Progress Report

Final Year Project

Deanonymization and Traceability of Blockchain-Based Cryptocurrencies: Visual Bit

Supervisor: Dr. Au Allen
Student: Zhang Yixing
UID: 3035449463

Date of submission: January 24, 2021
Abstract

With the blooming of the Bitcoin market, criminal usages of Bitcoin have become an issue. The pseudonymous design of Bitcoin has brought difficulties to the traceability of transactions. This project develops Visual Bit, a tool that can generate transaction graphs and group joint control addresses to analyze the Bitcoin transaction patterns. Until now, the project has finished implementing the data extraction based on block format and transaction network construction using 5 database tables. However, the program is facing an efficiency problem. The current applied solution is to construct partial networks. Another solution that creates the networks on the cloud is also suggested. Other functions, transaction graph generation and address grouping, will be finished by March.
Acknowledgment

I would like to express my greatest gratitude here to all the people who have provided supports during the project, especially the project supervisor, Dr. Au Allen, for his guidance on blockchain in terms of giving suggestions and providing related materials.
# Table of Contents

Abstract .......................................................................................................................... i  
Acknowledgment ............................................................................................................ ii  
List of Figures .................................................................................................................. iv  
List of Tables ................................................................................................................... iv  
Abbreviations .................................................................................................................. iv  

1. Introduction ................................................................................................................... 1  
   1.1. Background ........................................................................................................... 1  
   1.2. Problem ............................................................................................................... 1  
   1.3. Objectives ............................................................................................................ 2  
   1.4. Outline ................................................................................................................. 3  

2. Methodology ................................................................................................................. 3  
   2.1. Data Extraction from the Public Blockchain ....................................................... 3  
   2.2. Transaction Network Construction ..................................................................... 6  
   2.3. Transaction Graph Visualization ......................................................................... 6  
   2.4. Addresses Grouping ............................................................................................. 7  
   2.5. Function-integrated Software Development ....................................................... 8  

3. Current Progress .......................................................................................................... 9  
   3.1. Data Extraction from the Public Blockchain ....................................................... 9  
   3.2. Transaction Network Construction ..................................................................... 10  
   3.3. Address Clustering ............................................................................................. 10  
   3.4. Unit Test & Evaluation ....................................................................................... 10  
   3.5. Difficulty & Limitation ....................................................................................... 12  

4. Future Plan .................................................................................................................. 12  

5. Conclusion ................................................................................................................... 13  

References ..................................................................................................................... I
**List of Figures**

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Size and activity of the sample of &quot;observed&quot; illegal bitcoin users</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>The detailed structure of bitcoin block</td>
<td>4</td>
</tr>
<tr>
<td>3</td>
<td>Entity-centered graph example</td>
<td>7</td>
</tr>
<tr>
<td>4</td>
<td>Transaction-centered graph example</td>
<td>7</td>
</tr>
<tr>
<td>5</td>
<td>Software framework</td>
<td>9</td>
</tr>
<tr>
<td>6</td>
<td>Source code architecture</td>
<td>9</td>
</tr>
<tr>
<td>7</td>
<td>Results of running the program on blk00000.dat</td>
<td>11</td>
</tr>
<tr>
<td>8</td>
<td>Results of data parser</td>
<td>11</td>
</tr>
</tbody>
</table>

**List of Tables**

<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>The basic structure of a block adapted from <em>Mastering Bitcoin</em></td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>The structure of a block header adapted from <em>Mastering Bitcoin</em></td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>The structure of a block body adapted from <em>Mastering Bitcoin</em></td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>Future project schedule</td>
<td>12</td>
</tr>
</tbody>
</table>

**Abbreviations**

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CS</td>
<td>Common Spending</td>
</tr>
<tr>
<td>OTC</td>
<td>One-time Change</td>
</tr>
<tr>
<td>UTXO</td>
<td>Unspent Transaction Output</td>
</tr>
<tr>
<td>GUI</td>
<td>Graphical User Interface</td>
</tr>
</tbody>
</table>
1. Introduction
   1.1. Background
   As blockchain technology advanced rapidly in the last decade, bitcoin, a
decentralized digital currency based on blockchain, has become the most
famous cryptocurrency worldwide. Invented in 2009, bitcoin reached the all-
time highest price of $19,783.06 in 2017. As the Bitcoin market flourishing,
many websites and service providers like American retail giant Overstock
started to accept bitcoin as a payment method. The popularity of Bitcoin has
reached an unprecedented high level.

Despite storing all transactions in a public ledger within a decentralized
network, Bitcoin provides relatively anonymous uses by design. There is no
necessity for real identities in making bitcoin transactions. The only
identifiers of a transaction are the bitcoin addresses, or equivalently, public
key ids of the sender and receiver, making the users hard to trace.

1.2. Problem
   The difficulties in tracing transactions and the wide acceptance of bitcoin
have enabled itself for criminals' illegal financial activities, including money
laundering and payment for drugs or even for murder-for-hire [1]. Figure 1
illustrates that black-market users of Bitcoin accounted for a considerable
percentage of all the users from 2011 to 2017 (figure A), and they made about
half of the transactions (figure C) as well as the holdings (figure D) of
Bitcoin. Figure B shows that the dollar volume of black-market users' Bitcoin
was also relatively high over the years, except for some specific periods. To
catch these criminals, traceability and deanonymization of Bitcoin are
necessary.

Fortunately, bitcoin is never purely anonymous. There are already some ways
to uncover the identities thanks to continuous research. A practical method to
distinguish the owner of Bitcoin is by examining the transaction patterns [1].
Fleder, Kester, and Pillai [2] provided a system to match real users and
transactions by analyzing Bitcoin transaction graphs. Meanwhile, Moser [4]
also proposed some strategies against money laundering making use of transaction graphs.

The critical point of these methods lies in investigating transaction graphs to detect patterns, which requires grouping Bitcoin addresses of joint control and analysis of these addresses’ relationships. Therefore, to facilitate these activities, a tool called Visual Bit that can visualize the transaction networks and group the addresses will be designed in this project.

1.3. Objectives

The project focuses on implementing the software Visual Bit that integrates the previously mentioned functions. Specifically, the software should generate transaction graphs from the public blockchain's data and grouping bitcoin addresses according to the incoming and outgoing transactions. In addition, a user-friendly interface will be developed. Also, if time and
resources are sufficient, the graph generation function's further development will be carried out. For example, various graphs, such as entity-centered and transaction-centered graphs, which can accelerate transaction analysis, will be supported.

1.4. Outline
As follows, this report presents project methodology in 2, current progress in 3, future plan in 4, and conclusion in 5. The methods for transaction data extraction from the public ledger is in 2.1. 2.2 introduces how to use these data to construct a transaction network, followed by illustrations on transaction graph generation in 2.3, addresses grouping in 2.4, and software development in 2.5. The progress part will discuss the results of the works being done so far from 3.1 to 3.3 and show the difficulties encountered in 3.4. The future plan in 4 provides due dates for tasks.

2. Methodology
This section introduces the methodologies for the software's four sub-functions: data extraction, transaction network construction, transaction graph visualization, and addresses grouping. Also, the platform and framework of the software will be discussed.

2.1. Data Extraction from the Public Blockchain
The blockchain storing transaction history of Bitcoin is available publicly. We can obtain the data by running some Bitcoin wallet software. This project uses Bitcoin Core [5], an open-source Bitcoin transaction application, to automatically download the hexadecimal blk*.dat files containing all the blocks generated from 2009. Each blk*.dat file is around 128 MB, and the total size is over 300 GB.

The data structure is vital for the extraction of information, which is introduced by documentation from O'Reilly [6] and posts from Kiran Vaidya [7]. As shown in Table 1 below, the block consists of a magic number, a block size, a header, and a body. Following are the details of the fields:
### Table 1. The basic structure of a block adapted from *Mastering Bitcoin*[6]

<table>
<thead>
<tr>
<th>Field</th>
<th>Size</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Magic Number</td>
<td>4 bytes</td>
<td>An identifier for the blockchain network with a fixed value of 0xD9B4BEF9</td>
</tr>
<tr>
<td>Block Size</td>
<td>4 bytes</td>
<td>The size of the block with a theoretical limit of 4 MB and a realistic limit of 2 MB</td>
</tr>
<tr>
<td>Block Header</td>
<td>80 bytes</td>
<td>Several fields of the block header</td>
</tr>
<tr>
<td>Block Body</td>
<td>Variable</td>
<td>Information about the transactions</td>
</tr>
</tbody>
</table>

The detailed structure of each block is shown in figure 2.

**Figure 2. The detailed structure of bitcoin block**

Block header is made of a version, a previous block hash, a merkle tree root, a timestamp, a difficulty target, and nonce. The details of the block header are listed below (Table 2):

### Table 2. The detailed structure of the block header

<table>
<thead>
<tr>
<th>Field</th>
<th>Size</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td>4 bytes</td>
<td>The version of protocols for each node</td>
</tr>
<tr>
<td>Previous Block Hash</td>
<td>32 bytes</td>
<td>The hash of the parent block calculated by applying SHA-256 twice on all the header fields</td>
</tr>
<tr>
<td>Field</td>
<td>Size</td>
<td>Description</td>
</tr>
<tr>
<td>---------------</td>
<td>--------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Merkle Root</td>
<td>32 bytes</td>
<td>Stores the hash of this block's merkle tree root summarizing all the transactions contained in this block</td>
</tr>
<tr>
<td>Timestamp</td>
<td>4 bytes</td>
<td>The approximate time for creating this block encoded as a Unix 'Epoch' timestamp based on the seconds passed after January 1, 1970, midnight UTC/GMT</td>
</tr>
<tr>
<td>Difficulty Target</td>
<td>4 bytes</td>
<td>Representation of the difficulty in calculating the expected results</td>
</tr>
<tr>
<td>Nonce</td>
<td>4 bytes</td>
<td>A counter which incremented every 10 minutes while solving the puzzle using the proof-of-work algorithm</td>
</tr>
</tbody>
</table>

Table 2. The structure of a block header adapted from *Mastering Bitcoin* [6]

A blockchain body contains a transaction counter and a list of transactions (Table 3). Each transaction is at least 250 bytes on average, and each block has more than 500 transactions in general.

<table>
<thead>
<tr>
<th>Field</th>
<th>Size</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transaction Counter</td>
<td>1-9 bytes (VarInt)</td>
<td>Number of transactions</td>
</tr>
<tr>
<td>Transactions</td>
<td>Variable</td>
<td>Transactions in the block</td>
</tr>
</tbody>
</table>

Table 3. The structure of a block body adapted from *Mastering Bitcoin* [6]

As the data structure of the Bitcoin block is formatted in the order of the three tables, we can read the bits from blk*.dat files in sequence and get the values for each field. Any desired information can be extracted accordingly.

After getting the output script for a transaction from the transaction lists, we need to translate it into address. There are 8 types of scripts in general: P2PK (pay-to-public-key), P2PKH (pay-to-public-key-hash), P2SH (pay-to-script-hash), P2WPKH (pay-to-witnessed-public-key-hash), P2WSH (pay-to-witnessed-public-script-hash), P2MultiSig and OP_RETURN. P2PK, P2PKH, P2SH’s addresses are encoded in Base58 while P2WPKH and P2WSH’s addresses are encoded in Bech32. However, the P2MultiSig and
OP_RETURN scripts do not have standard addresses. Therefore, we would not discuss them here.

To calculate the Base58 address, we first get the public key hash from P2PK and P2PKH script or script hash form the P2SH script. Then we add network prefix 00 to public key hash and 05 to script hash, which can be denoted as H. After calculating the hash of H twice and adding this result’s first 4 bytes to the end of the H, we can finally encode it in Base58.

As for the Bech32 address, we add network prefix in the front and checksum in the end of the hash. After that, we translate the result in Bech32. The final address is ‘bc1’ followed by the Bech32 result.

2.2. Transaction Network Construction

After extracting transaction data from the public ledger, the software will store it in the database due to the massive number of transactions, making it easier to manage. The database should have the following tables:

- Transaction (tx_hash, block_hash, is_coinbase)
- Input (input_id, tx_hash, output_tx_hash, output_index)
- Output (tx_hash, output_index, address, value, tag)

The tx_hash fields in the Input and Output tables are foreign keys to the tx_hash in the Transaction table, which links the input and output to a particular transaction. The output_tx_hash and output_index in Input table are foreign keys to the tx_hash and output_index in the Output table respectively. They link the input to the previous transaction’s output where it comes from. In this way, we can find all the related transaction addresses given a specific transaction or an address. In other words, a transaction network is constructed.

2.3. Transaction Graph Visualization

There are two types of graphs for selection. One is the entity-centered graph; the other is the transaction-centered graph. The entity in an entity-centered graph (Figure 3) is a group of joint control addresses, which means these addresses belong to the same entity. Moreover, in this type of graph, entities represented by circles are the core of the graph. Each entity is linked to other
entities that have transactions between them. For a transaction-centered graph (Figure 4), the transactions are linked in a chain. Each input of one transaction can be the output of the previous transaction, and each output can be the input of the next transaction.

![Entity-centered graph example](image1)

Figure 3. Entity-centered graph example

![Transaction-centered graph example](image2)

Figure 4. Transaction-centered graph example

In this project, entity-centered graph generation will be first implemented due to its popularity for transaction pattern analyses. Furthermore, if time allows, transaction-centered graph generation will also be developed.

### 2.4. Addresses Grouping

The most straightforward idea for grouping Bitcoin addresses of joint ownership is marking all the input addresses of a transaction as belonging to one entity. This method is called *Common Spending (CS)* by Ermilov, Panov, and Yanovich [8]. Only the addresses of joint control can be used to pay in a transaction because the private keys for the addresses are needed to sign on the transaction.

Another method is *One-time Change (OTC)* [8]. The UTXO change of a transaction will go to a new address by design, which should also belong to the input addresses' owner. The dilemma is to find this change address from
the outputs. Many wallet software puts the change address in the last position while some arrange the output sequence randomly. One possible solution is to mark the transaction which has only one output as a “change transaction”, which means its input addresses and output addresses belong to one entity, because it is very unlikely that the inputs have the exact amount of UTXO paid to the output. The transactions with one output can be seen as the owner is reallocating the UTXO.

The third method is to use additional information out of the block. For example, when Alice is posting an address on the blog asking for donation, we can give this address a tag “Alice”. All the addresses with the same tag should belong to one entity. By doing so, we can link many groups of addresses which do not have interaction in the transaction network together.

2.5. **Function-integrated Software Development**

2.5.1. **Software Language and Platform Selection**

Due to the excellent readability of script language and wide use of Windows, this project's software will be implemented in Python 3.8 on a windows machine. PyQt will be used for the GUI development, while some libraries, such as matplotlib, will be included to help graph generation. Database MySQL will be used for data storage.

2.5.2. **Software Framework Design**

The software should consist of three layers (see Figure 5): front end (UI), back end (business logic), and infrastructure (data). The front end is the user interface. The back end has three modules: transaction network construction module, transaction graph generation module, and addresses grouping module. All these modules are based on the infrastructure database.
Using the methodologies introduced above, the software will first extract the data from the public ledger and then construct a network based on the data in the database. After that, the network will be visualized by two kinds of graphs. The software will group addresses of joint control through the networks at the same time.

3. Current Progress
The first two functions of the software, data extraction and transaction network construction, as well as part of the address grouping, were finished. And a unit test was carried out, of which the results faced an efficiency issue. The program’s source code architecture is shown in figure 6.

- **Main program**: VisualBit.py
- **Utilities**:
  - classHelper.py
  - config_default.py
  - dbManager.py
  - test.py
- **Data Extraction & network construction**:
  - parser.py
  - streamer.py
- **Address clustering**:
  - cluster.py

3.1. Data Extraction from the Public Blockchain
Using the data extraction techniques introduced in the methodology section, the class DataReader was implemented in the file parser.py. When running the main program, it called some of this class's methods and did the
following. The app first loaded the blk*.dat files with the specified file ids. Then data stored in the structures was unpacked and analyzed, after which the program put them into a list with format [block hash, timestamp, number of transactions, [transaction hash, [inputs], [outputs]]] ([] stands for lists).

3.2. Transaction Network Construction
The methods in the class DataReader also supported functions of storing data processed into the database, which made use of the methods in another class DBManager implemented in dbManager.py. It established connections to the database server and the server's databases, respectively. In addition, the program also created databases on the connected server or tables on the connected database. The server's hostname, username and password for login, and the database's name were stored in the config.py file. When the software finished extracting data, it continued to store the data in the well-structured database as a transaction network. In the meantime, the data can be optionally written into several txt files in human readable format for browsing transaction network manually.

3.3. Address Clustering
CS method and common tag method are supported in the cluster.py. The CS method queried the transaction network for each transaction’s inputs first. And for each transaction, it gave all of its input addresses a same new tag. After that, it made use of the common tag method to group addresses with common tags as one entity. An address can have multiple tags and two addresses belong together if they share at least one tag. This method queried for addresses lists with all the possible tags one by one, and group them accordingly.

3.4. Unit Test & Evaluation
A simple unit test was carried out on the first block file blk00000.dat, which only contained the first Bitcoin transaction in history. As shown in Figure 7, the program first connected to the MySQL server set up in advance, which was the localhost in this case. Then it tried to establish a connection to the database "bitcoin" but found that it was not created yet. Therefore, the app built this database as well as all the tables and then connected to it. After that,
the analysis of the block file data began. It extracted the hash and timestamp of the block and counted the number of transactions. Then it printed out all the transactions with their hash, input addresses, and output addresses. Finally, it stored the data processed into the database.

The results in Figure 7 and Figure 8 show that the demo program is implemented successfully. It achieves the two functions' designated goals: extracting useful information including block hash, timestamp, and transaction details from block files and storing data into the database to construct a transaction network.
3.5. Difficulty & Limitation

Despite meeting the minimum requirements, the software is facing the problem of inefficiency. Only one blk*.dat file is processed in the unit test, which costs around 20s. However, when increasing the number of block files, running time is becoming unacceptable. There are over 2300 block files in total, and if we want to analyze them all, it will take a considerable amount of computation time: 4600s.

The current applied solution is specifying the range of files. Most of the time, users only want part of the transactions for analysis. We can cut down the computation efforts by only extracting data from the selected files. Another future solution is building databases in the cloud instead of locally. By doing so, we will only have to maintain the block data online and extract the old blocks' information once. Users will connect to the same cloud database to retrieve data avoiding computing locally. The cloud server will be responsible for processing data when a new bitcoin block is generated.

4. Future Plan

As shown in Table 2, this project will finish implementing the graph generation function before the end of March. Until April 17, I will do some testing and integrating the software demo. Finalized tested software, and the final report will be available on 18. After that, a final presentation will be given following the department’s arrangement.

<table>
<thead>
<tr>
<th>Time</th>
<th>Tasks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Current - 2021.03</td>
<td>Implementation of the graph generation function</td>
</tr>
<tr>
<td>2021.03 - 04</td>
<td>Testing and integration</td>
</tr>
<tr>
<td>2021.04 - 2021.04.18</td>
<td>Finalized tested software with graph generation and addresses clustering functions</td>
</tr>
<tr>
<td>2021.04.19 - 23</td>
<td>Final report</td>
</tr>
</tbody>
</table>

Table 4. Future project schedule
5. **Conclusion**

This progress report introduces the project Visual Bit, a Bitcoin analysis tool that can help analyze bitcoin transaction patterns. The software can extract useful information from bitcoin's public ledger following three tables' block format and construct a network by putting them into the database's five connected tables. And it can generate two kinds of graphs, entity-centered and transaction-centered graph, to visualize the networks. Addresses of joint control will also be grouped by CS and OTC methods. Currently, the project has achieved the first two functions while facing an efficiency problem. The software adopted the partial construction method to reduce running time and may switch to cloud construction to solve the problem once for all in the future. The remaining functions of graph generation and address grouping will be finished by April. In the end, this project will deliver Visual Bit that can help fight against illegal usage of bitcoin.
References


