BITSCOPE: Scaling Bitcoin Address De-anonymization using Multi-Resolution Clustering

Zhen Zhang, Tianyi Zhou, and Zhitong Xie

University of Washington

Abstract

Bitcoin is a popular crypto-currency designed to protect anonymity of users. However, by analyzing the transaction pattern, tracking the fund flow, and associating addresses with real entities, it is possible to defeat this anonymity. One of the key challenges is scalability - it is difficult to produce useful results in a short time, while keeping up with terabytes of increasingly large blockchain data. We propose BITSCOPE, a de-anonymization system that uses a layered approach and exploits the domain-specific structures in Bitcoin transaction network. We used BITSCOPE to process a graph of nearly 1 billion nodes and evaluated its de-anonymization performance on a real world dataset.

1 Overview

1.1 Introduction

Bitcoin (BTC) is the first, the most popular and also the most valuable crypto-currency based on blockchain technology. It has been used by people for various purposes since its genesis in 2008. As a currency, its design features anonymous transaction and decentralized governance. At the same time, it is also used as an investment/asset which is traded 24x7 globally, and a programmable platform for many other decentralized financial services.

The anonymity of Bitcoin comes from the design that anyone can generate an unlimited number of addresses for receiving or sending Bitcoin. Bitcoin encourages users to create new addresses for receiving money and dispose them after use to protect privacy. Also, many Bitcoin clients support a feature called "wallet", which automatically manages the process of creating and maintaining a pool of addresses. However, this anonymity design only works in an ideal world where (1) all users uses Bitcoin in the recommended way (i.e. not reusing addresses) (2) all users make transactions with each other randomly in a decentralized manner (i.e. there is no centrality or community) (3) all users never leak their addresses to anyone other than the senders (counter-example: many people post their BTC address on personal website or forum user profile to receive donations).

In the real world, we have to make trade-offs between security and other aspects like usability and cost. Data mining can defeat anonymity by exploiting the trade-offs people make on the above three rules. Combining the publicly accessible transaction history with other metadata (like the famous work [1] that associates forum users with their Bitcoin addresses), we may be able to reveal (i.e. **de-anonymize**) the real entities (owners) behind anonymous addresses. Real entity or owner might either be a person or an organization, depending on what kind of metadata we have.

There have been many previous work on this topic since 2010. The data analysis and feature selection is pretty mature now, and the community has accumulated a few publicly available datasets [2][3]. However, they don't scale well to the full Bitcoin network (1M+ addresses and 0.4B+ transactions).

Preprint. Under review.

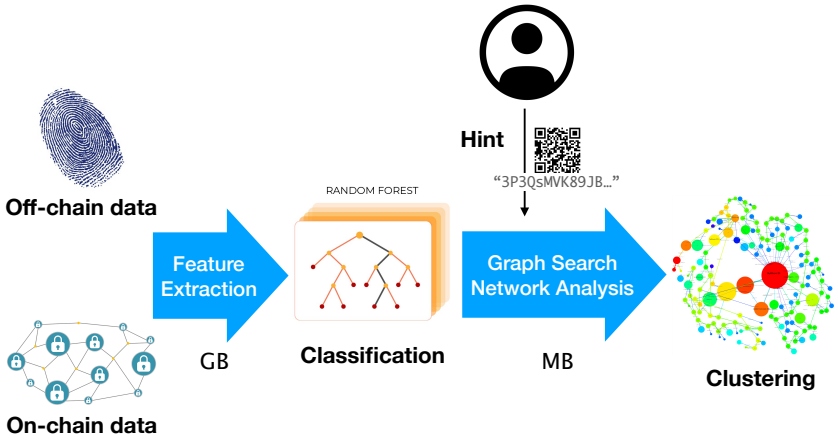


Figure 1: Overview of BITSCOPE

For example, the work from Yu-Jing Lin et al. [4] use a Bitcoin dataset with 26,308 addresses and 10M+ transactions and the work from Thai Pham and Steven Lee [5] use a dataset with 0.03B+ transactions. The clustering result is not very good either. We think this is because their systems didn't take the full advantage of domain specific properties in Bitcoin network. This motivates our idea of scaling de-anonymization by designing three resolution layers for classification and clustering. We will discuss more about them in section 7.

Our work[1] can potentially help government to investigate criminal activities that use Bitcoin. For example, the illegal Silkroad website on Darknet used Bitcoin as a payment method between buyers and sellers. On the other hand, normal users could be reminded about the privacy risk of Bitcoin.

1.2 Multi-Resolution Clustering for De-anonymization

We propose a multi-resolution clustering system called BITSCOPE (Figure 1) for better scalability. Our intuition comes from the following observations: (1) **Behavioral difference between human and bot**: In recent years, there is a growing trend of commercially-purposed, program-driven ("bot") transactions happening on blockchain, mixed with human users' activities. (2) **Symmetrical P2P structure versus asymmetrical service-user structure**: Bitcoin is designed as a P2P payment system, which should form a mostly symmetrical community structure between users. However, more traditional asymmetrical service-user structure is becoming increasingly dominant where a group of users center around an online service (which consists of many bot addresses). Our intuition is that we can first coarsely locate a suspicious address cluster, then "zoom in" and run more expensive inferences in this smaller cluster to understand which subset of addresses might belong to the same entity.

First resolution layer: address classification considers the "aggregated features". Using these features, we can build a classifier for basic categorization of addresses to classify between individual and service, between human and machine, or between exchange service and gambling service, depending on the labels we have. In practice, these data are linear to the size of graph; thus, we can incrementally update them and maintain the latest snapshot on disk or even in memory.

Second resolution layer: service-user communities. User constructs a query either as a set of addresses, or as a class label (in which case all addresses in that class will be considered). The system will use network analysis to break up the address set into smaller, closely related communities.

Final resolution layer: service address clustering, we will use network analysis for topological clustering and communities detection in this phase. They require us to consider the whole graph globally and compute iteratively, and thus requiring significantly more computation than previous resolution layers. However, our assumption is that these expensive features are most valuable in this resolution. After clustering, users can cross-validate the result with heuristic-based contraction.

[1] We will publish relevant data and code on https://izgzhen.github.io/bitscope-public.

2 Date Preparation and Statistics

2.1 Dataset Sources

We make our dataset based on the previous datasets and online data service [2] (we will give them short names here for future references):

- **ELTE dataset**[3] (ELTE): This dataset contains Bitcoin blockchain transactions. For each transaction, there can be multiple senders ("inputs") and multiple receivers ("outputs"). The latest version contains 397,571 blocks up to 2016.02.09.
- **BigQuery crypto_bitcoin dataset** (BQ): This crypto_bitcoin dataset is part of Google Cloud BigQuery Public Dataset program. It contains all the bitcoin transaction data from 2009/01/01 till now, updated every 24 hours. This dataset has two tables "blocks" (170 MB with 573,100 rows) and "transactions" (912 GB with 405,973,932 rows).
- **Walletexplorer.com**[6] (WE): This website maps some BTC address to entity name, and categorized into different types. One previous work by Toyoda et al. [2] collect a dataset of 26314 rows and 4 columns: address, owner, service name, class based on this.
- **Blockchain.com**[7] (BTAGS) Blockchain.com/tags contains tags for some addresses. Users label addresses based on their experience and website will verify their suggestions afterwards to determine the trustfulness. We crawled 1500 items from 73 pages using `scrapy` and replenished our ground truth with these newly collected data.

2.2 Extended Dataset

We have two kinds of datasets: on-chain and off-chain. On-chain data is purely blockchain transactions. Off-chain data indicates data gathered from external sources like website etc., which refers to things on blockchain. Our on-chain data is customized and extended based on ELTE and BQ (no need to parse transactions ourselves). Below is the introduction of the final datasets and their schema.

On-chain data The on-chain data is based on the second version of ELTE and extended with latest transactions collected from BQ. We downloaded 33,883,582 rows of tx inputs and 34,024,128 rows of tx outputs from BQ, and transformed them to the same format as ELTE. As a result, we added 5789 new addresses and 17,998,798 new transactions to the ELTE. The final on-chain dataset contains four tables (`addr` is bitcoin address, `txhash` is transaction hash; `addrID` and `txID` are all indices):

- `addresses (addrID, addr)`: 124,863,654 rows
- `tx (txID, txhash, timestamp)`: 126,995,411 rows
- `txin (value, addrID, txID, index, timestamp)`: 279,784,279 rows
- `txout (value, addrID, txID, index, timestamp)`: 313,751,127 rows

Off-chain data Our off-chain data is based on the previous work by Toyoda et al.[2]. We extend and update it with newest tag data from BTAGS and WE. We compared the similarity between two sources of literal descriptions of the real entities and confirmed the consistency.

The final extended off-chain dataset is a table of the following columns:
`offchain (addr, tag, link, name, class, owner)`. It has 31,422 rows. As a summary of our extension and customization efforts, there are 31,199 addresses shared between on-chain dataset (0.024% of total on-chain addresses) and off-chain dataset (99.29% of total off-chain addresses).

3 Data Analysis

3.1 On-chain Data Analysis

In-degree and out-degree distribution of addresses We sort the on-chain dataset by the frequency of addresses with the same number of in-/out-degrees. The distribution conforms to *the Power Law*.

[2] During dataset searching, we find that there is no updated and publicly accessible ground-truth dataset that we can directly download. Also, many previous work typically collected their own data without standardizing them for public use. Therefore, we write our own data crawlers. Also, the UIC dataset's server is not accessible at the time of writing. We do not consider it.

Note that there might be multiple edges connecting one unique address and one unique transaction. Therefore, we also calculate the *unique* in-/out-degrees by merging these edges. We find that the distribution of *unique* in-/out-degrees is same as the original one besides some negligible variance.

Top 10 address with most transactions Among the top-10 addresses, we found that addrID '114263245' (BTC address: '1PEDJAibfNetJzM289oXsW1qLAgjYDjLgN') has very high transaction frequencies in both txin and txout. This address is from wallet "CoinJoinMess". We believe that it is a bot service considering its unusual action history and a third-party article on "CoinJoin" [8].

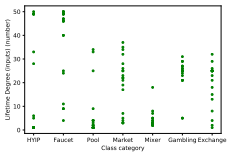
Similarity of transaction activities over time-span As proposed by Yu-Jing Lin et al. [4], time-series related features can improve prediction performance. We validate this and find that addresses belonging to one entity overlap more in transaction histories compared to randomly sampled ones.

3.2 Off-chain Data Analysis

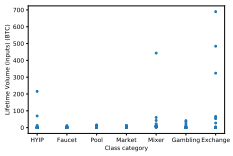
Off-chain data feature selection All the addresses from the off-chain data are labeled as one out of the seven classes, which are 'HYIP', 'Pool', 'Faucet', 'Market', 'Mixer', 'Gambling', and 'Exchange'. For feature exploration and selection, we randomly sample 20 addresses and obtain their transaction records. We plot the distribution of features "lifetime volume (outputs)", "lifetime volume (inputs)", "lifetime degree (outputs)", "lifetime degree (inputs)", "first active time", "last active time", "lifetime transaction time-span", "lifetime volume sum", and "average transaction time interval" to validate their usability.

We find that differences in volume (Figure 2b) for each class are not as significant as those in degrees (Figure 2a). Also, "lifetime transaction time-span" and "average transaction time interval" have significant differences in their distributions (Figure 2c [3] and Figure 2d).

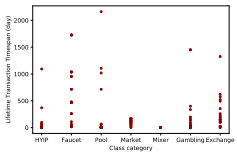
Based on our exploratory analysis, we use all the features proposed above except for "first active time", "last active time", and "lifetime volume sum".



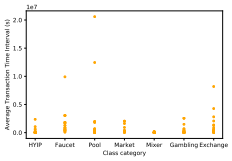
(a) Lifetime degree (inputs). Class HYIP and Faucet have the largest ones



(b) Class Exchange has the largest range, Faucet, Pool, and Market are around zero



(c) lifetime transaction time-span: Mixer has a small, while Pool has a large range



(d) Average transaction time interval. Mixer is different from others classes

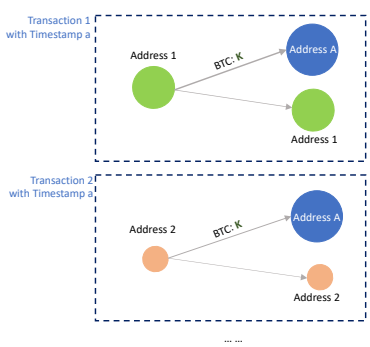
3.3 Pattern Analysis

After studying the bitcoin transaction records carefully, we find several interesting patterns that we may use in our clustering. (due to lack of space, we will only show two illustration of them here).

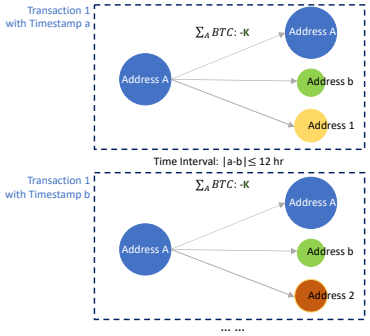
Suspicious address always receives the same amount of input from other addresses This pattern is special because there is only one input address and only two output addresses (the address in the input side and the suspicious address) in the output side for each transaction. Moreover, the time interval between two of the transaction records under such a pattern is always negligible (Figure 3a).

Suspicious address sends the same amount of Bitcoin in several transactions sharing a specific receiver within a short time frame The suspicious address under this pattern always exists both in the input and output sides of each transaction record. Also, the net amount of Bitcoin transferred out of the suspicious address in each record is the same. One thing should be noticed is that there always exists one certain address in the output side of each record that the suspicious address involved. Also, the time interval of two transaction should be no longer than 12 hours. (Figure 3b)

[3] As a side note, we can validate our claim in section 1 that address reuse is a serious problem from the time-span distribution in Figure 2c



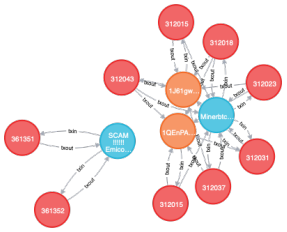
(a) Address A always receives the same amount of input from others.



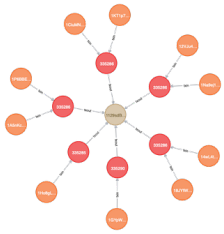
(b) Address A always has the same amount of money transferred out to one specific receiver in a short time frame

3.4 Micro Transaction Network Structures

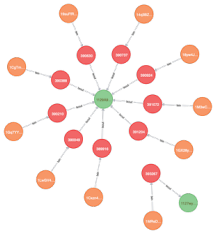
Here we present several of real transaction network structures (Figure 4).



(a) HYIP related transaction sub-graph



(b) Gambling related transaction sub-graph



(c) Exchange related transaction sub-graph

Figure 4: Different classes of addresses have totally different sub-graph patterns, here we present some interesting addresses to help explanation. For HYIP (High Yield Investment Programs) addresses, the interactions of central addresses and surrounding ones are frequent and sometimes there are even multiply edges, indicating high-volume transactions. As we know, investment actions has a relative higher transaction volume and time-span. The sub-graph pattern of 'Gambling' class has two explicit features: 1) There are more 'txout' than 'txin' edges in network.2) mutual transactions are rare and many addresses never get bitcoin back. The other class of addresses follow a normal pattern in which they play a transfer role between different addresses and the distribution of 'txin' and 'txout' are balanced surrounding central nodes. However, one tricky thing is that some addresses may have a extremely high centrality and behave like 'HYIP' nodes.

4 Data Structure and Algorithms

4.1 Graph Formalization

Bitcoin is a complex system as in subsection 1.1. Thus, we will discuss on top of a simplified formal model. We represent the *state* of bitcoin network as a directed *labeled property graph*.

Graph nodes have two types: address $(a, b, ... : Addr)$ or transaction $(t_1, t_2, ... : T)$. Graph edges represent the flow of value in transaction, either as an input $a \to t$, or an output $t \to b$. There is no direct edge between two addresses or two transactions. We use a special node "coinbase" to represent the creation of value during mining. Graph is a tuple of nodes and edges: $G \triangleq (\mathcal{N}, \mathcal{E})$.

Each address node has the following attributes/tags: category (string or null), i.e. a_x has category x. Each transaction node has attributes/tags: timestamp (date-time), i.e. t_k has timestamp k. Each edge has the following attributes/tags: value (positive float), formally $a \to_m t_1$ for value m.

There might be multiple edges between nodes, but they must be in the same direction. The set of edges between a and t is written as $\mathcal{E}(a, t)$. we define the set of transactions related to address a as $T(a) \triangleq \{t \mid \mathcal{E}(a, t) \cup \mathcal{E}(t, a) \neq \emptyset\}$

We define an undirected graph, called address graph $G_{\mathcal{A}}$, derived from transaction-address graph G: $G_{\mathcal{A}}$'s nodes are G's address nodes $\mathcal{N}_G$, and a is connected with b in $G_{\mathcal{A}}$ iff. exists $t \in \mathcal{N}_G$ such that $(a, t) \in \mathcal{E}_G \wedge (t, b) \in \mathcal{E}_G$ or $(b, t) \in \mathcal{E}_G \wedge (t, a) \in \mathcal{E}_G$. There will be at most one edge between any two address nodes, representing whether they had transacted before.

$$D_{\text{timestamp-out}}(a) \triangleq \{k \mid \forall t_k, \mathcal{E}(a, t_k) \neq \emptyset\} \quad D_{\text{timestamp-in}}(a) \triangleq \{k \mid \forall t_k, \mathcal{E}(t_k, a) \neq \emptyset\}$$

$$D_{\text{timestamp}}(a) \triangleq D_{\text{timestamp-out}}(a) \cup D_{\text{timestamp-in}}(a) \quad S_{\text{timestamp}}(a) \triangleq \{k \mapsto t_k \mid \forall t_k, \mathcal{E}(a, t_k) \neq \emptyset, \text{ordered by } k\}$$

$$D_{\text{degree-out}}(a) \triangleq \{|\mathcal{E}(a, t_k)| \mid \forall t_k, \mathcal{E}(a, t_k) \neq \emptyset\} \quad D_{\text{degree-in}}(a) \triangleq \{|\mathcal{E}(t_k, a)| \mid \forall t_k, \mathcal{E}(t_k, a) \neq \emptyset\}$$

$$D_{\text{value-out}}(a) \triangleq \{\sum_{e_v \in \mathcal{E}(a, t_k)} v \mid \forall t_k, \mathcal{E}(a, t_k) \neq \emptyset\} \quad D_{\text{value-in}}(a) \triangleq \{\sum_{e_v \in \mathcal{E}(t_k, a)} v \mid \forall t_k, \mathcal{E}(t_k, a) \neq \emptyset\}$$

$$D_{\text{timestamp-interval}}(a) \triangleq \{k_{i+1} - k_i \mid \forall i, t_{k_i} = S_{\text{timestamp}}(a)[i], t_{k_{i+1}} = S_{\text{timestamp}}(a)[i+1]\}$$

4.2 Aggregated Features

- **Lifetime volume (inputs)**: $LTV_I(a) \triangleq \sum D_{\text{value-in}}(a)$
- **Lifetime volume (outputs)**: $LTV_O(a) \triangleq \sum D_{\text{value-out}}(a)$
- **Lifetime degree (inputs)**: $LTD_I(a) \triangleq \sum D_{\text{degree-in}}(a)$
- **Lifetime degree (outputs)**: $LTD_O(a) \triangleq \sum D_{\text{degree-out}}(a)$
- **Lifetime transaction time-span**: $LTTT(a) \triangleq \max(D_{\text{timestamp}}(a)), \min(D_{\text{timestamp}}(a))$
- **Average transaction time interval**: $ATTI(a) \triangleq \text{Avg}(D_{\text{timestamp-interval}}(a))$

4.3 Iterative Search on Address Graph

Our experience with analyzing the dataset is that instead of randomly choosing a set of seed nodes (or not choosing at all) for network analysis purpose, we can start from a carefully curated set of nodes and search from these special seeds nodes carefully according to some heuristic that takes advantage of the domain-specific properties on edges and nodes.

In the address graph $G_{\mathcal{A}}$, given a set of address $\vec{a}$ (*seed nodes*) and a exploration heuristic H, we use the following iterative algorithm to find a $G^*_{\mathcal{A}}$ sub-graph of $G_{\mathcal{A}}$ such that $G^*_{\mathcal{A}}$ is much smaller than $G_{\mathcal{A}}$, and running community detection etc. *global* graph algorithm, for example, louvain algorithm, on $G^*_{\mathcal{A}}$ will output return result as running on $G_{\mathcal{A}}$. A sketch of this algorithm in described below:

```
def iterative_search(seed_nodes, H, p):
  S = seed_nodes
  Q = [ (s, 0) for s in S] # element and depth.
  while Q.length:
    a, depth = Q.pop()
    txns = find_in_tx(a) + find_out_tx(a)
    for tx in H(a, txns, p):
      for b in find_tx_neighbors(tx)
        if b not in S and depth < MAX_DEPTH:
          Q.add((b, depth + 1))
          S.add(b)
  return S
```

Probabilistic heuristic H considers three types of information for a given address a and related transaction t: (1) Class of a (from ground-truth or prediction) (2) The net value flowed in/out a during t (3) Timestamp of t. Also, after selecting K out of M according to the priority rules, it applies a probability p to select $p(M - K)$ out of the unselected ones, in order to avoid missing important nodes that is not apparent in the near term.

4.4 Addresses Clustering & De-anonymization

By analyzing the pattern in transaction graph (section 3), we find that it is very intuitive for human to guess that two (service) nodes belong to the same entity by looking at the their shared users etc. Thus, we propose the following metrics for measuring similarity between two nodes in $G_{\mathcal{A}}$ for clustering

purpose. Given a set of service addresses $\vec{a}$, we want to cluster them into different real entities. We have two ways to do this: (1) take advantage of domain-specific knowledge and prior knowledge on the data structure and design distance/similarity metrics ourselves (we define one metric we used here). (2) use off-the-shelf clustering/community algorithms.

Definition 4.1. Degree of shared users: For all nodes classified as *service nodes* using features in the previous resolution layer, we propose to count all neighboring connection individual-class nodes as this service node's user group Users(s). The degree of shared users for two service nodes s_1, s_2 is defined using Jaccard similarity

$$J(\text{Users}(s_1), \text{Users}(s_2)) \triangleq \frac{|\text{Users}(s_1) \cap \text{Users}(s_2)|}{|\text{Users}(s_1) \cup \text{Users}(s_2|}$$

5 Implementation

In total, our implementation is composed of 1235 lines of pure Python source code and 1344 lines of IPython notebook code, not including custom utility library we used. We also write 63 lines of queries in standard SQL and neo4j Cypher language. Our implementation of the system (Figure 1) has the following components:

Crawlers and RESTful API wrappers We implement our crawlers to get data from blockchain.com/tags, walletexplorer.com, and btc.com. They are written in `scrapy` library or bare-metal `requests` library. They are all implemented in Python.

Bulk Pre-processing using Spark and BigQuery We use Spark-RDD/Spark-SQL APIs in Python extensively in pre-processing our data. We also use BigQuery's standard SQL interface in pre-processing part of the data. We find the SQL interface to Spark is more readable and easier than RDD to use in many cases, while in some cases, RDD interface is sufficient, or necessary for nested structure or Python-native object manipulation.

Feature Extraction using Neo4j and Spark Neo4j is too slow for large-scale analytics since it is hard to predict time need until completion. Also, neo4j is not scaling *out* to multiple cores and multiple instances well since it needs to enforce the transactional semantics. On the other hand, Spark is faster and more measurable when generating features. But writing Spark needs a lot of pipeline planning and is harder to debug.

Machine Learning Using Python We use Scikit-learn library for many machine learning based inference (for example, the classifier based on decision tree model in subsection 6.1, the clustering using `SpectralClustering` etc.)

Network Analysis, Graphing and Visualization using Matplotlib, Seaborn, Neo4j and Gephi We use NetworkX in computing communities and graph properties. We have two types of visualizations: (1) micro-level, focusing on the interactions between a few nods in the full transactions graph G. (2) macro-level, focusing on the entire address graph $G_{\mathcal{A}}$. We use the Yifan Hu [9] and Force Atlas2 [10] layout algorithms and modularity based partitioning algorithm [11] in Gephi v0.9.2.

6 Evaluation

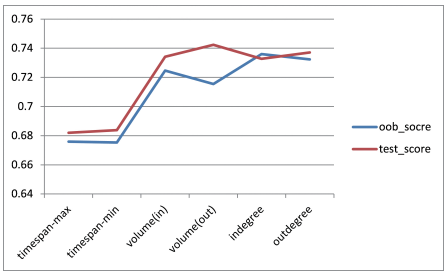
6.1 Classification

Experimental Setup We use "class" column in off-chain data as the ground-truth.

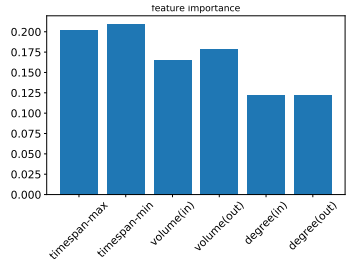
We propose *time-traveling* strategy to reuse ground-truth labels and generate more training data to prevent over-fitting. *Time-traveling* strategy means we choose a series of history snapshots $t_{k_1}, \ldots, t_{k_n}$, and then take the data all between the earliest time and t_{k_i} to compute that features aggregated at that point in the history. We find that most transactions happened between 2013.12-2015.12. Based on our observation of transaction activities, we decided to use 3 months as an interval to obtain 8 datasets, since there are more variance in snapshot data.

	Exchange	Faucet	Gambling	HYIP	Market	Mixer	Other	Pool	Service
precision	0.78	0.85	0.76	0.73	0.70	0.84	0.62	0.87	0.77
recall	0.89	0.62	0.69	0.43	0.59	0.76	0.44	0.76	0.71
F-1 score	0.83	0.72	0.72	0.54	0.64	0.80	0.51	0.81	0.74

Table 1: Precision, recall, and F1-score



(a) Ablation-study: we remove one feature and study if it will have impacts on final result. The experiment is conducted on combined snapshots data and we find that 'timespan-min' and 'timespan-max' has the largest impacts on model accuracy which decreases from average level by 5 percent.



(b) Feature importance from decision tree model: We use the dataset without historic snapshots. It is clear that 'timespan-min' and 'timespan-max' are the most important factors while 'in-deg' and 'out-deg' are the lowest ones.

We combine both current and historic data produced by time-traveling. The size of the final dataset used in evaluation has 50657 rows with 6 dimension. The distribution of labels is: Exchange 41%, Gambling 27%, Market 9%, Pool 7%, HYIP 7%, Service 3%, Mixer 3% and others 2%. Finally, we use 10-Fold cross validation after random shuffling for evaluating the classifier.

Features Our classifier consists of the following set of features: 'lifetime transaction time-span (min)', 'lifetime transaction time-span (max)', 'lifetime degree (inputs)', 'lifetime degree (outputs)', 'lifetime volume (inputs)', and 'lifetime volume (outputs)'. The total number of features $n_{\text{features}} is 6$.

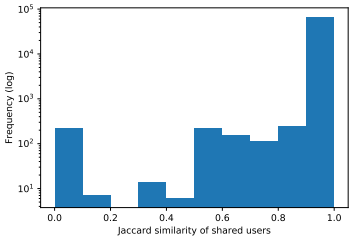
Model We use `RandomForestClassifier` from Scikit-learn python library and choose the following hyper-parameters with grid-search: $n_{\text{estimators}} = 100$ and $\max_{\text{features}} = \sqrt{n_{\text{features}}}$. Training and testing take 17.1157 s in total.

Results Table 1 shows other metrics such as precision, recall and F-1 score for its concatenated results. Average accuracy is 0.738 .

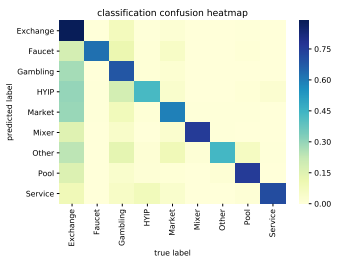
Error Analysis We looked into the classification results versus ground-truth and find out that there are two major classes with mis-classification: 'Exchange' and 'Gambling'. 'Gambling' addresses contribute to 38.07% of all the mis-classified addresses and 'Exchange' addresses contribute to 24.81% of all the mis-classified addresses. Also, 69.31% mis-classified 'Exchange' addresses were classified as 'Gambling' and 88.31% mis-classified 'Gambling' addresses were classified as 'Exchange'. This mis-classification is also showed in confusion matrix Figure 6b.

First of all, the unbalance in data might be one cause of this error. From subsection 6.1, we know that 'Exchange' and 'Gambling' have the largest advantage over others in the ground-truth data.

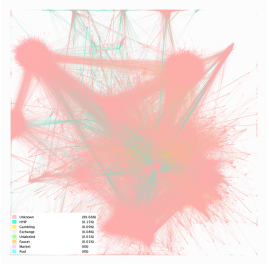
The further investigation on feature importance can give more possible reasons. Figure 5b shows that two time-related features ('lifetime transaction time-span (min)' and 'lifetime transaction time-span



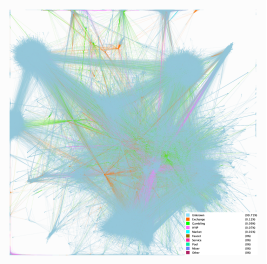
(a) Shared users metric



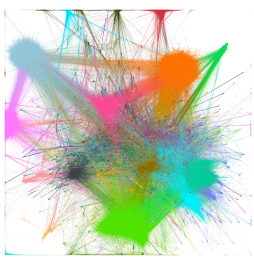
(b) KFold test data confusion matrix



(a) Color address nodes by ground-truth class. HYIP is the second majority besides unknown.



(b) Color address nodes by predicted class. Some HYIP addresses are classified as Exchange and Gambling.



(c) This network colored by modularity gives more information than the previous one. There are six major clusters in this figure.

Figure 7: Iter 2

name	edges	nodes	density
iter #0	114841	175576	0.000007
iter #1	726484	183319	0.000043
iter #2	1727496	501619	0.000014

(a) Address graph statistics

Method	AR	AMI	V
SharedUsers	0.0003	0.0013	0.7484
Contraction	0.0013	0.0041	0.7463
Baseline	0.0000	-0.0000	0.7664
K-Clique	0.0300	0.2394	0.5229

(b) Clustering results

(max)') have the largest importance weights, indicating the forest splitting strategy is mainly based on time period. However, the ground-truth is collected from a fixed time interval (2013-2016) when all the addresses have high transaction frequencies and there is high possibility that addresses existing at the same time interval are mis-classified to the same group.

After plotting the distributions of important features for addresses that should be 'Gambling' but classified as 'Exchange' and vice versa, we notice that most of the features, especially for the important ones such as 'timespan-max' and 'timespan-min', tend to look similar for the two worst-performed classes.

6.2 Iterative Graph Search

We run the iterative graph search with depth $K = 3$ using addresses with service labels as the query. The statistics of resulted sub-graphs is in Figure 8a.

We also use Gephi to plot iteration 2's graphs (Figure 7). We color them using ground-truth labels, predicted labels, and labels from modularity based partitioning as a reference.

6.3 Clustering

Experimental Setup We compared four methods in this evaluation: (1) **Baseline** that assigned nodes to different groups. (2) **Contraction** that use two simple, deterministic inference rules for de-anonymization. This is used widely in previous work like [12][6]. We prepared 30,003 addresses which are contracted to 27,526 groups. (3) **SharedUser** is clustering based on domain-specific network structures. (3) **K-Clique** is a general clustering algorithm.

We use the "name" field of our off-chain data as the ground-truth: if two addresses has the same "name" , then they are considered to be of same entity.

Results We use three type of metrics to evaluate clustering results: "V-measure" [13] (**V**), "Adjusted Mutual Information" [14]**AMI**, and "Rand index adjusted for chance" [15]**AR**. We get the following scores in Figure 8b. As we can see, all three methods are better than baseline. Contraction and K-Clique have better result than SharedUsers.

Error Analysis First, our iterative search heuristics is not good enough which might contribute to the erroneous cases. Also, compared to full transaction network, the address network is simplified and sparse. For SharedUsers method, it may lead to incomplete search results because of information loss. An intuitive visualization of these problem is from the classification error analysis, there are

some address node belonging to the same class that have high centrality but are unconnected to each other.

7 Related work

Classification for Bitcoin Addresses Lin et al. [4] proposes using summarized transactions as features for classification and abnormity detection of Bitcoin network addresses. Based on previous work [2], the authors added new statistics including time series, detailed transaction summaries and distribution data of transactions. Also, this work adds more variance to data and boosts model performance. The class imbalance and data scarcity were mentioned by authors. However, solely by focusing on address-based schema evaluations, authors didn't really solve this problem. More investigations can be done using entity-based schema.

Anomaly Detection There are two reports from Stanford machine learning and network analysis course in anomaly detection, while attacking different aspects, they were written by same authors and derived from a single project. The first paper [5] uses network analysis techniques including the laws of power degree & densification, and local outlier factor (LOF) method to detecting anomalies and transaction graph (transaction as node and UTXO flow as edge). The second paper [16] introduced and compared three unsupervised learning techniques, like SVM, k-mean clustering etc., for the same purpose. Their scalability can be improved: since original SVM method takes a long time to run, thus authors limited data set to 100000 data points for all methods. The use of K-means as a baseline might be unsound, however, no visualization, explanation or proof is given. Also, there is no plot of ground-truth distribution.

Bitcoin Network Analysis Paper [17] reviews the history of Bitcoin and de-anonymization. It first introduces concepts such as power-law distribution, shortest path length, and centrality (corresponding equations are provided in each section). Results from network analysis are combined with the business categories and geographic information for more insights. Authors concluded that the Bitcoin network follows a power-law distribution over large parts of the value range and it can be considered as a scale-free network since 2010. Also, major hubs in the Bitcoin network are identified when the degree centrality is analyzed. Finally, this work also measures the average clustering coefficient on the business and country aggregation level.

8 Future work and More

Limitations Current system is an offline, batch-style analysis system. It doesn't support streaming updates from receiving new blocks in real time. However, since the incoming traffic type and rate of blockchain is very stable and predictable, supporting efficient OLAP should not be difficult. Also, we fail to apply the feature of average transaction time interval into our algorithms due to time limitation. We may consider to explore it in the future work.

Future Work (1) Evaluation on other types of dataset, like Ethereum currency. (2) Application of our clustering results on anomaly detection etc. (3) Explore more features like transaction types (4) Use more ground-truth like `is_miner` [18] (5) Explore the similarity and difference between the class of 'Gambling' and 'Exchange' to improve performance. (6) Explore and apply average transaction time interval into the algorithms.

Conclusion Crypto-currency & Blockchain Data Mining (DM) is still a poorly explored topic, compared to other areas in DM. We believe that part of the reason is that crypto-currency is designed to hide identities, thus any relevant metadata is hard to infer automatically (like our efforts) and is not accessible to public.

Acknowledgment Thanks to Aleš Janda (author of WalletExplorer [6]) and Michele Spagnuolo (author of Bitiodine [19]) for answering our inquiries. Thanks to UW CSE and Google Cloud for providing computation support.

Great thanks to the instructor Tim Althoff, our project TA Swati Padmanabhan, and other TAs of CSE 573 (19sp) for their guidance, feedback and hard work!

References

[1] Michael Fleder, Michael S Kester, and Sudeep Pillai. Bitcoin transaction graph analysis. *arXiv preprint arXiv:1502.01657*, 2015.

[2] Kentaroh Toyoda, Tomoaki Othsuki, and P Takis Mathiopoulos. Multi class bitcoin-enabled service identification based on transaction history summarization. 2018.

[3] Dániel Kondor, Márton Pósfai, István Csabai, and Gábor Vattay. Do the rich get richer? an empirical analysis of the bitcoin transaction network. *PloS one*, 9(2):e86197, 2014.

[4] Yu-Jing Lin, Po-Wei Wu, Cheng-Han Hsu, I Tu, Shih-wei Liao, et al. An evaluation of bitcoin address classification based on transaction history summarization. *arXiv preprint arXiv:1903.07994*, 2019.

[5] Thai Pham and Steven Lee. Anomaly detection in the bitcoin system-a network perspective. *arXiv preprint arXiv:1611.03942*, 2016.

[6] WalletExplorer.com: smart Bitcoin block explorer. `https://www.walletexplorer.com/`, 2019. [Online; accessed 21-May-2019].

[7] Blockchain.com Address Tags: Label your public bitcoin addresses. . `https://www.blockchain.com/btc/tags`, 2019. [Online; accessed 21-May-2019].

[8] A 200-Year-Old Idea Offers a New Way to Trace Stolen Bitcoins. `https://www.wired.com/story/bitcoin-blockchain-fifo-dirty-coins/`, 2018. [Online; accessed 21-May-2019].

[9] Yifan Hu. Efficient, high-quality force-directed graph drawing. *Mathematica Journal*, 10(1):37–71, 2005.

[10] Mathieu Jacomy, Tommaso Venturini, Sebastien Heymann, and Mathieu Bastian. Forceatlas2, a continuous graph layout algorithm for handy network visualization designed for the gephi software. *PloS one*, 9(6):e98679, 2014.

[11] Renaud Lambiotte, J-C Delvenne, and Mauricio Barahona. Laplacian dynamics and multiscale modular structure in networks. *arXiv preprint arXiv:0812.1770*, 2008.

[12] Dániel Kondor, István Csabai, János Szüle, Márton Pósfai, and Gábor Vattay. Inferring the interplay between network structure and market effects in bitcoin. *New Journal of Physics*, 16(12):125003, 2014.

[13] Andrew Rosenberg and Julia Hirschberg. V-measure: A conditional entropy-based external cluster evaluation measure. In *Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL)*, 2007.

[14] Nguyen Xuan Vinh, Julien Epps, and James Bailey. Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. *Journal of Machine Learning Research*, 11(Oct):2837–2854, 2010.

[15] Lawrence Hubert and Phipps Arabie. Comparing partitions. *Journal of Classification*, 2(1):193–218, Dec 1985.

[16] Thai Pham and Steven Lee. Anomaly detection in bitcoin network using unsupervised learning methods. *arXiv preprint arXiv:1611.03941*, 2016.

[17] Matthias Lischke and Benjamin Fabian. Analyzing the bitcoin network: The first four years. *Future Internet*, 8(1):7, 2016.

[18] Bitcoin Mining Pool Classifier - Data from BigQuery. `https://www.kaggle.com/wprice/bitcoin-mining-pool-classifier`, 2019. [Online; accessed 21-May-2019].

[19] Michele Spagnuolo, Federico Maggi, and Stefano Zanero. Bitiodine: Extracting intelligence from the bitcoin network. In *International Conference on Financial Cryptography and Data Security*, pages 457–468. Springer, 2014.