# Security Analysis of Bitcoin Transactions through Statistical Graph Annotation

Pranav A

Big Data Institute, HKUST

**Abstract.** Money transaction using bitcoin is getting prevalent these days due to its government-free, secure, decentralized and open-sourced design. It has been exploited for money laundering. However, it is still possible to detect some trail by analysing the structure of the transaction network. This paper aims to construct a user transaction flow graph of bitcoins and analyze the security and networking properties. By developing basic heuristics, we bundled the public keys of the users from the transaction graph into the user bitcoin flow graph. In a case study of the Silk Road arrest, we demonstrated how coin-mixing trails could be discovered using the user flow graph.

## 1 Introduction

Anonymity and fungibility are the key characteristics of any financial transaction system. Bitcoin architecture aims for these characteristics through transparency and privacy. Bitcoin assigns pseudonyms to the entities. Here, by entities, we mean, users, wallets, and organizations. These entities are represented by the public keys or addresses. It should be noted that information regarding public keys of senders and receivers, and transaction values are made available to the public. However, it is unknown which public keys belong to which entity. A user can be thought of as a mapping of a set of public keys.

The main focus of this project is to discover this mapping and construct a networking graph through those mappings. Generally, this is achieved through clustering. Firstly, we parse the blockchain and construct a transaction graph. From that transaction graph, we apply our multi-input heuristic. This is a fact that all inputs involved in the multi-input transaction represent the same wallet. This heuristic allows us to create a user graph. Through this user graph, we do some statistical analysis to gather more insights on the bitcoin network. Finally, we provide a case study on Silk Road arrest [1]. We demonstrate how a trail could be discovered using graph search algorithms. We have proposed our own algorithms for blockchain parsing and graph creation, which are a lot more memory-efficient. The code is shared on our GitHub repository `https://github.com/pranav-ust/bitcoin`.

## 2 Related Work

This section discusses some pros and cons of the previous work done on multi-input heuristic and transaction graph analysis for the blockchain. In the seminal Bitcoin paper [8], Satoshi introduced the multi-input heuristic, which basically implies that all transaction inputs would correspond to the identical wallet. Fleder et al. [3] showed that using the multi-input heuristic, one could identify most social-able nodes in the bitcoin graphs using PageRank. Also with the usage of multi-input heuristic, Reid et al. [11] analyzes the relation between the topological structure of the transaction graphs and user inflow graph.

Besides, multi-input heuristics, there exists some work whose clustering depends on other heuristics. Ober et al. [10] used entity merging to observe structural patterns in the topological networks of the bitcoin structure. They also analyzed the dormant coins and performed statistical tests to show that they are close to Poisson independent process. Ermilov et al. [2] used blockchain information with off-chain information for clustering. Verified information and tags were used as the off-chain information. This helped in address separation and avoiding errors. Harrigan et al. [4] demonstrated the effectiveness of the address clustering in Bitcoin networks. The reasons include recycling of addresses, slow growth of clusters in the networks, disregarding the merging of clusters and clusters resulting in high degree of centrality. Dorit et al. [12] provided a qualitative analysis of the bitcoin transaction graph. Unfortunately, regarding their title, they did not provide a analysis on "full"graph. Also, no information regarding directions for incomplete edges is given. The construction of graph they proposed is clumsy and requires a lot of parameters.

By analyzing the anonymity risks involved in the bitcoin transactions, Moser et al. [6, 7] provided risk analysis and scoring model for the blockchain based businesses. They also provided an in-depth analysis of money laundering tools using reverse engineering. Nick et al. [9] substantiated experiments on deanonymization which were data-driven. They analyzed some performance on clustering heuristics, or in other words, the addresses which refer to the same wallet. They found a loophole in Connection Bloom Filtering, which was eventually used as a ground truth for verifying the results. They mentioned that even a strongest peer-to-peer network can be deanonymized using a multi-input heuristic. On an average, $68.59$ % addresses can be revealed using these heuristics, in addition to other ones as well. Lischke et al. [5] constructed graphical structures based on business distribution, adoption, and geo-locations. However, their construction of networking graphs relied on only one metric, betweenness centrality.

Clearly, none of the work above mention about memory management and dealing with the nodes of the subgraph. In the upcoming sections, we will walkthrough our designs on such algorithms.

## 3 Blockchain Parsing

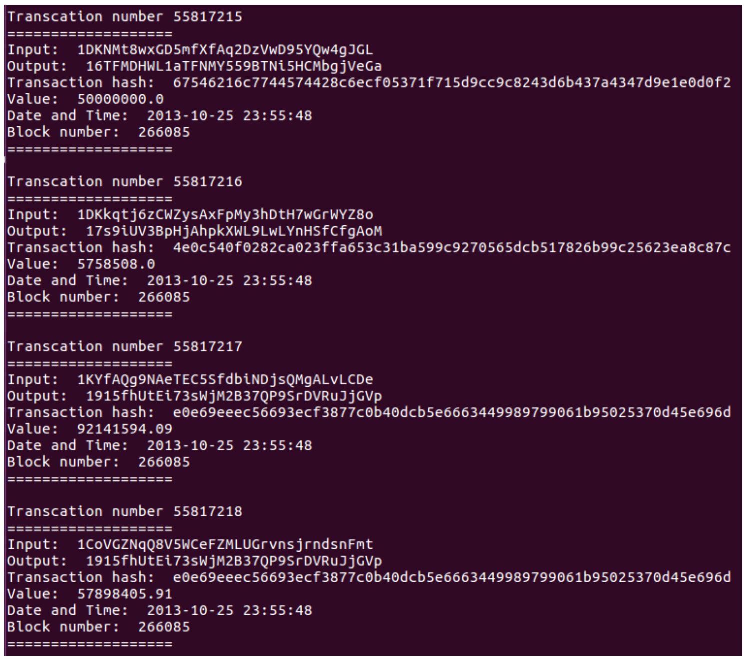
This section discusses on how to parse the data of the blockchain. We also provide our memory management algorithm, which aids in dealing with big data.

### 3.1 Process

Using the peer-to-peer client, we downloaded the bitcoin blockchain from 2009 to October 2013. Then we parsed the blockchain into following components.

1. Transaction input addresses and output addresses
2. Transaction hash
3. Transaction values
4. Transaction datetime
5. Block number

The outputs are shown in figure 1.

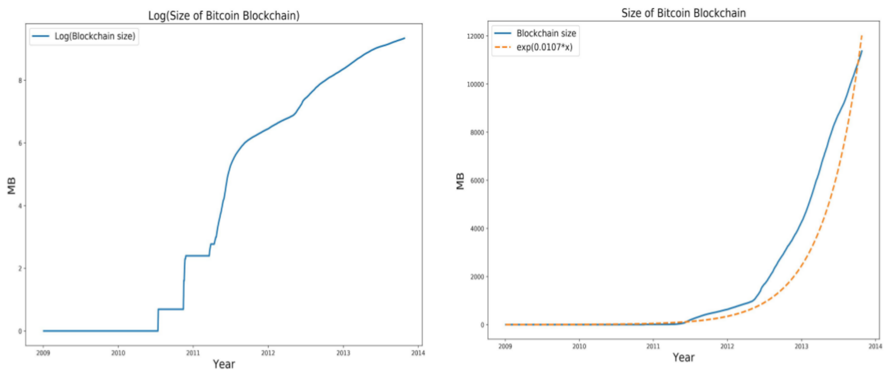


```
Transcation number 55817215
===================
Input:  1DKNMt8wxGD5mfXfAq2DzVwD95YQw4gJGL
Output:  16TFMDHWL1aTFNMY559BTNi5HCMbgjVeGa
Transaction hash:  67546216c7744574428c6ecf05371f715d9cc9c8243d6b437a4347d9e1e0d0f2
Value:  50000000.0
Date and Time:  2013-10-25 23:55:48
Block number:  266085
===================

Transcation number 55817216
===================
Input:  1DKkqtj6zCWZysAxFpMy3hDtH7wGrWYZ8o
Output:  17s9iUV3BpHjAhpkXWL9LwLYnHSfCfgAoM
Transaction hash:  4e0c540f0282ca023ffa653c31ba599c9270565dcb517826b99c25623ea8c87c
Value:  5758508.0
Date and Time:  2013-10-25 23:55:48
Block number:  266085
===================

Transcation number 55817217
===================
Input:  1KYfAQg9NAeTEC5SfdbiNDjsQMgALvLCDe
Output:  1915fhUtEi73sWjM2B37QP9SrDVRuJjGVp
Transaction hash:  e0e69eeec56693ecf3877c0b40dcb5e6663449989799061b95025370d45e696d
Value:  92141594.09
Date and Time:  2013-10-25 23:55:48
Block number:  266085
===================

Transcation number 55817218
===================
Input:  1CoVGZNqQ8V5WCeFZMLUGrvnsjrndsnFmt
Output:  1915fhUtEi73sWjM2B37QP9SrDVRuJjGVp
Transaction hash:  e0e69eeec56693ecf3877c0b40dcb5e6663449989799061b95025370d45e696d
Value:  57898405.91
Date and Time:  2013-10-25 23:55:48
Block number:  266085
===================
```

Fig. 1: Parsed Blockchain

### 3.2 Bottleneck

The blockchain size grows exponentially every year. Figure 2a and 2b show the size evolution of blockchain. A bitcoin transaction contains inputs and outputs. The input contains the hash of the previous transaction and an index of the corresponding output. The output contains the public addresses of receivers and the transaction amount. However, the public addresses of the sender are missing



(a) Log of the blockchain size vs year. (b) Yearwise comparision of Blockchain size

Fig. 2: Graphical plots of blockchain growth

in the current transaction. It's very essential to keep the outputs of the previous transaction.

Hence, we created an algorithm to manage this huge parsed blockchain data, shown in Algorithm 1.

**Algorithm 1** Bitcoin Blockchain Parsing with O(len(blocks)) memory usage

```
Input: Bitcoin transactions $T$, the end time $e_t$, and a empty file $F$.
Let $H(.)$ be a hash table pointing to memory
for each $tx$ in $T$ do
    if $tx.time > e_t$ then
        return
    if $tx$ is a coinbase transaction then
        skip
    $H(tx.hash) \leftarrow tx.outputs$
    $totalvalue \leftarrow$ sum of the values in all valid outputs of tx
    for each $o$ in $tx.outputs$ do
        for each $i$ in $tx.inputs$ do
            sender $\leftarrow$ the address of the sender retrieved from $H(i.hash)[i.index]$
            $recepient \leftarrow o.address$
            Write to F: $sender, recepient, o.value/totalvalue, tx.hash, tx.time$
            Remove $H(i.hash)[i.index]$ from $H(i.hash)$
            if $H(i.hash)$ is empty then
                Remove $H(i.hash)$ from $H$
Output: The parsed data $F$
```

For processing the algorithm 1, we used Azure virtual machine with 2 virtual CPUs and 32 GB memory. The performance is shown in figure 1.

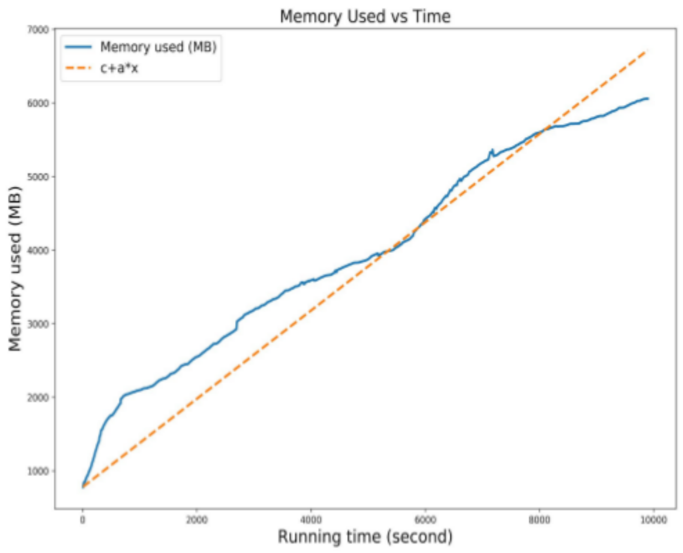
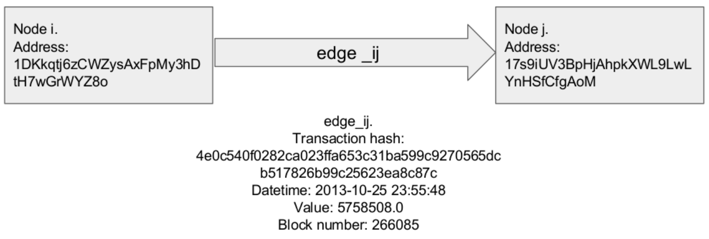


Fig. 3: Performance of Memory Management Algorithm

## 4 Graph Annotation

Once the blockchain data is parsed, we follow-up by creating a transaction graph flow.

Here, each node is a public key appeared in the outputs or inputs of a transaction. Each edge is transaction information including the transaction hash, value, datetime, and block number.



Node i.
Address:
1DKkqtj6zCWZysAxFpMy3hD
tH7wGrWYZ8o

Node j.
Address:
17s9iUV3BpHjAhpkXWL9LwL
YnHSfCfgAoM

edge_ij.
Transaction hash:
4e0c540f0282ca023ffa653c31ba599c9270565dc
b517826b99c25623ea8c87c
Datetime: 2013-10-25 23:55:48
Value: 5758508.0
Block number: 266085

Fig. 4: Data Structure of the Transaction Graph

The data structure of the transaction graph is shown in 4.

Usually a real world entity owns more than one address. We are interested in the transaction activities of the entity It is feasible to link addresses in blockchain based assumption: The input addresses of the same transaction belong to the same real world entity. Our algorithm on linking related public addresses is shown in algorithm 2.

**Algorithm 2** Linking Related Public Addresses

```
Input: Transactions between public addresses $T$
$G \leftarrow$ An empty undirected graph
$X \leftarrow$ The set of the hashes in $T$
for each hash $h$ in $X$ do
    $S \leftarrow$ Find the set of sender addresses in the transactions of hash $h$
    $N \leftarrow$ Size of $S$
    for $i = 0$ to $N - 2$ do
        Add $(S[i], S[i+1])$ as an edge to $G$
$C \leftarrow$ Find the set of all connect components of $G$
Output:
1. Mapping U2A: $i \rightarrow i^{th} set \in C$
2. Mapping A2U: a public address$p \leftarrow$ the index of the connected component in $C$
which contains $p$
```

After linking user addresses, the objective is to construct a user graph which shows flow of transactions from users. The data structure of the figure is shown in 5.

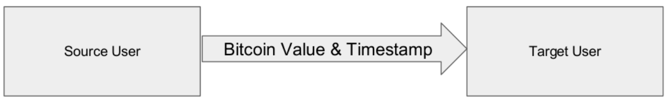


Fig. 5: Data Structure of the User Graph

Initially we constructed a trivial imperfect graph depicting through the public keys. It's usually considered a good practice to generate a new public key for every transaction. Hence, we need some heuristics to bundle these public keys together for a user. We constructed a supplementary graph which analyzes the relation between transactions inputs / outputs and public keys.

### 4.1 Bundling addresses into a user

For the supplementary graph, we represent a vertex with a public key. We constructed an undirected edge between the vertices which are the inputs to the same transaction. Each maximal connected subgraph would correspond to a single user.

In order to discover maximal connected subgraphs, we used Depth-First search to discover such components. Since, one vertex could lie in only one component, we picked seed nodes randomly. From that seed node, we traversed all the connected public keys and bundled them together into one user.

This is a complex process to understand. Hence, we have provdied a series of figures (6, 7, 8) to understand graph annotation.

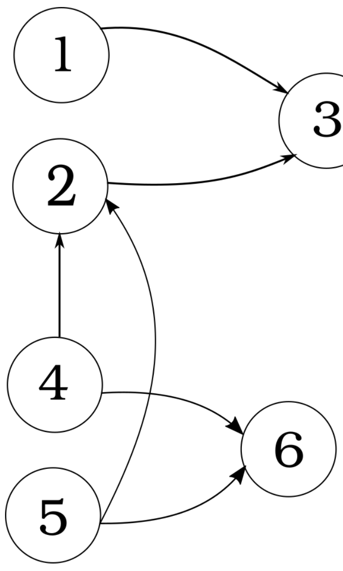


Fig. 6: The figure shows the flow graph of the transaction graph as mentioned earlier. Note that this is only a subgraph from the larger network.

Our graph will concur some missing and corrupted data due to our heuristic assumptions and partial blockchain analysis. Hence, we remove the outliers in following ways:

1. Remove the transactions where the algorithm was not able to bundle the public keys into users.
2. Remove the isolated components and self-loops.

## 4.2 Analysis on the user graphs

We draw a log-log plot of degree distribution which is shown 10a. It shows that the graph slightly deviates from power law.

We applied goodness of fit to our curve and found the p-value to be greater than threshold. This proves that bitcoin transactions do not follow preferential

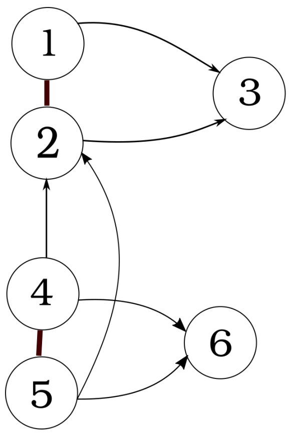


Fig. 7: We bundled 1 and 2 together because they were the inputs for the transaction to 3. Hence, we join 1 and 2 by an undirected edge. Similarly it goes for 4 and 5.

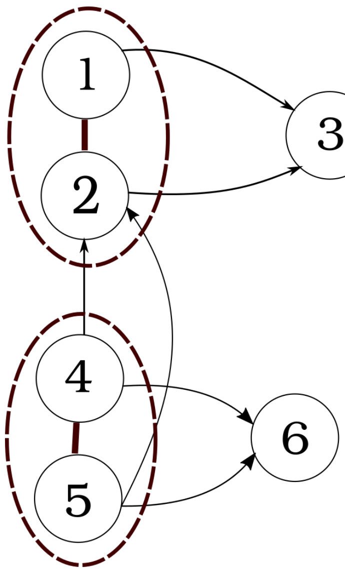
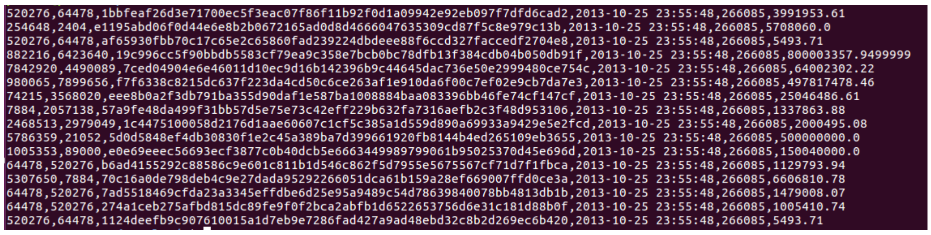


Fig. 8: Now, we only analyze the undirected edges. We extract the maximal connected component and annotate that component as a user.

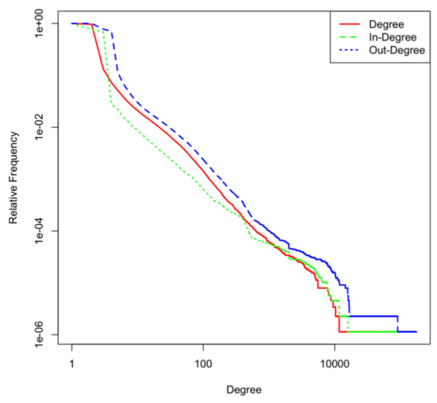
attachment and it is not a scale-free network. This is mainly due to decentralized architecture of Bitcoin.

The plot of component size distribution is shown in figure 10b. Clearly, this does not follow power law as well.

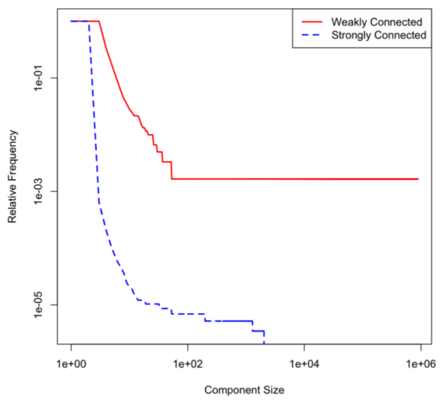


```
520276,64478,1bbfeaf26d3e71700ec5f3eac07f86f11b92f0d1a09942e92eb097f7dfd6cad2,2013-10-25 23:55:48,266085,3991953.61
254648,2404,e1195abd06f0d44e6e8b2b0672165ad0d8d4666047635309cd87f5c8e979c13b,2013-10-25 23:55:48,266085,5708060.0
520276,64478,af65930fbb70c17c65e2c65860fad239224dbdeee88f6ccd327faccedf2704e8,2013-10-25 23:55:48,266085,5493.71
882216,6423640,19c996cc5f90bbdb5583cf79ea9c358e7bcb0bc78dfb13f384cdb04b050db91f,2013-10-25 23:55:48,266085,800003357.9499999
7842920,4490089,7ced04904e6e46011d10ec9d16b142396b9c44645dac736e50e2999480ce754c,2013-10-25 23:55:48,266085,64002302.22
980065,7899656,f7f6338c8215dc637f223da4cd50c6ce263af1e910da6f00c7ef02e9cb7da7e3,2013-10-25 23:55:48,266085,497817478.46
74215,3568020,eee8b0a2f3db791ba355d90daf1e587ba1008884baa083396bb46fe74cf147cf,2013-10-25 23:55:48,266085,25046486.61
7884,2057138,57a9fe48da499f31bb57d5e75e73c42eff229b632fa7316aefb2c3f48d953106,2013-10-25 23:55:48,266085,1337863.88
2468513,2979049,1c4475100058d2176d1aae60607c1cf5c385a1d559d890a69933a9429e5e2fcd,2013-10-25 23:55:48,266085,2000495.08
5786359,21052,5d0d5848ef4db30830f1e2c45a389ba7d399661920fb8144b4ed265109eb3655,2013-10-25 23:55:48,266085,500000000.0
1005353,89000,e0e69eeec56693ecf3877c0b40dcb5e6663449989799061b95025370d45e696d,2013-10-25 23:55:48,266085,150040000.0
64478,520276,b6ad4155292c88586c9e601c811b1d546c862f5d7955e5675567cf71d7f1fbca,2013-10-25 23:55:48,266085,1129793.94
5307650,7884,70c16a0de798deb4c9e27dada95292266051dca61b159a28ef669007ffd0ce3a,2013-10-25 23:55:48,266085,6606810.78
64478,520276,7ad5518469cfda23a3345effdbe6d25e95a9489c54d78639840078bb4813db1b,2013-10-25 23:55:48,266085,1479008.07
64478,520276,274a1ceb275afbd815dc89fe9f0f2bca2abfb1d6522653756d6e31c181d88b0f,2013-10-25 23:55:48,266085,1005410.74
520276,64478,1124deefb9c907610015a1d7eb9e7286fad427a9ad48ebd32c8b2d269ec6b420,2013-10-25 23:55:48,266085,5493.71
```

Fig. 9: The final graph of the trasactions between the users is represented like this.



(a) Degree Distribution of the user graph



(b) Component Size Distribution of the user graph

Fig. 10: Size Distribution of the user graph

We found that most of the nodes with high in-degree and out-degree were from gambling websites like SatoshiDICE.

The visualization of the user graph is shown in figure 11.

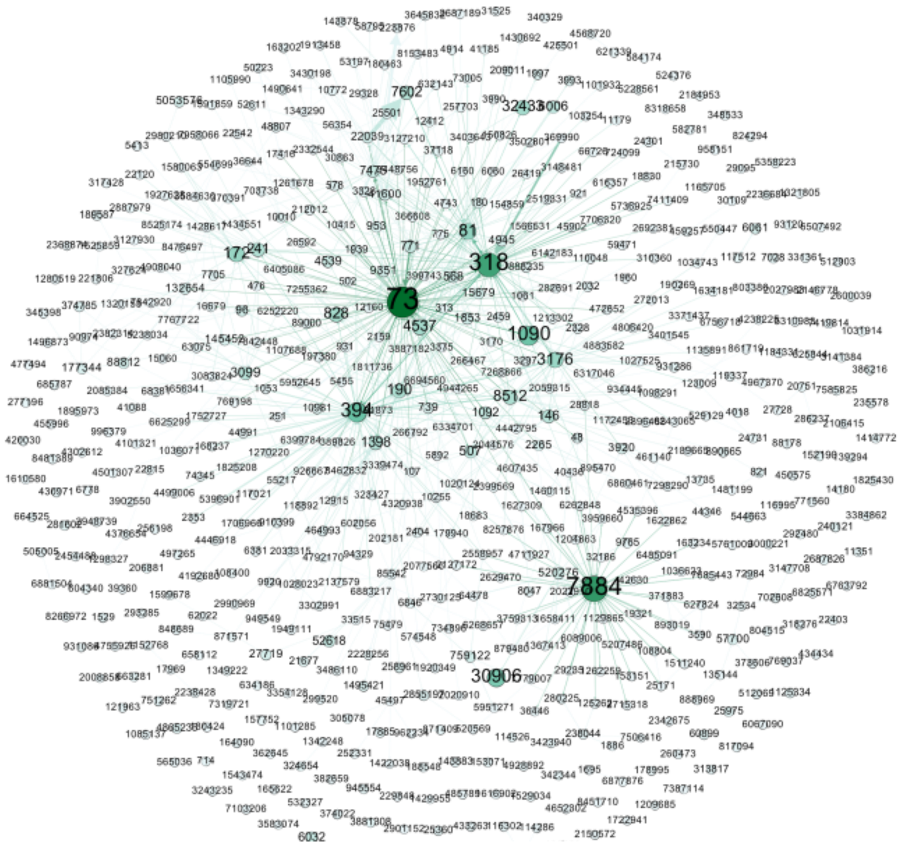


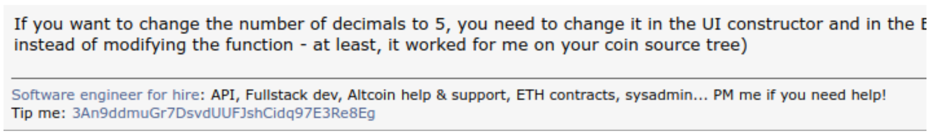
Fig. 11: The one day user graph on 25 October 2013, filtered by the condition degree $\geq 6$. Each node is labelled by the user id and scaled by the pagerank score.

## 5 Preparation for Security Analysis

### 5.1 Web Scraping

In order to link the real world entities with bitcoin public keys, we scraped the bitcoin forum, bitcointalk. Often, users will leave their public keys for donations in the signature of the forums. The example screenshot is shown in figure 12.

We used Beautiful Soup to scrape the pages and build the crawler. The crawler ran around 40 hours. Using breadth-first search we crawled around 5 links deep from the home pages. We parsed the signature section of each post and retrieved the public key through regular expression like $\Sigma\{26, 33\}$ where $\Sigma$ is a set of valid alphanumeric characters. For each regular expression match, we ran the checksum function to validate it as a public key. We finally retrieved around 3000 addresses by crawling around 50000 posts.



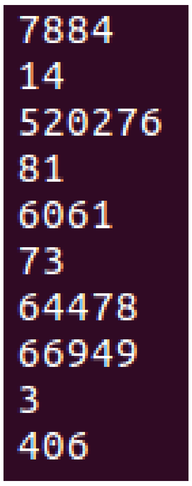
If you want to change the number of decimals to 5, you need to change it in the UI constructor and in the E
instead of modifying the function - at least, it worked for me on your coin source tree)

Software engineer for hire: API, Fullstack dev, Altcoin help & support, ETH contracts, sysadmin... PM me if you need help!
Tip me: 3An9ddmuGr7DsvdUUFJshCidq97E3Re8Eg

Fig. 12: Screenshot of the user leaving their addresses in the forum post

## 5.2 Pagerank

We applied PageRank algorithm to rank the user nodes in the order of importance. This may be used to understand how those forums users are linked with the nodes with high importance. Figure 13 shows top 10 users from pagerank. The user 7884 belongs to SatoshiDICE, a gambling site.



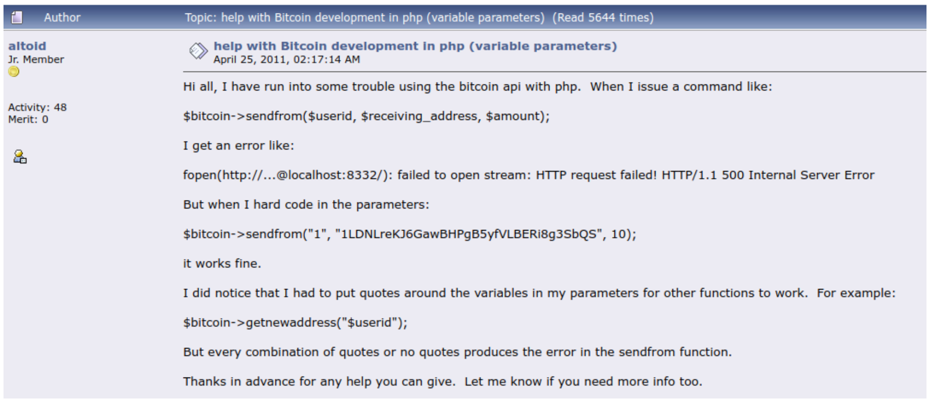
```
7884
14
520276
81
6061
73
64478
66949
3
406
```

Fig. 13: Top 10 users with high pagerank outputs. Most of the top ranked users were found to be gambling.

## 6 Case Study: Fraud Detection of Silk Road Arrest

Posted on April 25, 2011, the user altoid accidentally revealed their address.

We verified the address **“1LDNLreKJ6GawBHPgB5yfVLBERi8g3SbQS”** and found out that it was one of the highly ranked node of the pagerank output for that particular user node. The user was revealed to be **"Dread Pirate Roberts"** who was involved in money laundering for the Silk Road fraud. FBI arrested him in 2013.


Author | Topic: help with Bitcoin development in php (variable parameters) (Read 5644 times)

altoid
Jr. Member

Activity: 48
Merit: 0

**help with Bitcoin development in php (variable parameters)**
April 25, 2011, 02:17:14 AM

Hi all, I have run into some trouble using the bitcoin api with php. When I issue a command like:

```
$bitcoin->sendfrom($userid, $receiving_address, $amount);
```

I get an error like:

```
fopen(http://...@localhost:8332/): failed to open stream: HTTP request failed! HTTP/1.1 500 Internal Server Error
```

But when I hard code in the parameters:

```
$bitcoin->sendfrom("1", "1LDNLreKJ6GawBHPgB5yfVLBERi8g3SbQS", 10);
```

it works fine.

I did notice that I had to put quotes around the variables in my parameters for other functions to work. For example:

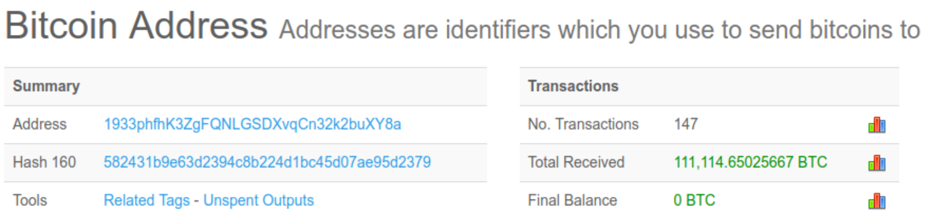
```
$bitcoin->getnewaddress("$userid");
```

But every combination of quotes or no quotes produces the error in the sendfrom function.

Thanks in advance for any help you can give. Let me know if you need more info too.

Fig. 14: Post of the user altoid who accidently revealed their address on a forum post

Using our bitcoin transaction graph, we found an address **"1933phfhK3ZgFQNLGSDXvqCn32k2buXY8a"** which has **111,114.65 BTC** with nothing spent, shown in figure 15.


Bitcoin Address Addresses are identifiers which you use to send bitcoins to

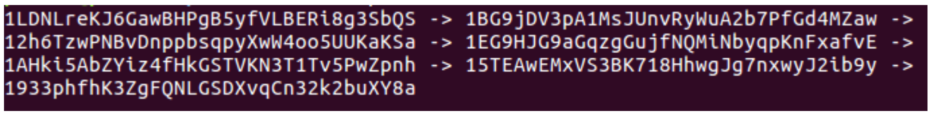
| Summary | |
|---|---|
| Address | 1933phfhK3ZgFQNLGSDXvqCn32k2buXY8a |
| Hash 160 | 582431b9e63d2394c8b224d1bc45d07ae95d2379 |
| Tools | Related Tags - Unspent Outputs |

| Transactions | |
|---|---|
| No. Transactions | 147 |
| Total Received | 111,114.65025667 BTC |
| Final Balance | 0 BTC |

Fig. 15: This user has around 1000 million US dollars in the wallet, but they have spent nothing. We found this using the visualization of the flow of our design of bitcoin transaction graph,

An interesting question, **is this address related to DPR's address?** We found that this address was in the same component as the user altoid's component. **Can we discover a trail for this address to reveal what addresses he used for coin mixing?**

We applied Djikstra's algorithm between these two nodes to discover the shortest path. Hence, this could be trail he used to mix the coins and store them. The final result of the trail is shown in figure 16.



```
1LDNLreKJ6GawBHPgB5yfVLBERi8g3SbQS -> 1BG9jDV3pA1MsJUnvRyWuA2b7PfGd4MZaw ->
12h6TzwPNBvDnppbsqpyXwW4oo5UUKaKSa -> 1EG9HJG9aGqzgGujfNQMiNbyqpKnFxafvE ->
1AHki5AbZYiz4fHkGSTVKN3T1Tv5PwZpnh -> 15TEAwEMxVS3BK718HhwgJg7nxwyJ2ib9y ->
1933phfhK3ZgFQNLGSDXvqCn32k2buXY8a
```

Fig. 16: Trail discovered from user's address to the public key of the wallet having a sketchy transaction.

## 7 Conclusions

Bitcoin architecture is not entirely robust to the completely anonymous transactions. Multi-input heuristics could be used to cluster the users from public key addresses. We presented a novel edge bundling algorithm on how transactions could be annotated to users by making use of this heuristic. We have demonstrated that the exponentially growing data of bitcoin blockchain can still be parsed by a personal computer using our memory efficient algorithm. We proved empirically that bitcoin graphs are indeed decentralized. We presented some visualization to analyze the transaction flow and users of high importance. We applied our findings on Silk Road arrest and our algorithms were able to detect the coin-mixing trail used by the deceiver.

# Bibliography

1. Anderson, N. and Farivar, C. (2013). How the feds took down the dread pirate roberts.
2. Ermilov, D., Panov, M., and Yanovich, Y. (2017). Automatic bitcoin address clustering. In *Machine Learning and Applications (ICMLA), 2017 16th IEEE International Conference on*, pages 461–466. IEEE.
3. Fleder, M., Kester, M. S., and Pillai, S. (2015). Bitcoin transaction graph analysis. *arXiv preprint arXiv:1502.01657*.
4. Harrigan, M. and Fretter, C. (2016). The unreasonable effectiveness of address clustering. In *Ubiquitous Intelligence & Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCom/IoP/SmartWorld), 2016 Intl IEEE Conferences*, pages 368–373. IEEE.
5. Lischke, M. and Fabian, B. (2016). Analyzing the bitcoin network: The first four years. *Future Internet*, 8(1):7.
6. Moser, M., Bohme, R., and Breuker, D. (2013). An inquiry into money laundering tools in the bitcoin ecosystem. In *eCrime Researchers Summit (eCRS), 2013*, pages 1–14. IEEE.
7. Möser, M., Böhme, R., and Breuker, D. (2014). Towards risk scoring of bitcoin transactions. In *International Conference on Financial Cryptography and Data Security*, pages 16–32. Springer.
8. Nakamoto, S. (2008). Bitcoin: A peer-to-peer electronic cash system.
9. Nick, J. D. (2015). Data-driven de-anonymization in bitcoin. Master's thesis, ETH-Zürich.
10. Ober, M., Katzenbeisser, S., and Hamacher, K. (2013). Structure and anonymity of the bitcoin transaction graph. *Future internet*, 5(2):237–250.
11. Reid, F. and Harrigan, M. (2013). An analysis of anonymity in the bitcoin system. In *Security and privacy in social networks*, pages 197–223. Springer.
12. Ron, D. and Shamir, A. (2013). Quantitative analysis of the full bitcoin transaction graph. In *International Conference on Financial Cryptography and Data Security*, pages 6–24. Springer.