UNIVERSITÀ DI PISA

Deanonymization techniques for Bitcoin

Matteo Pinna

Introduction

Deanonymization techniques in Bitcoin aim at breaking the pseudo-anonymity provided by the blockchain system through address clustering, i.e. by linking the addresses controlled by the same user leveraging information available from the blockchain itself.
Two widely used heuristics are going to be presented, one based on *Multi Input Addresses (MIH)* and one based on *One-time Change Address (OCH)*. Then, analyses related to the application of the first heuristic and to the application of both heuristics together will be displayed.

The considered data-set starts from the genesis block and ends at the lock mined on 29-12-210 (block height $100,001$). Notice that the validated version of such data-set (*Bitcoin analysis* midterm) has been considered, to avoid performing validity checks on transactions while applying the heuristics logic.

Implementation

The proposed heuristics and their evaluation have been implemented in Python. The external libraries that have been used are the following: *pandas* and *numpy* to convert the data-set into a data-frame and process it, *networkx* to compute the transitive closure between clusters, finally *matplotlib* and *matplotlib.pyplot* in order to plot and visualize the results.

Multiple Input Addresses Heuristic (MIH)

The first heuristic performs address clustering based on the idea that, to make a valid transaction, the sender needs to provide the signatures signed using the private keys corresponding to the public key addresses in all inputs to prove the ownership of those addresses. Hence, it is safe to assume that all addresses in the inputs of a transaction belong to a single user.

Def. MIH [1]: For any transaction $t = (I, O, c, s)$, if the numbers of inputs in t is more than one ($|I| > 1$), then all addresses in I belong to the same user, i.e., for any $(A, v) \in I, A \rightarrow u$, where u is the ID for the user-controlled addresses in I. Each new address in the outputs O belongs to a new clusters.

In Algorithm 1 the pseudo-code related to the implementation of the first heuristic.

Notice that, since the first heuristic creates clusters just by looking at the addresses in the inputs, it is trivial to notice that those addresses that appear only as outputs will necessarily form isolated clusters (size 1). Hence, at the beginning of the algorithm those addresses are stored so that at the end it is possible to just concatenate them to the other clusters found performing the transitive closure, which is implemented by leveraging properties of graphs (i.e. connected components).

Accuracy

This heuristic is generally accurate but it can still lead to false positives. For instance, Bitcoin allows multiple payments from multiple spenders to be combined into a single transaction in a trust-less way (e.g. CoinJoin [2]) and nowadays there exists multiple services offering input transactions. Hence, this heuristic would cluster such inputs addresses to one single user even though they belong to different ones.

To improve the accuracy of the heuristic and possibly reduce false positives it would be ideal to be able to identify when such services are being used. In this way it would be possible to avoid considering multi sender transactions during the analysis.

One-time Change Address Heuristic (OCH)

The second heuristic performs clustering based on one-time change addresses, which are change addresses automatically generated by the system when change is required in a transaction.

Algorithm 1 MIH

1: $T \leftarrow transactions$
2: **procedure** MIH(T)
3: Get all addresses never appearing as input
4:
5: Generate pairs (edges) between tx ids sharing addresses ▷ $tx_x \rightarrow tx_y$
6: G = pairsToGraph() ▷ Create graphs with generated pairs
7: $grps \leftarrow G.connectedComponents()$ ▷ Transitive closure through graph connected components
8:
9: Label each address with a cluster id according to $grps$
10: Add cluster for each address found on step **(3)** ▷ Output addresses never appearing as input form isolated clusters

Def. OCH [1]: An output address O is a (one-time) change address of a transaction t if the following four conditions are satisfied:

1. This is the first appearance of address O;
2. The transaction t is not coin generation (coinbase);
3. There is no address among the outputs that also appears in the inputs (self-change address);
4. The output addresses other than O do not satisfy condition (1.);

The (one-time) change address in the outputs belongs to the same user as the input addresses.

In Algorithm 2 the pseudo-code related to the implementation of the first heuristic.

The implementation is relatively simple, we're just iterating over inputs-outputs of each transaction after grouping by tx_id and checking all the conditions previously defined in the definition of the heuristic.

Accuracy

Identifying the change address in a transaction is extremely difficult and there is no absolutely reliable way to detect one-time change addresses, hence the second heuristic may lead to both false positives and false negatives. For instance, if the inputs are spent without change then this heuristic could identify one of the destination addresses as a change address (false positive) or if the output addresses of a transaction do not appear in previous transactions then it is impossible to determine the change address even if it is present (false negative).

One way to improve this heuristic accuracy and efficiency would be to further restrict its definition like proposed by Emilov *et al.* [3]. Specifically, by considering only non-coinbase transactions that have exactly two outputs $\#OutputAddr = 2$ and with a number of inputs not equal to two $\#InputAddr \neq 2$.

MIH + OCH

Before presenting the analyses, it is relevant to mention how the results from the heuristics are merged together, since the following charts will represent the results of the application of the first heuristic and the application of both the first and the second one.
The resulting clusters from an heuristic are merged with the clusters of the previous one in the following way:

- If the subset has not intersection with any previous cluster, a new cluster ID is assigned to the addresses in such subset
- If there are previous clusters that have intersection with this subset, all these clusters are merged together to obtain a single larger cluster

In Algorithm 3 the pseudo-code related to the merging of the resulting clusters from the two heuristics.

The idea is to merge the two set of clusters in a single data-frame, and temporarily assign a new cluster id to the clusters from OCH, in order to avoid collisions before computing the transitive closure. Then, like in Algorithm 1 the transitive closure between clusters having an intersection is computed by leveraging graph properties.

Accuracy

Besides the improvements on MIH and OCH previously mentioned, a possible strategy that could be used to improve the clustering would be to apply a new layer (heuristic) based on the idea of address reuse, like proposed in [1]. The idea is that, since a one-time change address is system generated it will appear only one time as output address, when change is required and when its generation occurs.

It is possible to use this intuition to avoid both false positives and false negatives after applying OCH, let's take a look at two examples. For instance, if in a transaction T an output O_1 is identified as one-time

Algorithm 2 OCH

```
1: T ← transactions
2: procedure OCH(T)
3:     Group by transaction id
4:     for each group do
5:         candidates ← list()
6:         if !coinbase AND group.outputs > 1 then          ▷ Not coinbase and not only one output
7:             for each output in group do
8:                 if output in inputs then                  ▷ Violates (3.)
9:                     goto 20
10:                else if output first occurrence then
11:                    candidates.append(output)
12:                end if
13:            end for
14:        end if
15:
16:        if len(candidates) == 1 then          ▷ Only one address must be at its first occurrence
17:            Create cluster with (group.inputs + change address)
18:        end if
19:
20:        Add inputs and outputs to encountered addresses          ▷ Keep track of occurrences for (1.) and (4.)
21:    end for
```

Algorithm 3 MIH+OCH

```
1: procedure MIH+OCH(MIH_clusters, OCH_clusters)
2:     Merge the two set of clusters together, if two cluster id collide create a new temporary one
3:
4:     Generate pairs (edges) between cluster ids sharing addresses          ▷ cluster_x → cluster_y
5:     G = pairsToGraph()                                  ▷ Create graphs with generated pairs
6:     grps ← G.connectedComponents()          ▷ Transitive closure through graph connected components
7:
8:     Label each address with a cluster id according to grps
```

change address but the same output appears in a future transaction T', that means such output was in fact a false positive. Hence, it is now possible to adjust the clustering and remove the false positive, as shown in Figure 1.

Another example, related to false negatives is shown in Figure 2. If in a transaction T it is not possible to identify any one-time change address because two transactions are at their first occurrence (O_1 and O_2), then if in a future transaction O_2 appears again as output it is clear that such address can't be a one-time change address, instead O_1 is now (possibly) identified as one.

Analyses

In this section, the following analyses, both when applying MIH alone and when applying MIH and OCH together, are going to be presented:

- Distribution of the size of the address clusters
- Address reduction ratio
- Temporal trend of the address reduction over different time scales (# of blocks)

Finally, as an additional analysis of choice, in the last subsection the address reduction improvement over time (i.e. the difference between MIH and MIH+OCH address reduction at each time-step) will be displayed.

Clusters distribution

In Figure 3 the distribution of the size of the address clusters when applying the first heuristic, and in Figure 4 the same distribution when applying both heuristics.

Address reduction

In order to evaluate the effectiveness (address reduction) of the applied heuristics, the metric introduced in [1] has been considered,

$$R = \frac{|A| - |C|}{|A|} \tag{1}$$

with A being the set of all addresses in the transactions and C the set of all clusters after the clustering.

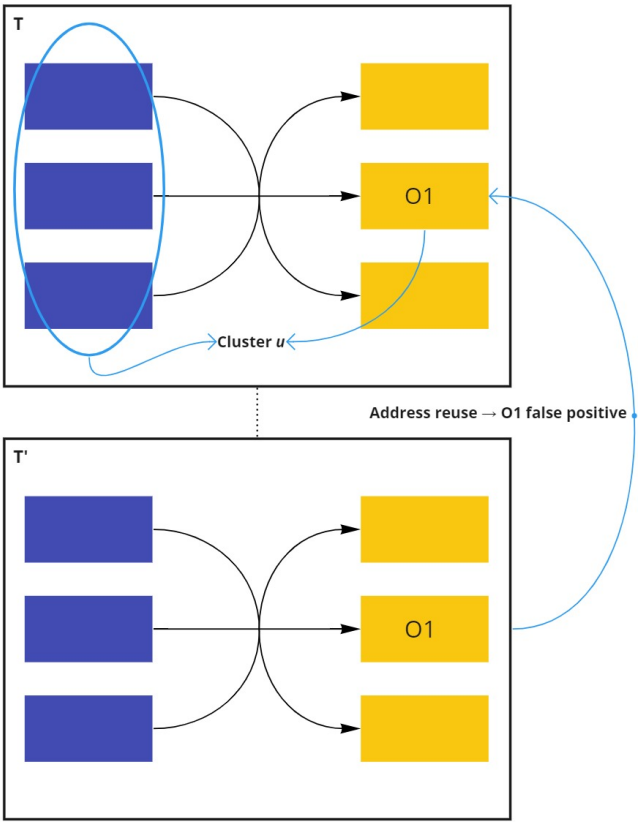

Figure 1: False positive identification with address reuse strategy.

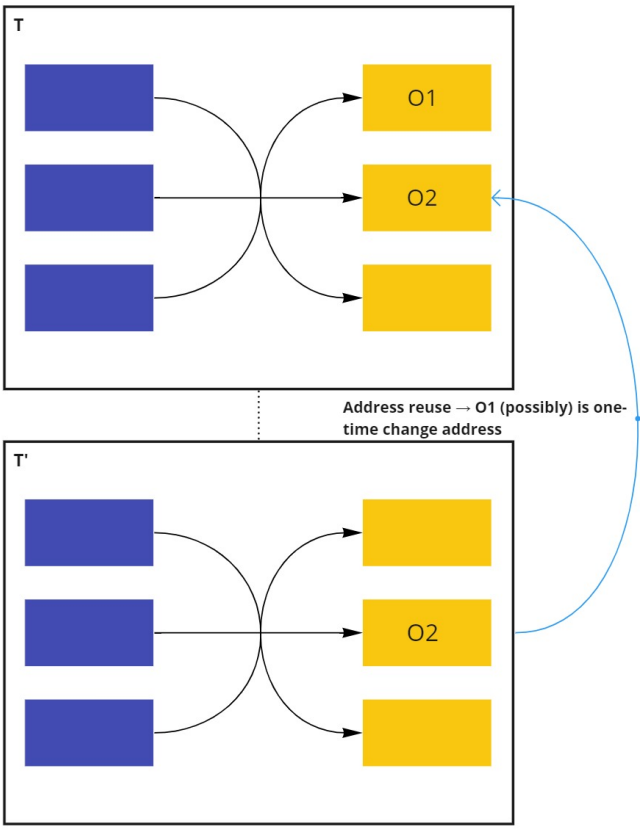

Figure 2: (Possible) false negative identification with address reuse strategy.

In Table 1 the address reduction, and the parameters needed for its computation.

Address reduction trend

The temporal trend of addresses reduction has been performed according to different time scales. Consider K as the number of blocks of each time-step, the analyses have been computed for the following values $K = \{4.380, 8.760, 17.520\}$. Considering that a block is mined every 10 minutes, such parameters respectively correspond to 1, 2 and 4 months.

It is possible to notice from Figures 5, 6 and 7 that the address reduction increases rapidly after the creation of Bitcoin. By considering a bigger data-set it could be possible to observe how the behaviour changes (e.g. whether it reaches a plateau or not).

It is also interesting to observe how the second heuristic does provide any improvement in the reduction ratio until approximately block height $50,000$.

Reduction improvement trend

As additional analysis it has been decided to analyze the temporal trend of the address reduction ratio improvement, i.e. the difference between the address reduction when applying only MIH and when applying both MIH and OCH. The time-step considered is the smallest one ($K = 4,380$ blocks), in order to obtain more accurate results.

In Figure 8 it is possible to observe the temporal trend of the improvement, as well as the average improvement, which, in the considered time-frame, is equivalent to $0.053 \rightarrow 5.3\%$.

Heuristic	No. of Addresses	No. of Clusters	Percent of Reduction
MIH	174, 698	124, 643	28.65
MIH+OCH	174, 698	97, 682	44.08

Table 1: Addresses set size, clusters set size and percent of addresses reduction.

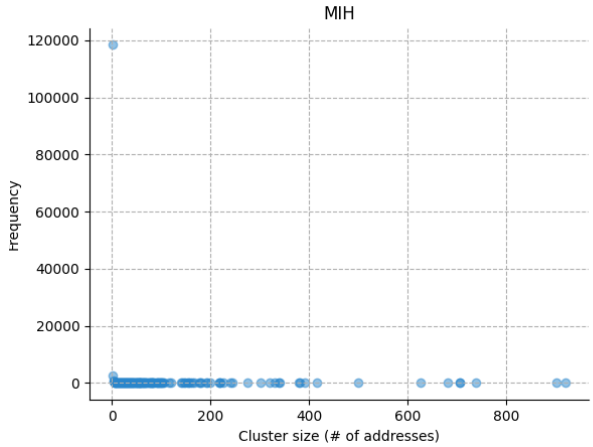

Figure 3: Address clusters size distribution (MIH)

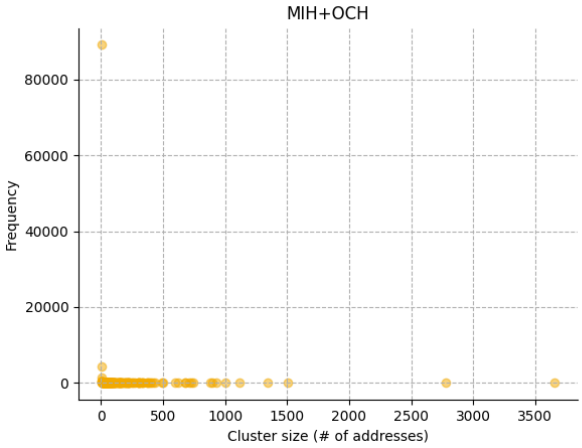

Figure 4: Address clusters size distribution (MIH + OCH)

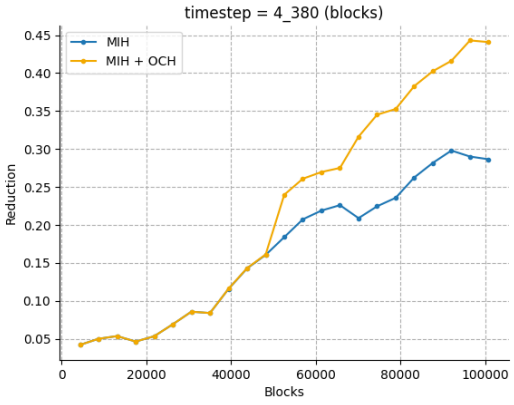

Figure 5: Address reduction ratio temporal trend with 1 month as time-step.

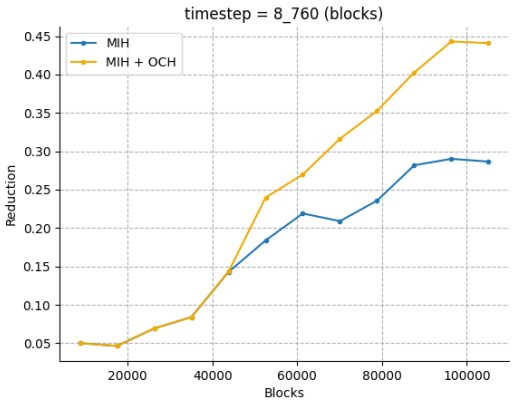

Figure 6: Address reduction ratio temporal trend with 2 months as time-step.

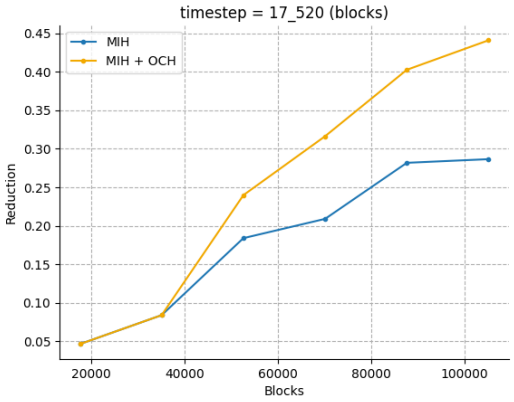

Figure 7: Address reduction ratio temporal trend with 4 months as time-step.

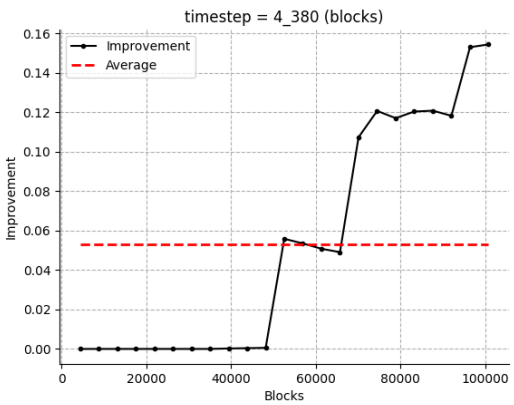

Figure 8: Address reduction improvement temporal trend with 1 month as time-step.

References

[1] Yuhang Zhang, Jun Wang, and Jie Luo. "Heuristic-based address clustering in bitcoin." In: *IEEE Access* 8 (2020), pp. 210582–210591.

[2] Gregory Maxwell. "Coinjoin: Bitcoin privacy for the real world, 2013." In: *https://bitcointalk.org/?topic=279249* (2013).

[3] Dmitry Ermilov, Maxim Panov, and Yury Yanovich. "Automatic bitcoin address clustering." In: *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE. 2017, pp. 461–466.