Follow the Money

Louise Almér, Nina Argillander
Email: {loual201, ninar253}@student.liu.se
Supervisor: Niklas Carlsson, {niklas.carlsson@liu.se}
Project Report for Information Security Course
24th May 2021
Linköping University, Sweden

***Abstract*—Illegal activities have been linked more and more to the usage of cryptocurrencies since the users are anonymous and therefore can not be tracked by authorities. Since anyone can use cryptocurrencies, it is also practical for criminals to use. This report investigates if it is possible to visualize suspicious activities based on the transactions made from suspicious Bitcoin addresses. A tool was implemented, which retrieves data from a given input Bitcoin address, filters the data and displays a transaction flow from that address. From the result, the transaction flow can be visualized but the results varies depending on the filtering type. When displaying the top *k* transactions for each step in the transaction history and placing remaining transactions as a single *other* transaction, the best visualization were obtained. Therefore, it can be concluded that it is possible to visualize the transaction flow from one Bitcoin address.**

1. Introduction

Bitcoin is an electronic cryptocurrency. It has no central authorization that controls and can manipulate the currency, like a government or other large authority. Bitcoin users are only known by their Bitcoin address and it is nearly impossible to find out who is the owner of a certain Bitcoin address. Since all users are anonymous, Bitcoin can be used for illegal activities without repercussions.

This project investigates the possibility to visualize the flow of money from suspicious addresses. The project have some limitations, only one suspicious address is used as an input address to the tool i.e., it will not be possible to visualize the flow from multiple input addresses at once. The input addresses used for testing the tool were received from the supervisor. From earlier projects these addresses had been located and deemed suspicious and therefore interesting to visualize the money flow from.

2. Theory

In this section the background of Bitcoin is presented. The theory for the data retrieval and visualization is thereafter explained.

2.1. Bitcoin

Bitcoin is an electronic cryptocurrency created by Satoshi Nakamoto, an unknown person or group, in 2008. It is a decentralized currency, meaning that there is no central authority that controls the currency, e.g. a government or other large authority. Instead Bitcoin uses *ledgers* and the *block chain* technology. Ledgers are, traditionally, a book where transactions to and from an account are recorded. Bitcoin uses a block chain as a virtual ledger, which records all transactions. The transactions are put into blocks which are hashed and the blocks are connected through the hashes. Each block contains the hash of the previous block, creating a block chain. All users keeps their own copy of the block chain and the copy is updated when new blocks are added into the chain [1].

Since each block is hashed, a lot of computations are needed to make the transactions valid and inserted into the block chain. These computations are done by *miners* who listens for transactions, creates blocks for them and computes the hashes. The hashes need to fulfill certain requirements, e.g. have a certain amount of zeros in the beginning of the hash, which makes it harder to find a correct hash. When the hash has been computed the block is broadcast to all users and then added to the block chain. The block chain ensures that all users are agreeing that the transactions in the different blocks are valid, therefore no central authority is needed. The miners get a profit for each block they manage to find the first correct hash for, therefore the miners competes on computing a correct hash [2].

There is a possibility that an attacker creates fake blocks and broadcast the computed hashes. This would then be added to

the block chain the users keep. In the block chain technology there exists safety mechanisms that will prevent fake blocks being added to the block chain. The users receiving fake blocks will be listening to the miners creating valid blocks as well, i.e. receiving blocks from two sources. The block chain of the users creates two forks, one from the miners and one from the attacker. The block chain that is valid is always the longest, so for the attacker to succeed they have to compute blocks faster than the miners. This is almost impossible to achieve since there are many thousands of miners and perhaps only one attacker, so the chance of the attacker computing hashes faster is very limited. If the attacker would be faster than the miners it would be more profitable for the attacker to add the valid blocks instead of the fake blocks since the fastest miner gets a reward [2].

All transactions are made public by the block chain, which means that anyone can see any transaction. The transactions are connected to the input and output addresses, therefore can the transactions of a specific address be located. Even if the transactions of an address are examined the owner will remain anonymous, therefore it is common for criminals to use bitcoin for illegal activities [3].

2.2. API:s

Since all transactions are made public by the block chain, it is possible to access the transactions even if one is not a Bitcoin user. There are several different strategies for retrieving data from the Bitcoin block chain, either by writing an API or using preexisting API. In this project preexisting APIs were used so that the focus of the project could be the visualization and not data retrieval. Two API:s for data retrieval were tested; Blockcypher Data API and Blockchain API. The Blockchain API was not as documented as the Blockcypher API, therefore the Blockcypher API was used for data retrieval.

Blockcypher has multiple API:s for different retrieval purposes e.g., transaction API, wallet API and Blockchain API. Only the transaction API was used since the only information needed was about transactions. The transaction API have rate limits for both daily and hourly requests, these were not specified but from some experimenting seems to be 120 requests/hour and 2000 requests/day. This limitation was taken in consideration in the implementation of the tool, presented in section 3.1.1.

2.3. Visualization

The visualization had to visualize Bitcoin addresses and how these are connected through transactions. Different possible diagrams for the visualization were considered and after some research it was concluded that a Sankey diagram could be a good diagram to use for this type visualization. With a Sankey diagram the addresses can be represented by nodes and the transactions can be represented by the connections between the nodes. Furthermore, the width and color of the nodes and connections could represent different aspects, e.g. the amount of money or the number of transactions between the addresses, to give more dimensions in the diagram.

3. Implementation method

In this section the programming language and framework used in the project is presented. The implementation of the tool is explained as well.

3.1. The Tool

Below are the three different parts of the system presented and how they were implemented.

3.1.1. Data retrieval. Python was used to access the BlockCypher API. With the function `get_address_full` of the Transaction API all of the information about the input address was received, including a list of all transactions made to and from the address. For the visualization purpose only the output transactions were interesting as these are the transactions made from the current address. The function has a limit of how many transactions that can be received in one request, 50 transactions at the most, and if the address has more transactions the flag `has_more` is true. The tool will then call `get_address_full` again, but with the flag `after` set to the number of transactions that has already been received. This makes it possible to collect all of the transactions made from an address.

In the tool the transaction values for all transactions made to the same address are added together and the number of transactions are counted. Therefore, only one connection between a source and a target address is saved. The source address, all of the target addresses with their total transaction value and the number of transactions made are saved into a Python dictionary. For each target address in the dictionary the `get_address_full` function is called and the process is repeated.

When all data has been retrieved, or the daily limit is reached, the data is saved to a CSV file as well as a pickle file. A pickle converts a Python object to a character stream that can be unpickled at a later time. By saving it as a pickle it is possible to filter the pickled data in several ways, without using the API to collect the data every time.

Since the API has rate limits, a counter for the hourly respective the daily calls to `get_address_full` counts

how many requests are made. If the hourly limit is reached the program will pause for an hour and if the daily limit is reached the program will save the data retrieved and exit.

3.1.2. Filtering the Data. The filtering process was implemented in Python. The transaction data that was retrieved contained plenty of transactions that were too small to be deemed important. To visualize all of the transactions in the diagram it would be clustered with unimportant transactions, therefore the data had to be filtered before visualization. Three different types of filtering were implemented; filter by the number of transactions between addresses, filter by the money exchange between addresses and filtering based on the top k transactions.

Filtering based on number of transactions was implemented since it could be interesting to see how many transactions were made between two addresses. If an address sends a small amount of money frequently to an address it could indicate suspicious behaviour. Another suspicious behaviour could be to send a large amount of money in a few transactions, therefore was the filtering by money also implemented. Another type of filtering implemented was filtering by number of top transactions. For every step in the transaction flow the k largest transactions were saved whilst the rest of the transactions were bundled together in one large transaction called $other$. By limiting the number of transactions on each step to $k + 1$, a cluttered diagram could be avoided and the transaction flow from the input address were more understandable.

3.1.3. Visualize the Data. JavaScript with React and the framework D3 were used to visualize the data. In the CVS file, with the collected data, the source, target, transaction value, number of transactions and the step number were stored. When using Sankey diagrams every address becomes a node and the nodes are connected by the transactions. The height of the node is determined by the money amount the address have and the width of the connection between two nodes are determined by the transaction value. The colour of the nodes and transactions could be manipulated. The colour of the transaction were set to different colour depending on the number of transactions between two addresses. The different intervals and the corresponding colour are shown in Table 1.

TABLE 1. CORRESPONDING COLOUR AND INTERVAL FOR THE NUMBER OF TRANSACTIONS.

Colour	Hex	Interval
Dark Blue	#003f5c	0-1
Medium Dark Blue	#374c80	2-10
Blue-Magenta	#7a5195	11-20
Magenta-Pink	#bc5090	21-50
Pink-Red	#ef5675	51-100
Red-Orange	#ff764a	101-400
Yellow-Red	#ffa600	401-700

4. Results

With a given, suspicious, input address *1M3jWAPH6Uq5ZS1GwB1jcfkPuRXBscdXw1* the following results were gathered. Figure 1 shows the Sankey diagram of the unfiltered data, i.e. all of the transactions retrieved from the API.

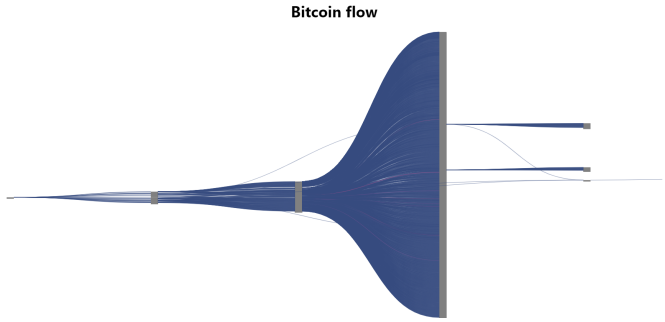

Figure 1. Resulting graph without any filtering

Figure 2 shows the Sankey diagram of the top three transactions of each step and the rest is combined into one node.

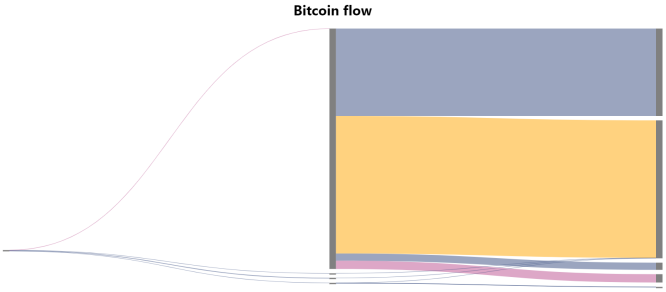

Figure 2. Resulting graph with top three transactions in each step

5. Discussion

The tool uses BlockCyphers API to collect the information about the Bitcoin transactions. The API comes with restrictions regarding how much information can be collected during a period of time as described in section 2.2. This limitation has been a problem during the project since it could take several hours/days to collect a sufficient amount of data. An improvement that could have been implemented is to write an API instead of using a preexisting one. Without using a preexisting API the rate limits could have been avoided and the data would have been faster to retrieve.

As mentioned in section 3.1.1 the transactions made between the same source address and target address were added together to one transaction and a counter was used to keep track on the number of transactions between the two addresses. One negative aspect of bundling the transaction

together is that it is no longer possible to examine the transactions individually, e.g. if there is one large transaction and several smaller ones. However, the amount of data is relatively large and if every transaction between two addresses were used the data would have been difficult to visualize. Therefore this simplification was necessary so that the result was better visualized.

From Figure 1 one can observe without any form of filtering it is hard to gain any real knowledge about the transactions. When applying the filtering of $k = 3$ top transactions, see Figure 2, it became easier to understand how the money was sent from the input address. One negative aspect of this type of filtering was that we lose some information regarding the addresses that are bundled together in the $other$ transactions. The other types of filtering were not as effective as the top k filtering and the visualizations were still very cluttered with many transaction. Therefore it was hard to gain any knowledge about the money flow with the other types of filtering.

To separate the transactions from each other the colour of the transactions were set to a fixed interval, see Table 1, depending on the number of transaction between the addresses. This implementation gave another dimension to the diagram. Some improvements to the tool could be an adaptive interval range of the colours e.g., if all addresses had 22-27 transactions each, they could have different colours. Another way of using the colour dimension could be to show how active the addresses are, for example if an address has been active less than a year or if it has been used during a long period of time. This aspect could be interesting to visualize since it could be suspicious if an address is only active for a short period of time but sends large amount of Bitcoins. Another improvement would be to add the color scale legend, to show which interval the different colors represent, in the final visualization. The colour scale legend was partly implemented, however it was not finished in time for the deadline and therefore it was not added into the final tool.

6. Conclusions

In conclusion, it is possible to visualize Bitcoin data using one suspicious source address as input. However, what can be seen in the result is that the diagrams can easily get cluttered and hard to gain any specific knowledge about the transactions without the correct type of filtering. The best results were obtained when filtering on the k largest transaction for each step. Therefore it is possible to visualize, with varying result, the transaction flow from one suspicious Bitcoin address.

References

[1] S. Nakamoto, “Bitcoin: A Peer-to-Peer Electronic Cash System“, 2008, accessed: 05-03-2021, https://bitcoin.org/bitcoin.pdf

[2] 3Blue1Brown, “But how does bitcoin actually work¿‘, published: 07-07-2017, accessed: 25-05-2021 https://www.youtube.com/watch?v=bBC-nXj3Ng4&t=1243s&ab_channel=3Blue1Brown

[3] Bitcoin.org, “Some things you need to know“, accessed: 06-05-2021, https://bitcoin.org/en/you-need-to-know