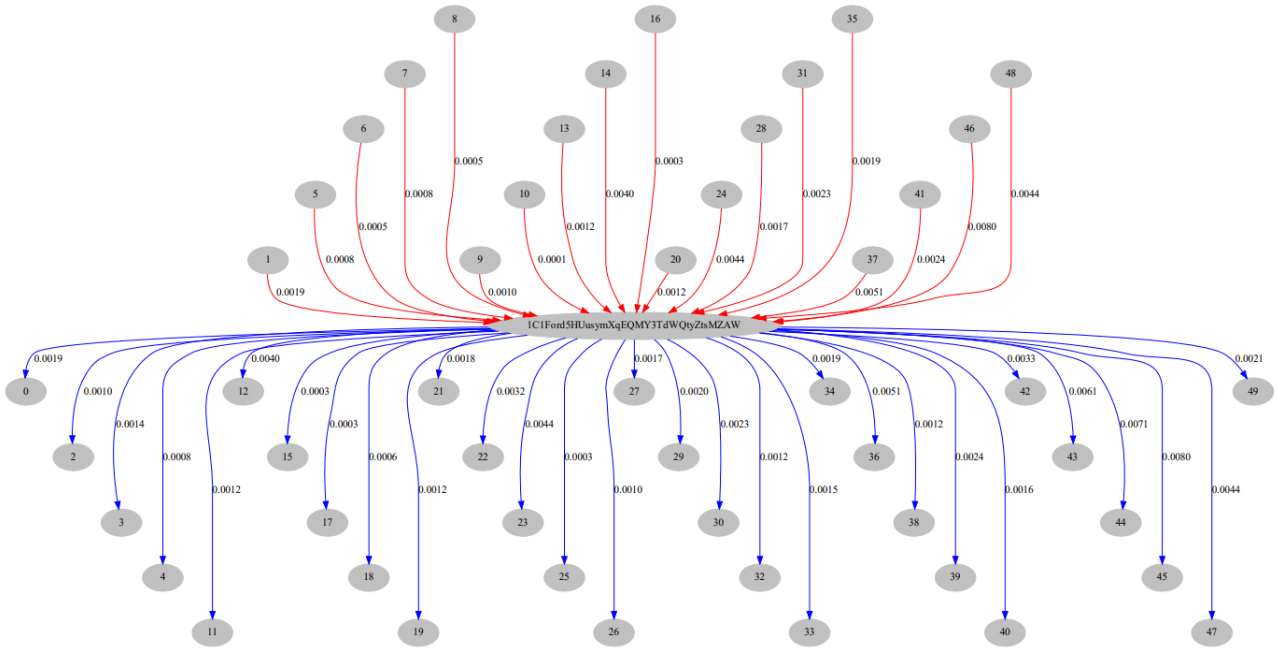

Figure 5: Sample from graph without filtering