The University of Hong Kong

Faculty of Engineering

Department of Computer Science

Project Plan for Final Year Project

Deanonymization and Traceability of Blockchain-Based Cryptocurrencies

By Zhang Yixing
Supervised by Dr. Au Allen

2020-2021
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 by an anonymous identity called Satoshi Nakamoto in 2009, bitcoin reached a price of $770 in 2013 and an even higher price of $19,783.06 in 2017, which is also the all-time highest price of Bitcoin. Many websites and service providers such as American retail giant Overstock, Japanese commercial company Recruit Lifestyle, the food ordering website Pyszne.pl in Poland are now accepting bitcoin. The market treats this blooming cryptocurrency as useful as real money.

Despite storing all transactions in a public ledger within a decentralized network, Bitcoin provides relatively anonymous uses by design. There is no necessity of 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 associated itself with the illegal financial activities of criminals, including money laundering and payment for drugs or even murder-for-hire [1]. Figure 1.2 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 at a relatively high level over the years, except for specific periods. To catch these criminals, traceability and deanonymization of Bitcoin are necessary.
Figure 1.2. Size and activity of the sample of "observed" illegal bitcoin users [1]

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 [2]. Besides, Fleder, Kester, and Pillai [3] 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 grouping addresses of joint control and analysis of these addresses' relationships. Therefore, to facilitate these processes, a tool that can visualize the transaction networks and group the addresses will be designed in this project.

1.3. Objectives

The project focuses on implementing software that integrates the previously mentioned functions. In specific, the software should be capable of generating transaction graphs from the data in the public blockchain and grouping bitcoin addresses according to the incoming and outgoing transactions.
The intermediate objective for this project is to provide a demo software that can extract transaction information automatically from the public ledger, form a transaction network, and stored them in certain data structures.

The final deliverable will be an application that can generate graphs from the transaction networks and group addresses with joint ownership. As a bonus, a user-friendly interface will also be implemented. Also, if time and resources are sufficient, further development of graph generation function will be carried out. For example, various categories of graphs, such as entity-centered and transaction-centered graphs, which can accelerate the analysis of transactions, will be supported.

1.4. Outline

As follow, this project plan presents project methodology, project schedule, and conclusion in order. The methodology includes the methods for data extraction, transaction network construction, transaction graph generation, addresses grouping, and software development. The schedule provides due dates for tasks and milestones.

2. Methodology

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 makes use of Bitcoin Core [5], an open-source Bitcoin transaction application, which will automatically download the hexadecimal blk*.dat files containing all the blocks generated from 2009 after running the application for the first time. Each blk*.dat file is around 128 MB, and the total size is over 300 GB for now.

The data structure is vital for the extraction of information, which is introduced by documentations from O'Reilly [6] and posts from Kiran Vaidya [7]. As shown in Table 2.1.1 below, the block consists of a magic number, a block size, a header, and a body. Following are the details of the fields:

- **Magic Number** (4 bytes) is an identifier for the blockchain network with a fixed value of 0xD9B4BEF9. It represents the start of the block.
- **Block Size** (4 bytes) indicates the size of the block. It has a theoretical limit of 4 MB and a more realistic limit of 2 MB currently.
- **Block Header** (80 bytes) will be discussed in the following paragraph.
- **Block Body** (variable) will be introduced shortly.
<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</td>
</tr>
<tr>
<td>Block Size</td>
<td>4 bytes</td>
<td>The size of the block</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>

Table 2.1.1. The basic structure of a block [6]

Block header is made of a version, a previous block hash, a merkle tree root, a timestamp, a difficulty target, and a nonce. The details of the block header are listed below (see Table 2.1.2):

- **Version** (4 bytes) indicates the version of protocols each node should be running.
- **Previous Block Hash** (32 bytes) is the hash of the parent block, calculated by applying SHA-256 twice on all the header fields.
- **Merkle Root** (32 bytes) stores the hash of this block's merkle tree root. A merkle tree root is a data structure summarizing all the transactions contained in this block.
- **Timestamp** (4 bytes) is encoded as a Unix 'Epoch' timestamp based on the seconds passed after Jan. 1, 1970, midnight UTC/GMT. It is not very accurate, and the main aim of this field is to add difficulty to hashing.
- **Difficulty Target** (4 bytes) represents the difficulty in calculating the expected results.
- **Nonce** (4 bytes) is a counter which incremented every 10 minutes while solving the puzzle using the proof-of-work algorithm.
<table>
<thead>
<tr>
<th>Field</th>
<th>Size</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td>4 bytes</td>
<td>A version number of Bitcoin protocol</td>
</tr>
<tr>
<td>Previous Block Hash</td>
<td>32 bytes</td>
<td>The previous block's (parent block's) hash</td>
</tr>
<tr>
<td>Merkle Root</td>
<td>32 bytes</td>
<td>The hash of the root of this block's merkle tree</td>
</tr>
<tr>
<td>Timestamp</td>
<td>4 bytes</td>
<td>The approximate time for the creation of this block</td>
</tr>
<tr>
<td>Difficulty Target</td>
<td>4 bytes</td>
<td>The difficulty target of this block in the proof-of-work algorithm</td>
</tr>
<tr>
<td>Nonce</td>
<td>4 bytes</td>
<td>A counter used for the proof-of-work algorithm</td>
</tr>
</tbody>
</table>

Table 2.1.2. The structure of a block header [6]

A blockchain body contains a transaction counter and a list of transactions (see Table 2.1.3). Each transaction is at least 250 bytes averagely, 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 2.1.3. The structure of a block body [6]

The structure for a specific transaction will be included in the final report for a more straightforward explanation.

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

- **Entity** (Entity ID, Name, Remark)
- **Address** (Address hash, Entity ID)
- **Transaction** (Transaction ID, Input address number, Output address number, Confirmed time)
- **Input** (Input ID, Transaction ID, Address hash, Value)
- **Output** (Output ID, Transaction ID, Address hash, Value)

The `Entity ID` in the `Address` table is a foreign key to the `Entity ID` in the `Entity` table, which links the bitcoin address to a particular user. This field can be `Null` because it is usually not known whom the address belongs to. The `Transaction ID` in the `Input` and `Output` tables are foreign keys to the `Transaction ID` in the `Transaction` table. The `Address hash` in both tables are foreign keys to the `Address hash` in the `Address` table, which links the transaction and its corresponding input addresses and output addresses. 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. Visualization of transaction graph

There are two types of graphs for selection. One is the entity-centered graph (see figure 2.3.1), the other is the transaction-centered graph (see figure 2.3.2). The entity in an entity-centered graph is a group of addresses of joint control, 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 each transaction, two arrows and one rectangle are used as representations. Two entities can have multiple transactions in between.

![Figure 2.3.1. Entity-centered graph example](image-url)
For a transaction-centered graph, the transactions are linked in a chain. Each input of one transaction can be the output of another transaction, and each output can be the input of another transaction.

![Transaction-centered graph example](image)

In this project, entity-centered graph generation will be first implemented. Furthermore, if time allows, transaction-centered graph generation will also be developed. However, this will depend on the actual progress of the project.

### 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 [7]. 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)* [7]. By design, the UTXO change of a transaction will go to a new address, which should also belong to the owner of the input addresses. 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. More research on this problem is needed during the project.

Making use of both methods, we can conclude that for a transaction, all the input addresses and the change address should belong to one group.

### 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, the software of this project will be implemented in Python 3.7 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: 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.

![Software framework diagram](image)

Figure 2.5.2 Software framework

3. **Schedule**

<table>
<thead>
<tr>
<th>Time</th>
<th>Tasks</th>
</tr>
</thead>
<tbody>
<tr>
<td>2020.10.04</td>
<td>Deliverables of Phase 1 (Inception)</td>
</tr>
<tr>
<td></td>
<td>- Detailed project plan</td>
</tr>
<tr>
<td></td>
<td>- Project web page</td>
</tr>
<tr>
<td>2021.01.11 - 15</td>
<td>First presentation</td>
</tr>
<tr>
<td>2021.01.24</td>
<td>Deliverables of Phase 2 (Elaboration)</td>
</tr>
<tr>
<td></td>
<td>- Demo software with data extraction and transaction network construction functions</td>
</tr>
<tr>
<td></td>
<td>- Detailed interim report</td>
</tr>
<tr>
<td>2021.04.18</td>
<td>Deliverables of Phase 3 (Construction)</td>
</tr>
<tr>
<td></td>
<td>- Finalized tested software with graph generation and addresses clustering functions</td>
</tr>
<tr>
<td></td>
<td>- Final report</td>
</tr>
<tr>
<td>2021.04.19 - 23</td>
<td>Final presentation</td>
</tr>
</tbody>
</table>

Table 3. Project schedule
4. Conclusion

In this project plan, by introducing the background of Bitcoin, the criminal usages of Bitcoin, the primary methodologies to be implemented, and the schedule of the project, the general information of this final year project *Deanonymization and Traceability of Blockchain-Based Cryptocurrencies*, is clarified. However, this is only a proposal for the project. Methods, implementation, and schedule may still change according to the actual progress of the project.
REFERENCES


