Could Network Information Facilitate Address Clustering in Bitcoin?

Till Neudecker and Hannes Hartenstein
Institute of Telematics, Karlsruhe Institute of Technology, Germany
{till.neudecker,hannes.hartenstein}@kit.edu

Abstract. Address clustering tries to break the privacy of bitcoin users by linking all addresses created by an individual user, based on information available from the blockchain. As an alternative information source, observations of the underlying peer-to-peer network have also been used to attack the privacy of users. In this paper, we assess whether combining blockchain and network information may facilitate the clustering process. For this purpose, we apply all applicable clustering heuristics that are known to us to current blockchain information and associate the resulting clusters with IP address information extracted from observing the message flooding process of the bitcoin network. The results indicate that only a small share of clusters (less than 8 %) were conspicuously associated with a single IP address. Also, only a small number of IP addresses showed a conspicuous association with a single cluster.

1 Introduction

The electronic currency system bitcoin [13] allows users to transfer money using pseudonyms represented by public keys (*addresses*). As all transactions in bitcoin are stored in a public blockchain, it is common practice to create new addresses for each transaction. This aims at ensuring the privacy of participants by making the linkage of several addresses difficult. Previous research (e.g., [16]), however, proposed heuristics for the clustering of addresses and showed that it is possible to link several addresses to one user. It was also shown that it can be possible to establish a link between one of a user's addresses and information from additional sources that reveals the user's identity. In the worst case, this knowledge can be used to learn about all financial transactions of an identifiable user.

Before becoming part of the blockchain, transactions are broadcasted through a public peer-to-peer (P2P) network. By joining and observing that network, additional information about the issuer of a transaction might be gained. Several works indicate that such linking is possible (e.g., [6]). However, with users using dynamically assigned IP addresses, operating from clients behind NAT routers or using wallet services, it is not clear whether information obtained by participating in the network and observing the normal message flow could be used in deanonymization of bitcoin users.

One fundamental challenge is that neither for a blockchain based clustering nor for the extracted network based information a ground truth validation can be performed (with a few exceptions). Therefore, we analyze whether clusters created using known heuristics are correlated to IP addresses associated to transactions based on network observations (cf. Fig. 1). As both approaches operate

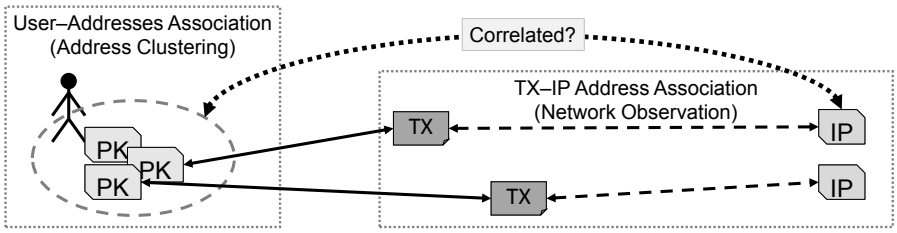

Fig. 1. High-level overview of the used approach: Addresses are clustered using known heuristics; transactions are assigned to IP addresses based on network observations and to clusters based on their content. We then check whether single clusters are conspicuously often associated to a single IP address, and whether single IP addresses are conspicuously often associated to a single cluster.

on disjoint data (blockchain vs. network) but aim at indicating the same outcome (addresses controlled by one user), a correlation would likely mean that both approaches in fact approximate the desired outcome.

The contributions of this paper are twofold:

- We review all published heuristics known to us and apply them to the current blockchain state in a comparable and reproducible manner.
- We show that although for the majority of users no correlation between network information and the clustering performed on blockchain data could be found, a small number of participants exhibit correlations that might make them susceptible to network based deanonymization attacks.

2 Related Work

The anonymity of users in bitcoin has been analyzed in several ways in the past. The fact that all transactions are publicly available facilitated clustering approaches with the goal to group addresses by the controlling user. We will review all published heuristics known to us in detail in Section 4 and briefly sketch related work here. The first analysis was performed by Reid et al. [16] and already made use of the most commonly used heuristic. Meiklejohn et al. [10] proposed additional heuristics based on the behavior of standard clients.

Blockchain information has not only been used for clustering but also for large scale analysis of the distribution of wealth, common transaction patterns, behavior analysis, etc. [17], and for an evaluation of user privacy [1]. More recently, Nick was able to use ground truth data of consumer wallets due to a bug in a client implementation [15]. This work also proposes a heuristic specific to the behavior of consumers in bitcoin. Reasons for the effectiveness of clustering have been given by Harrigan et al. [5], e.g., the incremental growth of clusters.

Network based information has also been used previously. It was shown that the topology of the bitcoin peer-to-peer network can be inferred by using marker IP addresses [2], by exploiting flaws in the bitcoin reference client implementation [12], or by observing the information propagation through the network [14].

Furthermore, the observation of anomalous relaying behavior has been used to map bitcoin addresses to IP addresses [7]. It was also shown that the creation time of transactions can be used to infer the user's time zone [4]. Biryukov et al. [3] performed a man in the middle attack on clients using Tor by becoming the only possible Tor exit node by banning all other exit nodes in the bitcoin network. This also enabled them to link IP addresses to bitcoin addresses.

3 Fundamentals

The two main data objects in bitcoin are *transactions* and *blocks*. Transactions are used to transfer bitcoins between users. Blocks are created in the process of mining and contain a set of accepted transactions that the bitcoin network has agreed on to be valid. We will exclude the details of mining here. Transactions specify inputs and outputs, i.e., sources and destinations of the money flow. With the exception of coinbase transactions, inputs refer to an output of another, previous transaction. These inputs are then spending the output. Obviously, one output must not be spent more than once.

Transactions: All accepted transactions form the transaction graph. The transaction graph is constructed by using all accepted transactions as vertices and by adding one edge from every output to the input that is spending the output. The transaction graph is a directed, acyclic, append-only graph, which represents the current ownership of bitcoins. Intuitively, ownership of bitcoins is the right to spend them. Technically, ownership of bitcoins equals the possession of a private key that corresponds to a public key, which is defined in the output of a transaction. Hence, in order to create a valid transaction, a user must be able to sign the transaction spending an input using the private key corresponding to the public key defined in the spent output. The public keys are also called *addresses*, as they specify where the money is sent to.

For the definition of the heuristics used in clustering, we will use the following notation, which loosely follows the notation used in [10]: Let $t \in T$ be a transaction. Let $\mathcal{P}$ be the set of all addresses specified in all transactions in T. Let the set $\text{inputs}(t) \subseteq \mathcal{P}$ include all addresses referenced by the inputs of a transaction t and the set $\text{outputs}(t) \subseteq \mathcal{P}$ include all addresses contained in the outputs of a transaction t. Let $o_j(t) \in \text{outputs}(t)$ be the j-th output ($j \leq |\text{outputs}(t)|$), and let $i_j(t) \in \text{inputs}(t)$ be the j-th input ($j \leq |\text{inputs}(t)|$).

Each user can create a practically unlimited number of distinct public/private key pairs and use each of them only for one transaction. Hence, each address can be seen as a pseudonym of the user. The goal of address clustering is to partition the set of addresses into subsets (*clusters*), so that each subset contains the addresses under the control of one user.

Network Information: After a transaction is created it needs to be broadcasted through the bitcoin P2P network in order to reach all participants. Especially miners need to receive the transaction, check its correctness, and include the transaction in an upcoming block. The bitcoin P2P network currently con-

sists of 4,200 - 5,700 reachable peers[1] and an unknown number of unreachable peers.

In order to publish a transaction on the network, the user has to either run one of the reachable peers or connect to one of the reachable peers and transmit the transaction. When a new transaction arrives at a peer, the peer checks the correctness of the transaction and rebroadcasts the transaction to all of its neighbors. Therefore, the transaction gets flooded through the whole network. For rebroadcasting, the bitcoin reference client *bitcoind*, which is used by the vast majority of network peers, implements a mechanism called *trickling*: Transactions are not immediately rebroadcasted to all neighbors, but are randomly delayed according to a Poisson distribution.

4 Clustering based on Blockchain Information

Several heuristics for address clustering in bitcoin have been proposed. We will first briefly describe the general procedure for clustering, which uses one or more heuristics, and then describe and discuss the used heuristics.

4.1 Clustering Procedure & Heuristics

The clustering procedure computes a partition $\Pi = \{C_1, C_2, ..., C_n\}$ of the set of all addresses $\mathcal{P}$ with $C_1, ..., C_n$ denoting the resulting clusters. For this, it processes all transactions in their temporal sequence. For each transaction t, all selected heuristics compute a partition $\Pi_t = \{\Pi_t^1, ..., \Pi_t^m\}$ of all input and output addresses of t ($\text{outputs}(t) \cup \text{inputs}(t)$). This transaction specific partition Π_t encodes which addresses used in the transaction are controlled by one user (i.e., those addresses being in one Π_t^i).

The heuristics are applied in a predefined order, each heuristic further altering Π_t. Π_t is then used to update Π: First, all clusters Π_t^i are added to P. Then each added cluster Π_t^i is merged with all existing clusters in Π that contain any of the addresses in Π_t^i. This transitively connects all addresses controlled by one user (according to the applied heuristics).

Heuristic 1 (H1): Multi-Input If a transaction spends more than one input, the transaction needs to be signed using the private keys corresponding to the public keys from *all* inputs. Assuming that the transaction was created by a single user, that user controls all addresses that are input to the transaction. This heuristic was first used in [16] and [10].

For a transaction t the partition determined by this heuristic is

$$\Pi_t = \{\text{inputs}(t), \{o_1(t)\}, ..., \{o_{|\text{outputs}(t)|}(t)\}\}.$$

This heuristic is always applied first and is used for all our clusterings. This heuristic only produces false positives (i.e., clustering addresses that are not

[1] According to our measurements (http://dsn.tm.kit.edu/bitcoin), there are ≈4,200 peers reachable via IPv4 and an additional ≈1,500 peers reachable via IPv6. As we do not know how many peers are dual-stacked (reachable via IPv4 *and* IPv6), we cannot directly determine the exact number of reachable peers.

controlled by the same user into the same cluster), if the assumptions are not correct. This can be either the case if users give services access to their private key (e.g., Mt.Gox) or if transactions are assembled by multiple users in a decentralized fashion (e.g., CoinJoin [9]).

Heuristic 2 (H2): Change Address One output of a transaction can only be spent in its entirety. Hence, if Alice controls an unspent output worth 2 BTC and wants to pay Bob 1 BTC, Alice creates a transaction claiming the 2 BTC as an input with two outputs: One output of 1 BTC to Bob's address and one output of 1 BTC to a *change address* [10] under the control of Alice (assuming no transaction fees). Since the change address as well as the addresses of the inputs (cf. *H1*) are all controlled by Alice, they should be clustered together. The challenge is to identify which output is the change address and which output is the address of the payee, which should be in a different cluster. Meiklejohn et al. [10] proposed the following heuristic to identify the change address: An output $o_j(t)$ is the change address if these four conditions are met:

1. This is the first appearance of the address $o_j(t)$.
2. The transaction t is not a coin generation.
3. There is no address within the outputs, which also appears on the input side (self-change address).
4. Condition 1 is only met for $o_j(t)$ and not also for some $o_k(t)$ with $j \neq k$.

For a transaction t the partition determined by this heuristic (based on Π_t from *H1*) is

$$\Pi_t = \{\text{inputs}(t) \cup \{o_j(t)\}, \{o_1(t)\}, ..., \{o_{j-1}(t)\}, \{o_{j+1}(t)\}, ..., \{o_{|\text{outputs}(t)|}(t)\}\}.$$

The rationale behind this heuristic is that the standard bitcoin client creates a new key pair for change addresses and only uses these addresses once when the received change is spent again. Ancient version of the client used to send change to an address that was also used as input (self-change address).

Obviously, this heuristic can lead to false positives and false negatives. In a transaction with two outputs, which have not appeared before, it is not possible to determine the change address (cond. 4), although there might be one. Also, a transaction could spend money to two payees without any change and the heuristic could mistake one of the payees addresses for the change address.

Heuristic 2 exceptions In order to capture changing wallet behavior, two exceptions to Heuristic 2 have been proposed in [10]. There is no change address in a transaction t if there is an output that...

- had already received exactly one input (**H2a**)
- had been used in a self-change transaction before (**H2b**)

These exceptions captured common behavior in 2013, however, it is not clear whether the exceptions are useful anymore.

We now define an additional exception to heuristic *H2* that makes use of blockchain information that is newer than the current processed transaction t.

The behavior for change addresses is that they are only used once. In *H2* we demand that, in order to qualify as a change address, an address must not occur before t. However, with **H2c** we demand that the address also does not occur in later transactions (except for one occurrence as an input).

Value based (HV): Optimal Change If a transaction has only one output, whose value is smaller than any of its inputs, this output address is likely the change address. This heuristic is based on the behavior of bitcoin clients to minimize the transaction size, i.e., the number of inputs and outputs. If the change was larger than any input, the input could be omitted and the change could be reduced by this input. This heuristic was used in [15].

Consumer based: Redeeming Transaction Nick [15] proposed a heuristic that uses properties of the redeeming transaction of a possible change output (i.e., the transaction with the change output as an input). For a change address it requires that the redeeming transaction has at most two outputs. The heuristic was used specifically for clustering consumer wallets that show this characteristic. As we cannot distinguish between consumer wallets and other wallets, we omit this heuristic from further analysis.

Cluster Growth (HG) In [5] it has been shown that clusters normally grow in steady, but small steps. Especially the merger of two already large clusters by a new transaction is unlikely and might hint at a false positive from one of the applied heuristics. This observation can be formulated as a heuristic that can be applied after other heuristics have already established a transaction specific partition.

HG$_\mathbf{k}$: If updating Π with Π_t would cause the largest affected partition in Π to grow by more than a constant number of k addresses, then set

$$\Pi_t = \{\{i_1(t)\}, ..., \{i_{|\text{inputs}(t)|}\}, \{o_1(t)\}, ..., \{o_{|\text{outputs}(t)|}(t)\}\}.$$

Discussion To our knowledge, we list all heuristics that were published. However, there is a whole class of heuristics that we barely cover. Most described heuristics only consider single transactions. However, heuristics could use the whole transactions graph and base their decisions on any property derived from the graph. The consumer based heuristic and the Cluster Growth heuristic use simple transaction graph information, but much more sophisticated methods, e.g., facilitating metrics such as connectivity or centrality are possible.

Furthermore, we acknowledge that a lot of manual effort can be put into a better clustering by carefully inspecting special cases, modeling specific behavior and manually merging or splitting clusters. For the sake of comparability, we chose not to do any manual intervention in our clustering process.

4.2 Results

We will now compare the results of the clustering process with different combinations of heuristics. The clustering was performed at block 440,349. Using

Table 1. Comparison of all heuristics. Total number of addresses: 196,963,722, total number of transactions: 172,868,721.

Heuristics	# Cluster	∅Size	max size	#clusters w/ size 1
H1	88 m	2.24	12 m	65 m
H1+H2	46 m	4.25	92 m	29 m
H1+H2a	51 m	3.89	87 m	32 m
H1+H2b	63 m	3.10	66 m	40 m
H1+H2c	48 m	4.13	85 m	30 m
H1+HV	72 m	2.71	76 m	62 m
H1+HG$_{10}$	146 m	1.34	0.1 m	123 m
H1+HG$_{100}$	121 m	1.62	0.25 m	97 m
H1+HG$_{1000}$	108 m	1.83	1 m	84 m
H1+HG$_{10000}$	104 m	1.88	8 m	81 m

machines equipped with a Xeon E7-8837 and 512 GB memory, one run of our implementation[2] of the clustering process took about 30 minutes to complete. Prior to clustering we generated the transaction graph as a pointer-based data structure. This data structure is then read to memory by the clustering process, which is run completely in-memory and requires no further hard disk accesses.

Table 1 lists a comparison of key properties of the resulting clusterings for the heuristics *H1*, all discussed variants of *H2*, *HV*, and several variants of *HG*. Details on the distribution of cluster sizes are given in the Appendix. Applying only heuristic *H1* results in a clustering with 88 m clusters. Additionally applying *H2* causes more clusters to be merged, hence resulting in fewer, but bigger, clusters. Additionally applying variants of *HG*, however, causes fewer clusters to be merged, hence resulting in more, but smaller, clusters.

The different variants of heuristic *H2* lead to 46 m to 63 m clusters. The three exceptions to *H2* cause fewer clusters to be merged than by applying *H1* and *H2* only. The strongest effect on the resulting clusters has *H2b*, which reduces the average cluster size from 4.25 for *H2* to 3.1 addresses per cluster for *H2b*.

The value based heuristic *HV* has only a small effect on the average cluster size (grows to 2.71 addresses per cluster) but a large effect on the size of the largest cluster (from 12 m to 76 m). A possible explanation for the result is that a disproportionately large share of transactions that originated from that super-cluster have a combination of input and output values that makes *HV* applicable to them, thus merging more addresses into the super-cluster.

A small choice of the parameter k for the heuristic *HG* causes fewer clusters to be merged as the threshold is easily exceeded. This causes the average cluster size to decrease down to 1.34 addresses per cluster for HG_{10}. Notably, there are only minor changes in the number of clusters with a size of 10 to 100,000 addresses (cf. Appendix). Most likely, transactions that cause a false positive in *H1* are less likely to occur in these medium sized clusters.

In all variants the largest identified cluster contains between 100,000 and 92 m addresses. This cluster contains among others the addresses of the former

[2] https://github.com/tillneu/bitcoin-clusterer

exchange Mt.Gox. The existence of this super-cluster was also discussed in [5]. The size of that cluster is substantially increased by application of variants of *H2* and *HV*, whereas the application of *HG* can limit the growth of that cluster.

5 Network Information

We will now explain how network based information was acquired and how that information is compared to the blockchain information based clustering results. The main idea is to associate IP addresses to transactions based on observations on the bitcoin P2P network and then use the previously established linking between clusters and transactions in order to determine the correlation between clusters and IP addresses.

5.1 Association of Transactions and IP Addresses

In order to observe transactions being flooded through the network, we deployed two monitor peers that maintain connections to all reachable peers in the network and log for each transaction, when it is received from each peer in the network. For each transaction there is one peer (*originator*) which first sent the transaction to our monitor peer. We want to associate one IP address to each transaction. However, we cannot conclude that the first peer we received a transaction from has really first brought the transaction to the network, nor can we conclude that the peer generated the transaction. First, the user could connect to any reachable peer in the network, send the transaction to that peer and leave the network afterwards. Secondly, due to trickling, the transaction can be sent to other network peers, which might forward the transaction to our monitor peers before we receive the transaction from the creating peer. Therefore, we apply several heuristics that aim at reducing the number of obviously false mappings:

- If both monitor peers first received a transaction from different peers, we discard both possible originators.
- If the time difference at which the transaction is received from the originator by both monitor peers differs by $\geq$100 ms, the originator is discarded.
- The subsequent receptions of the transaction from other peers must not be faster than what the speed of light in fiber allows. By using GeoIP services[3], we can approximate the location of the other network peers and establish a lower bound on the time it takes for a transaction to be transmitted from the originator to our monitor peer *via* any other network peer. If we receive a transaction faster than that lower bound, we discard the originator.

During the monitored period between block 366,000 (2015-07-19) and block 440,349 (2016-11-24), 96,520,958 transactions were added to the blockchain. For 9,934,056 of these transactions ($\approx 10\,\%$), we identified an originator IP address using the heuristics described above. In total, 79,079 unique IP addresses appeared as originators. This leads to an average of about 125 transactions per IP address. However, the number of transactions associated per IP address follows a heavy tailed distribution. Fig. 2 shows the distribution of how many transactions were associated with each IP address. Most IP addresses were an originator

[3] `http://dev.maxmind.com/geoip/`

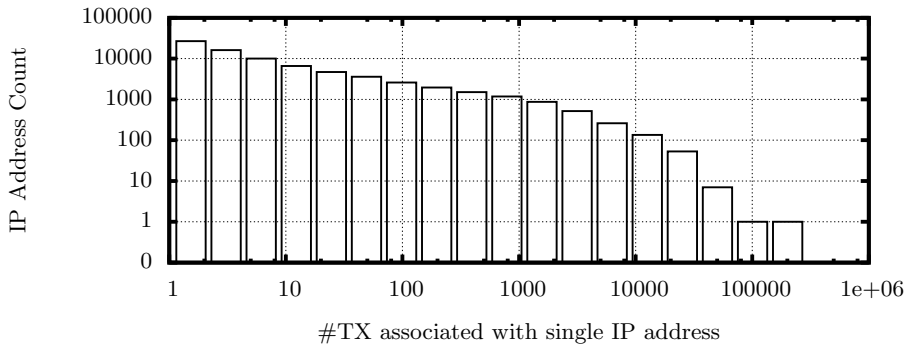

Fig. 2. Histogram of the number of unique transactions associated per IP address. Read as: *There are 10,000 IP addresses, each of which are associated to 4 to 8 transactions.*

address only for a small number of transactions. However, two IP addresses were originators for more than 65,000 transactions. Interestingly, both of these IP addresses (one of which IPv4 and one IPv6) are in IP ranges assigned to the same hosting provider.

Although we are able to associate IP addresses to transactions, we do not know whether the mapped IP addresses in fact identify the user that issued the transaction and simply regard the IP address as a piece of information that might be linked to the user. In order to analyze that linking, we will now compare the results from the clustering based on the transaction graph to the collected IP address information.

5.2 Methodology

We will now introduce the notation used for the association of clusters with IP address information. For the association between transactions and clusters we use the following notation: Let $c(t)$ describe the cluster that issued a transaction t according to *H1*. Let the set of transactions issued by a cluster C be $T_C := \{t \in T : c(t) = C\}$. For the association between transactions and IP addresses as described in Section 5.1 we use the following notation: Let $\mathcal{A}$ be the set of all observed IP addresses. Let $a(t) \in \mathcal{A}$ describe the IP address of the originator (if any) of a transaction t. Finally, we define the tuple of all IP addresses associated with a cluster C as $A_C = (a(t) : t \in T_C)$. A_C is defined as a tuple because single IP addresses can occur multiple times in A_C and we are interested in that count.

The main question now is whether there is a correlation between clusters and IP addresses or whether for each transaction the originator is simply a random IP address. Both, IP addresses and clusters, are nominal variables that cannot be ranked in any way. Standard statistical methods (e.g., [11]) would suggest to fill a contingency table with all observed IP addresses as one dimension and all clusters as the other dimension. Then, for each tuple (IP address, Cluster) the expected frequency and the observed frequency could be compared. However, a problem with the data is that the contingency table is very sparsely populated.

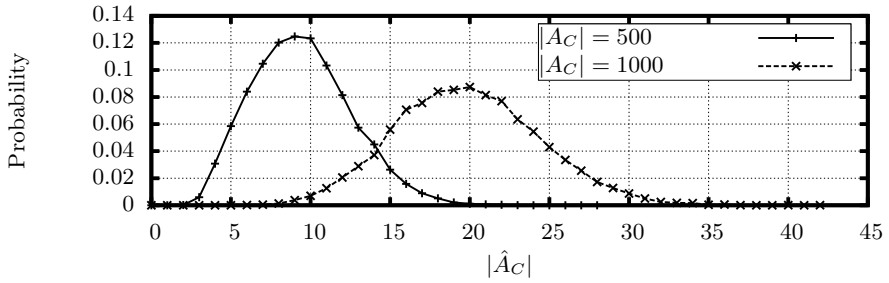

Fig. 3. Probability distribution $P_i(X = |\hat{A}_C|)$ for $|A_C| = 500$ and 1,000 transactions, respectively, assuming independence and given the empirical IP address counts (cf. Fig. 2). Values numerically approximated.

In order to perform the chi squared test, no more than 20% of the expected frequencies should be less than 5 and all individual expected frequencies should be 1 or greater [18], which is not the case for our data. Even if the frequencies were sufficient, the large sample size would cause biased results [8].

Therefore, we analyze each cluster C separately in order to see whether the associated IP addresses A_C are independent. The tuple of associated IP addresses A_C can be seen as the result of a random experiment, where for each cluster C $|A_C|$ addresses are chosen according to a probability distribution. If clusters and IP addresses are independent, the probability to choose an IP address A would be $P(A) = |A| / \sum_{A' \in \mathcal{A}} |A'|$ (with $|A|$ being the total observation count of A, i.e., the share of an IP address in all observations, cf. Fig. 2).

Again, most statistical tests to check whether the sample A_C was chosen according to $P(A)$ cannot be used due to the low sample sizes and low expected frequencies. Hence, we limit our analysis to the IP address $\hat{A}$ that occurs most frequently in A_C, and its frequency $|\hat{A}_C|$. Under the hypothesis of independence, we can calculate the probability of observing any value for $|\hat{A}_C|$. Fig. 3 shows the probability distribution $P_i(X = |\hat{A}_C|)$ that describes the probability of observing a specific value for $|\hat{A}_C|$, assuming independence of IP addresses and clusters. For large values of $|A_C|$, the distribution can be approximated with the binomial distribution with p being the probability of the most likely IP address ($p \approx 0.02$ for our data). For a cluster C, we reject the independence hypothesis if the probability of observing the most frequent IP address $\hat{A_C}$ at least $|\hat{A}_C|$ times is less than 1 %, according to P_i. We then add this cluster to the set of conspicuous clusters $C^+ = \{C : P_i(X \geq |\hat{A}_C|) < 1\,\%\}$. The chosen significance level implies that about 1 % of the clusters in C^+ actually are not conspicuous.

Obviously, in addition to checking for each cluster whether the associated IP addresses were randomly chosen, we can also check for each IP address whether the associated clusters are randomly chosen. This analysis has been also performed using the same method as described above for the opposite direction with T_A denoting the set of transactions associated with an IP address A and $\mathcal{A}^+$ the set of conspicuous IP addresses according to the hypothesis testing.

Table 2. Comparison of the number of clusters with at least two associated IP addresses ($|\{C : |A_C| \geq 2\}|$) and the number and share of conspicuous clusters (C^+), and the share of conspicuous IP addresses ($\mathcal{A}^+$) for various heuristics.

Heuristics	$\lvert\{C : \lvert A_C\rvert \geq 2\}\rvert$	$\lvert C^+\rvert$	$\frac{\lvert C^+\rvert}{\lvert\{C:\lvert A_C\rvert\geq 2\}\rvert}$	$\frac{\lvert\mathcal{A}^+\rvert}{\lvert\{A:\lvert T_A\rvert\geq 2\}\rvert}$
H1	282,950	14,879	5.26 %	18.7 %
H1+H2	398,802	32,623	8.18 %	6.2 %
H1+H2a	387,696	32,026	8.26 %	6.2 %
H1+H2b	456,063	35,138	7.70 %	6.5 %
H1+H2c	452,189	35,602	7.87 %	6.7 %
H1+HV	296,132	14,736	4.97 %	6.9 %
H1+HG$_{10}$	299,140	15,537	5.19 %	16.7 %
H1+HG$_{100}$	300,927	15,755	5.23 %	19.6 %
H1+HG$_{1000}$	301,775	16,434	5.45 %	20.2 %
H1+HG$_{10000}$	308,900	18,788	6.08 %	19.7 %

5.3 Results & Discussion

From our data we selected all clusters with at least two IP addresses associated ($|A_C| \geq 2$), determined $|\hat{A}_C|$ for these clusters, and calculated the set of conspicuous clusters C^+. Table 2 shows the number of clusters with at least two associated IP addresses ($|\{C : |A_C| \geq 2\}|$) and the number of conspicuous clusters $|C^+|$ for various heuristics. The number of clusters with at least two associated IP addresses varies between 283k and 456k clusters. Comparing these numbers to the total number of clusters (cf. Table 1) shows, that only a small percentage of all clusters has two IP addresses associated, with the highest percentage for the *H1+H2c* combination.

The number of clusters $|C^+|$ with a too-large $|\hat{A}_C|$ varies between 15k and 35k, which corresponds to 5 % to 8.3 % of the considered clusters. For comparison, when randomly selecting IP addresses based on their a-priori probability $P(a)$, the share of conspicuous clusters is around 1 %. The results indicate that the highest correlation between clusters and their associated IP addresses exists, when clustering using variants of *H2*. For the value based heuristic, the growth based heuristic, and the base heuristic *H1*, fewer conspicuous clusters were found.

Table 2 also shows the share of conspicuous IP addresses $\mathcal{A}^+$ among those IP addresses with at least two associated transactions. The share varies between 6.2 % and 20.2 % with the smallest percentages for clusterings with variants of *H2*. This is caused by the extremely large super cluster that is created by these heuristics (cf. Table 1): The probability to randomly select that cluster very often (assuming independence) rises with the number of transactions associated with that cluster. Therefore, the independence hypothesis gets accepted for more IP addresses.

Only for a small share of clusters and IP addresses, a correlation between clusters and network information could be shown. At least for these clusters, information obtained by observing the network could also be used in a constructive way during the clustering process. For example, the set of candidate clusters for a transaction could be reduced based on networking information. Also, the

information could be used for tie breaking when having multiple change address candidates.

For the majority of clusters and IP addresses, we did not observe any correlation to network information. This could mean that there is no correlation, or that the used method did not reveal a correlation. For example, a more powerful observer with more monitoring nodes could be able to associate IP addresses to transactions more precisely. Furthermore, the statistical analysis used here only reveals certain correlations between a cluster and a single IP address.

6 Conclusion

In this paper we performed address clustering in bitcoin according to published heuristics, compared the resulting clusters to IP address information obtained from observations in the bitcoin P2P network, and showed that only a small share of clusters was conspicuously associated with a single IP address, and that only a small number of IP addresses showed a conspicuous association with a single cluster.

Our results indicate that for the vast majority of users network information cannot facilitate address clustering easily. However, a small number of participants exhibit correlations that might make them susceptible to network based deanonymization attacks. A more precise network observation or better clustering heuristics might reveal further correlations that could not be observed with our approach. A next step could be to identify the anomalous behavior that caused the revealed correlations. Since this would require an in-depth analysis of single entities on the network, we decided not to carry out such an analysis without ensuring the user's privacy. We emphasize that for ethical reasons no further attempt at linking the conspicuous IP addresses or clusters to other available information was performed.

In future work, a privacy preserving method for identifying the causes of the correlation should be developed. Such an analysis could point to possible improvements in the P2P protocol or specific client implementations. Furthermore, the used heuristic for extracting the originator from the network observation could be improved to consider IP address changes over time or the aggregation of IP addresses by provider or location. Finally, the statistical analysis might benefit from more advanced methods to establish sharper bounds on possible correlations.

Acknowledgement

This work was supported by the German Federal Ministry of Education and Research (BMBF) within the project *KASTEL_IoE* in the Competence Center for Applied Security Technology (*KASTEL*). The authors would like to thank the anonymous reviewers for their valuable comments and suggestions.

References

1. Androulaki, E., Karame, G.O., Roeschlin, M., Scherer, T., Capkun, S.: Evaluating user privacy in bitcoin. In: International Conference on Financial Cryptography and Data Security. pp. 34–51. Springer (2013)
2. Biryukov, A., Khovratovich, D., Pustogarov, I.: Deanonymisation of clients in bitcoin p2p network. In: Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security. ACM (2014)
3. Biryukov, A., Pustogarov, I.: Bitcoin over tor isn't a good idea. arXiv preprint arXiv:1410.6079 (2014)
4. DuPont, J., Squicciarini, A.C.: Toward de-anonymizing bitcoin by mapping users location. In: Proceedings of the 5th ACM Conference on Data and Application Security and Privacy. pp. 139–141. ACM (2015)
5. Harrigan, M., Fretter, C.: The unreasonable effectiveness of address clustering. arXiv preprint arXiv:1605.06369 (2016)
6. Kaminsky, D.: black ops of tcp/ip. Black Hat USA (2011)
7. Koshy, P., Koshy, D., McDaniel, P.: An analysis of anonymity in bitcoin using p2p network traffic. In: Financial Cryptography and Data Security. Lecture Notes in Computer Science, vol. 8437, pp. 469–485. Springer Berlin Heidelberg (2014), `http://dx.doi.org/10.1007/978-3-662-45472-5_30`
8. Lin, M., Lucas Jr, H.C., Shmueli, G.: Research commentary-too big to fail: large samples and the p-value problem. Information Systems Research 24(4), 906–917 (2013)
9. Maxwell, G.: Coinjoin: Bitcoin privacy for the real world. `https://bitcointalk.org/index.php?topic=279249` (2013), accessed: 27.09.2016
10. Meiklejohn, S., Pomarole, M., Jordan, G., Levchenko, K., McCoy, D., Voelker, G.M., Savage, S.: A fistful of bitcoins: characterizing payments among men with no names. In: Proceedings of the 2013 conference on Internet measurement conference. pp. 127–140. ACM (2013)
11. Mendenhall, W., Beaver, R.J., Beaver, B.M.: Introduction to probability and statistics. Cengage Learning (2012)
12. Miller, A., Litton, J., Pachulski, A., Gupta, N., Levin, D., Spring, N., Bhattacharjee, B.: Discovering bitcoin's public topology and influential nodes (2015)
13. Nakamoto, S.: Bitcoin: A Peer-to-Peer Electronic Cash System 1(2012), 28 (2008)
14. Neudecker, T., Andelfinger, P., Hartenstein, H.: Timing analysis for inferring the topology of the bitcoin peer-to-peer network. In: 2016 Intl IEEE Conference on Advanced and Trusted Computing (ATC). pp. 358–367 (July 2016)
15. Nick, J.D.: Data-Driven De-Anonymization in Bitcoin. Master's thesis, ETH-Zürich (2015)
16. Reid, F., Harrigan, M.: An analysis of anonymity in the bitcoin system. In: Security and privacy in social networks, pp. 197–223. Springer (2013)
17. Ron, D., Shamir, A.: Quantitative analysis of the full bitcoin transaction graph. In: International Conference on Financial Cryptography and Data Security. pp. 6–24. Springer (2013)
18. Yates, D., Moore, D., McCabe, G.: The Practice of Statistics. WH Freeman and Company, New York, NY (1996)

Appendix

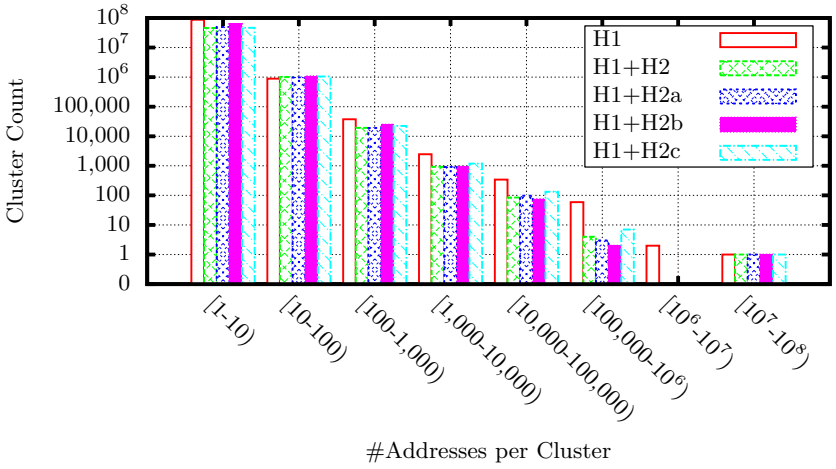

Fig. 4. Histogram of the number of clusters for various sizes (i.e., number of addresses per cluster).

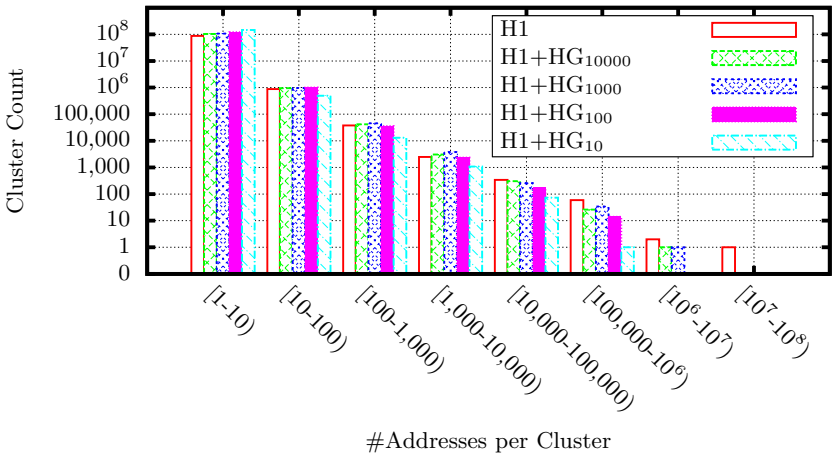

Fig. 5. Histogram of the number of clusters for various sizes (i.e., number of addresses per cluster).

Fig. 4 and Fig. 5 show a comparison of the resulting cluster sizes for all discussed clustering heuristics and various parameterizations of the growth based heuristic *HG*. For all heuristics, the cluster sizes roughly follow a power-law distribution.