Unsupervised Approaches to Detecting Anomalous Behavior in the Bitcoin Transaction Network

Jason Hirshman
Stanford University
Department of Mathematics
Stanford, CA, USA
hirshman@stanford.edu

Yifei Huang
Stanford University
Department of Computer Science
Stanford, CA, USA
yifei@stanford.edu

Stephen Macke
Stanford University
Department of Computer Science
Stanford, CA, USA
smacke@stanford.edu

I. Introduction

Bitcoin is an electronic crypto-currency created in 2008 by Satoshi Nakamoto (pseudonym). At the time the original bitcoin client was written, the idea of a purely peer-to-peer (P2P) digital currency which did not require a trusted-third-party to confirm transactions / prevent double spending was unique. In the bitcoin network, all transactions are public, effectively rendering double-spending impossible. A criminal who wishes to double-spend or falsify some segment of the transaction history must convince the majority of the bitcoin network that his transaction history is correct, but in order to do that, he must provide the appropriate proof of work. Under the assumption that the majority of the network is honest, the criminal would have to have more computational power than the majority of the network in order to falsify the transaction history, as described in [1]. Since the onset of bitcoin, several other crypto-currencies have sprung into existence, but bitcoin continues to be the most popular.

Because transactions in the bitcoin network are specified by the public keys of the payer and payee, some level of anonymity is guaranteed provided public keys are not traceable to real-world identities. For criminal organizations and others using bitcoin which require strong anonymity, this is not enough, so a so-called "mixing service" is employed. The mixing service takes in bitcoins from a group of individuals requiring strong anonymity, sends the coins around randomly in an attempt to obfuscate their origins, and then sends similar amounts of bitcoins back to new addresses specified by the individuals using the service. This is discussed in more detail in [2].

For our CS229 project, we were interested in using machine learning techniques to explore a dataset of bitcoin transactions; in particular, we were interested in exploring the anonymity guarantees of the bitcoin network. The questions we were hoping to answer are:

1) Can we cluster the dataset in order to make exploration easier?
2) Can we detect attempts at money laundering / mixing services?
3) Can we trace the outputs of a mixing service back to its inputs?

While tracing mixing service outputs to the corresponding inputs eluded us due to the complexity of mixing services, we do believe we made some headway on their detection. In the subsequent sections, we describe the dataset we are using and the initial preprocessing performed upon it. We then describe an initial attempt to explore the dataset by clustering **hubs** (our term for users with high numbers of transactions) based on a particular feature set, first using the K-means algorithm then by applying an unsupervised learning algorithm, "RolX", which assigns our users to various roles. Finally, we note some interesting, anomalous behavior that we were able to discover thanks to our unsupervised restructuring of the dataset.

II. Data

The bitcoin dataset was obtained from [3], generously made available by Ivan Brugere. It contains information on all transactions up through May of 2013, and provides a nice relational structure. The relational schema is detailed in figure 1.

A. Resolving Public Keys into Users

The dataset uses techniques from [4] in order to perform an initial clustering of public keys into the coarser notion of a "user." Briefly, two public keys may be assumed to belong to the same entity if they both appear as inputs to a particular transaction, since this means that whoever authorized the transaction had access to both corresponding private keys. Thus, the "users" as given in the dataset are probably not totally accurate; in reality some of them probably need to be merged.

B. Granularity of the Data

The dataset indicates transactions at a user-to-user level, but not at an "address-to-address" level; that is, it does not specify, for a particular transaction, precisely which keys were involved, or how many bitcoins moved to particular keys. We are interested in this information for future work with this dataset; particularly, this finer-grain detail is of interest for trying to detect mixing, and also for trying to further resolve public keys into "users." Fortunately, even though this finer-level detail is not explicitly stored, the transaction keys and public keys associated with each transaction are specified, and the finer detail may thus be queried from the bitcoin blockchain.

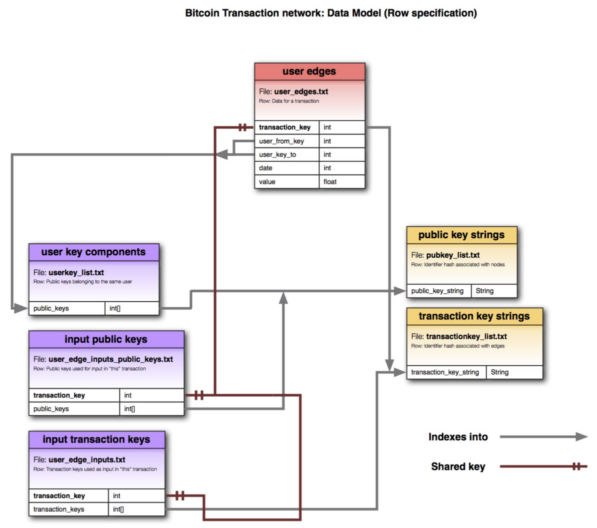

Bitcoin Transaction network: Data Model (Row specification)
user edges
File: user_edges.txt
Row: Data for a transaction
transaction_key int
user_from_key int
user_key_to int
date int
value float
user key components
File: userkey_list.txt
Row: Public keys belonging to the same user
public_keys int[]
input public keys
File: user_edge_inputs_public_keys.txt
Row: Public keys used for input in "this" transaction
transaction_key int
public_keys int[]
input transaction keys
File: user_edge_inputs.txt
Row: Transaction keys used as input in "this" transaction
transaction_key int
transaction_keys int[]
public key strings
File: pubkey_list.txt
Row: Identifier hash associated with nodes
public_key_string String
transaction key strings
File: transactionkey_list.txt
Row: Identifier hash associated with edges
transaction_key_string String
Indexes into
Shared key

Figure 1: Relational diagram for bitcoin data.[3]

III. K-Means

A. Summary of Method

K-means is a heuristic method used to partially solve the NP-hard problem of clustering some number of observations into K clusters, where the sum of the Euclidean distances between each observation to its centroid is minimized. [5] To better understand our dataset, we used K-means to cluster the users into groups with similar features. Since the data contains over 6.3 million "users" (defined as a set of public keys), we selected the high-transaction-traffic users (hubs, as defined previously) for preliminary analysis. This allowed us to look at a more interesting subset of the data without requiring enormous initial computational power. The subset included users with 650 transactions or more, resulting in a subgraph of 6,058 hubs.

For each hub, features were selected based on the properties of the hub and its children, or other users with whom the hub has interacted with. The following 15 features were selected:

- The hub's degree (total # of transactions), in-degree (total # of receiving transactions), and out-degree (total # of sending transactions).
- The mean and variance of the childrens' total degree, in-degree, and out-degree
- The mean and variance of all the transactions amounts, incoming transactions amounts, and outgoing transactions amounts

These correspond with so-called "egonet" features as described in [6].

We normalized or "whitened" the feature vectors so that all features were weighted equally. Because the K-means results are sensitive to the initialization of the centroids, the centroids were selected ramdomly from a Gaussian distribution with mean and covariance matrix estimated from the data provided. K-means was run with 1000 iterations for various number of clusters with different random initializations. To determine the optimal number of clusters, we decreased the number of clusters until the drop in "error", or the sum of the distances from each point to its corresponding centroid, was not as significant. Through this method, we determined that the optimal number of clusters is 5, although, as we discuss shortly, two of these clusters were necessary for only a few outlier observations.

B. Clusters

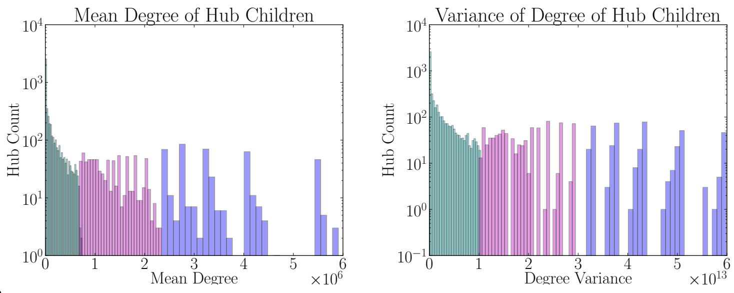


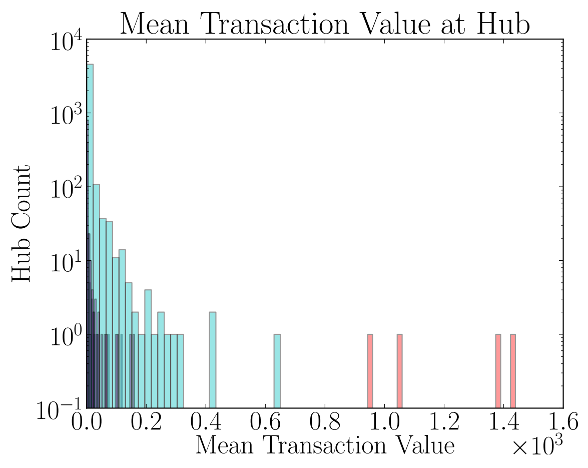
Figure 2: Features corresponding to non-outlier clusters. Each graph is for a particular feature, and each cluster corresponds to a particular color. Note: The darker turqoise in these graphs actually corresponds with the lighter turqoise in the graphs of figure 3; the reason it appears darker in this set is because the histograms are denser. Lastly, the pink outlier clusters do not appear in this set because they are grouped very close to the "0" mark.

After clustering using $k = 5$, the vast, vast majority of hubs fell into three categories, corresponding to the three colors in figure 2. The features which were most important in determining these clusters were:

1) Mean node degree of other users transacted with.
2) Variance of node degree of other users transacted with.

Unfortunately, this does not tell us very much apart from the fact that some hubs have transacted with users with more transactions than others, and that the users with more transactions are spread over a wider range in terms of number of transactions.

The interesting bits of our initial results come from the other two clusters – one of which contains 5 hubs, and



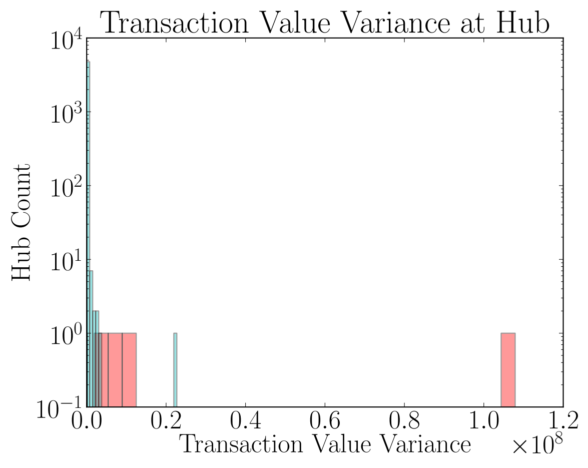


Figure 3: Frequencies at which feature values appear for those features that contributed to the outlier clusters, both colored pink. Note that certain clusters do not appear because they are so close to the "0" mark. The turquoise, though not an outlier cluster, is not so close to the "0" mark to completely disappear.

the other which contains a single hub. As seen in figure 3, the features which segmented out these clusters (both colored pink, for the sake of simplicity) corresponded to mean transaction amount and transaction variance of the hub. For whatever reason, these hubs had enormous variance in transaction amount when compared with the other hubs – they tended to be both "big spenders/receivers" and "inconsistent spenders/receivers." This in and of itself is a simple type of anomaly since it means that these hubs were engaged in transactions which were worth very little, but also in transactions worth many thousands of bitcoins.

IV. RolX

A. Summary of Method

To validate and expand upon the results of the K-means clustering, we attempted to assign roles to the different hubs based on the same features. [7] proposes an unsupervised learning algorithm called *RolX* (Role eXtraction) which seeks to classify nodes of a graph into various classes termed "roles." These roles contain nodes with similar structural features in the sense that they have the same connectivity structure within the graph. For example, one role might contain those nodes that are part of highly-connected subgraphs or alternatively those nodes that have a disproportionate number of incoming edges. RolX performs role discovery by factoring the feature matrix into two nonnegative matrices. Given n nodes (or in our case, hubs) and f features, construct a matrix $V_{n\times f}$ and factorize it into nonnegative matrices $G_{n\times r}$ and $F_{r\times f}$ where r is a chosen low rank. In other words, find:

$$\underset{G,F}{\operatorname{argmin}} \|V - GF\|, G, F \geq 0, G \in \mathbb{R}^{n\times r}, F \in \mathbb{R}^{r\times f}$$

where $\|A\| = \sqrt{\sum_{i,j} a_{ij}^2}$ is the Frobenius norm. This factorization reduces the dimensionality of the data and provides a representation of the feature matrix as linear combinations of the r roles.

The factors are computed iteratively by applying the multiplicative updates, given in [8], as follows:

$$F_{bj}^{(t+1)} := F_{bj}^{(t)} \frac{[(G^{(t)})^T V]_{bj}}{[(G^{(t)})^T G^{(t)} F^{(t)}]_{bj}}$$

$$G_{ia}^{(t+1)} := G_{ia}^{(t)} \frac{[V(F^{(t+1)})^T]_{ia}}{[G^{(t)} F^{(t+1)} (F^{(t+1)})^T]_{ia}}$$

for every j in the range $1 \dots f$, i in the range $1 \dots n$, and a and b in the range $1 \dots r$.

However, before factorization, RolX first provides a method for choosing an appropriate value for r. The algorithm calls for minimizing the model description length which is defined to be the sum of the cost of representing the model in memory and the cost of correcting the errors. This minimization is done by computing the matrices G, F for various values of r.

Unfortunately, we were unable to follow this minimization procedure, as it calls for using KL divergence to measure the cost of error correction which is not feasible given the presence of zero-valued features. KL divergence was chosen since [7] found that the model errors were not normally distributed, but we believe that the Frobenius norm can still be applied to choose the parameter for our purposes since we are not as concerned with whether two roles may be merged or separated unnecessarily, as we are instead trying to find the outlier roles with very few members.

Prior to factorizing the feature matrix for our data, each column was normalized by dividing the column by its mean (a column corresponds to a single feature). In this way, the non-negativity of the data was preserved while still ensuring that no one feature dominated the role selection. Figure 4 shows the results of factorizing the feature matrix with r ranging from 2 to 8. It was found that when r was greater than seven, the factor matrices were not of full rank which means that more than seven roles were unable to isolated when optimizing those matrices. $r = 7$ was chosen as it minimized the error, $\|V - Gf\|$.

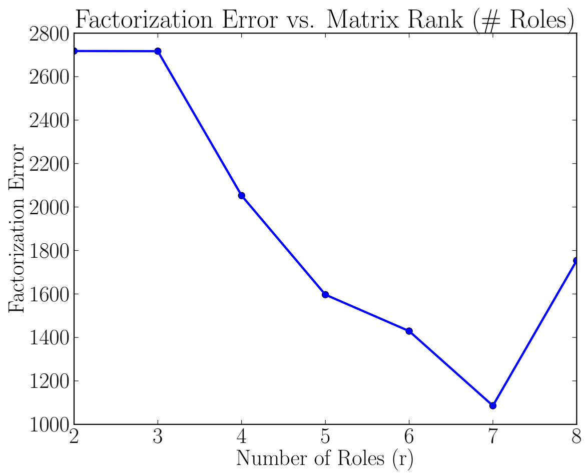


Figure 4: We found the optimal number of roles occurred at $r = 7$.

B. Roles

Upon choosing the target number of roles, the factorization was once again completed, and the hubs were placed into arrays according to their most prominent roles. The most prominent role was chosen by taking the maximum over the hubs row in the G matrix which contains the coefficients for reproducing a hubs feature values by taking a linear combination of the role characteristics in the F matrix. Two roles had between 1500 and 2500 members; three had between 500 and 1000 members; and two had 6 and 15 members. These last two were examined more thoroughly to determine whether there was overlap with the anomalies found through clustering.

The weights in the F matrix reveals that these two roles were distinguished by their heavy weight on the variance of transaction values which also distinguished the anomalous clusters from the more populous clusters. The two roles only differed themselves in that one role most heavily weighted the variance of the incoming transaction values whereas the other most heavily weighted the variance of the outgoing transaction values. Moreover, there was overlap in the membership of these roles with the membership of the small clusters, i.e. the set of hubs in the anomalous clusters was similar to the set of hubs in these two roles.

This overlap between the two algorithms validates our later choice to investigate those users with high variance in transaction values. Intuitively, it makes sense that mixing services would be involved in high variance transactions, as they could service patrons of vastly different wealth.

V. Example Mixing

Before we investigate our outlier users, let us look at an example of mixing in action. In order to get an idea of what to look for, we ran a small number of our own bitcoins through the mixing service `http://app.bitlaundry.com/`. Again, because all bitcoin transactions are public, we were able to trace some of the mixing through the blockchain.

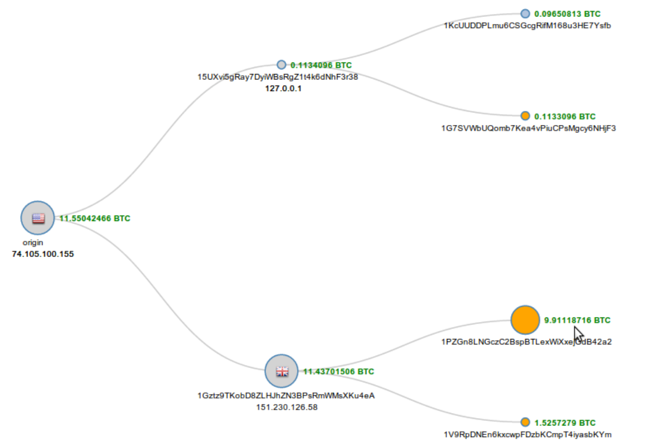

11.55042466 BTC
origin
74.105.100.155
0.1134096 BTC
15UXvi5gRay7DyiWBsRgZ1t4k6dNhF3r38
127.0.0.1
0.09650813 BTC
1KcUUDDPLmu6CSGcgRifM168u3HE7Ysfb
0.1133096 BTC
1G7SVWbUQomb7Kea4vPiuCPsMgcy6NHjF3
11.43701506 BTC
1Gztz9TKobD8ZLHJhZN3BPsRmWMsXKu4eA
151.230.126.58
9.91118716 BTC
1PZGn8LNGczC2BspBTLexWiXxeJGdB42a2
1.5257279 BTC
1V9RpDNEn6kxcwpFDzbKCmpT4iyasbKYm

Figure 5: Example mixing in action. Notice how the transactions appear to recursively split the bitcoins and send to two new addresses.

As you can see, the mixer seems to be forming a complete binary tree of transactions, dividing the coins at each successive address and sending to two new addresses. If we trace further along, we will then find that some of these addresses will then pool their bitcoins together with other coins which were split up in a different series of transactions. This continues for many iterations, making it quite difficult to determine to whom the coins originally belonged by the time they make it to their destinations.

VI. Investigation

Because we do not have transaction details at the level of individual public keys, we are limited by the analysis we can perform with this dataset alone. Fortunately, such fine-tuned granularity is available since, as previously mentioned, the entire blockchain is publically available on sites such as `http://blockchain.info`.

Looking through some of the transactions involving public-keys on the anomalous users, we noticed some interesting behavior. As an example, let us follow some transactions of a particular user, heretofore referred to as "user X." In what follows, we will refer to public keys by the first 4 characters of their SHA-256 hashes. In June 2011 (when bitcoin prices peaked at $17 per coin) we see that user X sent 132 thousand bitcoins from address 1MCZ to addresses 1Cc2 and 1Q4E. 1Cc2 received 82 thousand bitcoins, and 1Q4E received 50 thousand bitcoins. Furthermore, we see that on the same day, 1Cc2 sent 32 thousand bitcoins to 18UF, and 50 thousand bitcoins to 15oG. Next, 15oG sent part of the 50 thousand coins it received to address 1eHh as part of a 424 thousand bitcoin transaction. The remainder went to another address.

Notice that, in the cases of 1MCZ, 1Cc2, and 15oG, they sent all of their bitcoins to other addresses in a single transaction by splitting the money between two receiving addresses. We have only followed a single path along the transaction tree, and the interesting thing is that if we follow other paths, we see this behavior repeated. That is, 1Q4E also sends all of its coins to precisely two addresses, each of which send all of their coins to precisely two addresses, and so on. It is as though user X is recursively splitting the initial 132 thousand bitcoin fortune, as we described in the previous section.

The *really* interesting thing, however, is what happens at the end of some of these splits. In the case of 15oG, it sent part of its coins to 1eHh as part of a 424 thousand bitcoin transaction. This means that other addresses were also involved in this massive transaction. If we trace other transaction paths down, we will see that, for example, money from 1Q4E *also* ends up going to 1eHh as part of the *same* 424 thousand bitcoin transaction. This result is summarized in figure 6.

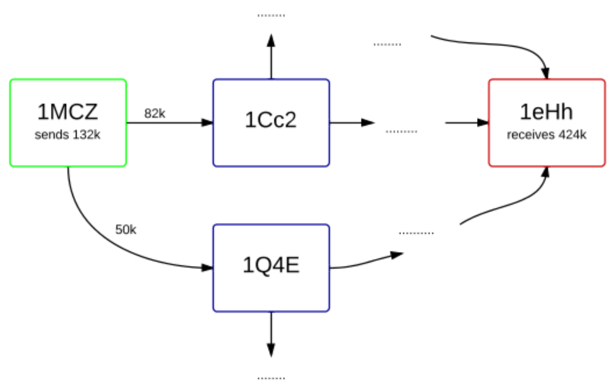


Figure 6: User X splits a large fortune, some of which ends up at 1eHh, from multiple intermediate addresses.

Thus we see that user X appears to be splitting a large sum of money, but then sending portions to the same public key. This seems suspicious. Under normal circumstances, it seems like somebody who wants to send money somewhere all in the same day should have no need to route it to that spot through a bunch of intermediate places. This suggests that we could be looking at a mixing service or some other suspicious service in action.

VII. Conclusion and Future Work

Unsupervised learning techniques revealed anomalies in a large bitcoin transaction network. We were able to identify certain users that conducted transactions in an atypical fashion, one that suggested some sort of money laundering.

Unfortunately, we have no way of proving our suspicions, as we do not have labeled data that points us to cases of these hypothesized mixing services. However, our work here could help pave the way for future clustering techniques, especially by allowing one to choose features that are more revealing of patterns in the data.

The unsupervised learning algorithms we applied, K-means and RolX, ended up achieving our intended ends of locating strange behavior in the network. Through clustering and role detection, we now have a much better idea of what to look for in a suspicious transaction, or, in particular, a string of suspicious transactions. Additional work should be done to both categorize and quantify these anomalies.

References

[1] Nakamoto, Satoshi. Bitcoin: A peer-to-peer electronic cash system. 2008. http://bitcoin.org/bitcoin.pdf

[2] Bitcoin Forum. https://bitcointalk.org/index.php?topic=241.0

[3] Brugere, Ivan. Bitcoin Transaction Network Dataset. http://compbio.cs.uic.edu/data/bitcoin/

[4] Reid, Fergal, and Martin Harrigan. "An analysis of anonymity in the bitcoin system." Security and Privacy in Social Networks. Springer New York, 2013. 197-223.

[5] MacQueen, James. "Some methods for classification and analysis of multivariate observations." Proceedings of the fifth Berkeley symposium on mathematical statistics and probability. Vol. 1. No. 281-297. 1967.

[6] Henderson, Keith, et al. "It's who you know: graph mining using recursive structural features." Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2011.

[7] Henderson, Keith, et al. "RolX: structural role extraction & mining in large graphs." Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, 2012.

[8] Seung, D., and L. Lee. "Algorithms for non-negative matrix factorization." Advances in neural information processing systems 13 (2001): 556-562.