Towards a better identification of Bitcoin actors by supervised learning

Rafael Ramos Tubino, Céline Robardet, Rémy Cazabet

Towards a better identification of Bitcoin actors by supervised learning

Rafael Ramos Tubino[a], Céline Robardet[b], Rémy Cazabet[a]

[a] *Univ Lyon,* ***UCBL****, CNRS, INSA Lyon, LIRIS, UMR5205, F-69622 Villeurbanne, France*
[b] *Univ Lyon,* ***INSA Lyon****, CNRS, UCBL, LIRIS, UMR5205, F-69621 Villeurbanne, France*

Abstract

Bitcoin is the most widely used crypto-currency, and one of the most studied. Thanks to the open nature of the Blockchain, transaction records are freely accessible and can be analyzed by anyone. The first step in most analytics work is to group anonymous addresses into a set of addresses, called aggregates, that are meant to correspond to unique actors. In this paper, we propose a new method to discover more accurate address aggregates using supervised learning. We introduce a way to create a labeled training set based on reliable heuristics and external information, and propose two methods. The first automatically finds address aggregates from a set of transactions. The second improves an address aggregate of a target actor. We empirically validate our results on large-scale datasets. A striking result of our analysis is that training a model to recognize the change addresses of a particular actor is more efficient than using a larger dataset that does not target that particular actor. In doing so, we clearly show the feasibility and interest of supervised machine learning to identify Bitcoin actors.

1. Introduction

Introduced in 2009, Bitcoin (BTC) is a crypto-currency based on the principle of the Blockchain. It is a decentralized monetary system that operates without a central authority. The Blockchain acts as an open ledger, where anyone can verify that the account trying to spend coins owns these coins. The consequence of this principle is that the transaction data between – pseudonymous – actors have been freely accessible since the beginning of the operation of Bitcoin until today.

The anonymity of these operations is guaranteed by two factors. First, the transactions are made between pseudonymous Bitcoin addresses corresponding to public keys of a cryptographic mechanism making it possible to identify the owner of the coins [1]. Second, anyone is free to create new addresses at any time without charge. Consequently, the same actor receiving several payments can use a different address each time, thus prohibiting observers from easily following their activity.

Identifying the multiple addresses used by a same actor is one of the major challenges of Bitcoin analysis, and is usually the first pre-processing step of any transactional analysis, as for instance in [2, 3]. Indeed, to describe the activity of an actor in this system, it is necessary to identify all their transactions and therefore the different addresses they uses to carry them out. The task of identifying actors has been considered so far either by ad-hoc heuristics, or by unsupervised approaches such as the detection of communities in the graph linking the bitcoin addresses involved in the same transactions, or by machine learning methods computing vectors in a euclidean space such that similar address usages are closed in that space. These vectors make possible the use of clustering methods.

In this article, we rather formalize the problem into a supervised task. Our objective is to detect so-called *change addresses*: in Bitcoin, when one needs to do a payment, one must spend the whole amount received in another payment. If the amount received is strictly superior to the amount one needs to send, then the rest (the change) of the transaction is sent to another address of the same user, a change address. There is no way, from the blockchain data, to know which address is the change and which is the payment. Recognizing one from the other is the task of change address detection.

The main difficulty in using supervised learning for this task is to have a reliable and high quality training set. We propose to build it by an *a posteriori* analysis of the Blockchain. At time t, we use all previous information about the address's usage to build the descriptive characteristics of the address, and we assign it a label based on its subsequent use as an entry address for a transaction involving the same actor as input.

We show that this method yields convincing results. It is able to detect change addresses that were not found by traditional methods, and thus enlarge the set of known addresses of an actor. Moreover, having a supervised method makes it possible to define new tasks, such as the identification of specific actors. We show that this method is promising, because training a model to recognize change for a particular actor is much more reliable than training it for all actors without distinction.

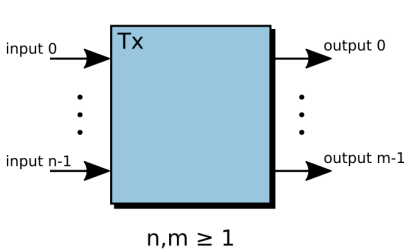
The article is organized as follows. In Section 2, we formally introduce the problem of identifying Bitcoin actors, and the associated state of the art. Section 3 presents the dataset, and how we formalize the actor identification problem into two possible supervised tasks. Section 4 then details the first of these approaches, which is a transposition of the class actor identification problem as addressed by similar approaches in the state of the art. Section 5 introduces a different task, whose objective is to identify all transactions of a particular actor, by specifically training a change address recognition model for this target actor. Both methods are evaluated in the respective sections. Finally, Section 6 concludes and provides future directions of research.

2. Formalization of the problem and related work

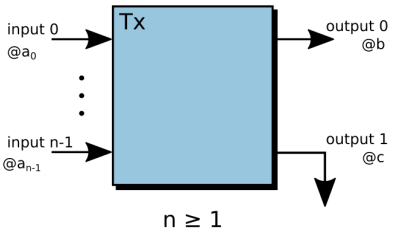
The problem of actor identification and desanonymisation in Bitcoin has multiple aspects, and needs to be properly defined. We call *actor* a person, a group of persons, a company or any entity that owns a set of Bitcoin keys. These cryptographic keys are composed of a *private key* which identifies the owner of the associated *public key*. The public keys are called in the following *Bitcoin addresses*, and are written in clear text in the Blockchain. These addresses are linked to transactions, being either an input which will be debited, or an output which will be credited.

Definition 1 (Transaction link). *In the Blockchain, the inputs and outputs of a transaction are associated with an address and an amount. The input to a transaction is the output of a previous transaction, identified by a transaction number and an output index, and which has not been used as input by another transaction before. We call transaction link the tuple (transaction number, output index), whether it is at the input or at the output of a transaction.*

A typical Bitcoin transaction consists in debiting some accounts in favor of some others. If we except mining transactions, all transactions have one or more inputs, and one or more outputs (see Figure 1a). The sum of all inputs must be equal to the sum of all outputs plus transaction fees. By definition, each input must be the output of a previous transaction, and a transaction output must always be spent in full. This mechanism, known as UTXO (Unspent transaction output) [4], is a common design for Blockchain-based cryptocurrencies. It is important to note that, in such a system, there is no account or global balance for a user: if the user receives two payments on the same Bitcoin address, he owns these two outputs, but they remain distinct, and can therefore be spent separately in different transactions, or jointly if they are included as inputs of the same transaction.



(a) A Bitcoin transaction with n inputs and m outputs.



(b) Illustration of a Bitcoin transaction with change output: @c belongs to the same actor that control addresses $@a_0$ to $@a_{n-1}$.

Figure 1: Illustration of Bitcoin transactions.

By means discussed later in this section, it is possible to identify that a set of addresses belong to a single actor. The associated set of addresses is called an *address aggregate*. By this definition, an aggregate is owned by a single

actor, but a single actor can control multiple address aggregates. Also note that the problem we address in this article is the discovery of address aggregates, and the association of several address aggregates to possibly the same actor. This should not be confused with another popular knowledge discovery task in Bitcoin data, *de-anonymizing actors*, which involves uncovering the actual identity – or nature, or category – of actors.

The *address aggregate discovery* task is typically accomplished with two approaches: the *heuristic of common input addresses* and the *change address discovery*. These two approaches are detailed below.

2.1. The heuristic of common input addresses

The first method for discovering aggregates of addresses takes advantage of the fact that a user having several transaction links, each with an amount less than that of the transaction he wishes to carry out, is led to combine them as input. The *co-inputs heuristic* has already been mentioned as a possibility in Satoshi Nakamoto's white paper describing Bitcoin [1]. It consists of considering that addresses appearing in transaction links, used as inputs to the same transaction, belong to the same actor.

Large aggregates of addresses can thus be found by such chains of co-usages, as was done since the early years of bitcoin [5]. The problem can be effectively solved in practice by searching for the connected components of the graph obtained by representing each address by a node and each co-appearance as input of a same transaction by an edge [6]. In the remaining of this article, we will call this *common input addresses* approach **heuristic H1**. This approach is widely used in the literature, and its effectiveness has been repeatedly confirmed, for example in [7].

Definition 2 (Heuristic H1). *The addresses associated to the input transaction links of a same transaction belong to the same actor.*

The strength of heuristic H1 is to have a precision close to 1: all addresses in the same aggregate actually belong to the same actor, apart from rare exceptions due to *CoinJoin* obfuscation. Under the name of CoinJoin, we regroup the different tools and methods allowing two actors to sign a same transaction, in the objective or tricking heuristic H1. The techniques and usages of these approaches have been studied in the literature [8, 9, 10], showing that, although the phenomenon is real and observed in the Blockchain, it concerns only a minor fraction of the transactions, with little impact on the identification of major actors.

Until now, heuristic H1 is still the standard in the literature, and commonly used as the first step of Bitcoin actor analysis, e.g., [11]. Its main weakness, however, is that its excellent precision is obtained at the cost of a relatively poor recall: many addresses or aggregates belonging to the same actor are grouped in different aggregates.

However, if the entries must be grouped together to collect the quantity of money necessary for the transaction, it is also almost certain that this amount is

strictly greater than that which must be paid to a third party. There is therefore an excess of money that the user returns to himself using what we call a *change address*. Identifying such change addresses makes it possible to group already found address aggregates and thus increase the recall of the obtained result.

2.2. Change address detection

In a Bitcoin transaction, each input must be the output of a previous transaction, and the value received must be used in full. As a result, since the value of the payment an actor wishes to make at time t is unlikely to exactly match any of the transaction links it controls at that time, the actor sends the *change* (the difference between the transaction inputs and the value he wishes to pay) to himself, i.e. to an address belonging to him. For example, in the Figure 1b, the address *@c* belongs to the same actor as the addresses *@a*. The address corresponding to the change is called a *change address*. Other addresses are called *payment addresses*.

Definition 3 (Change and payment addresses). *The addresses associated with the output transaction links of a transaction are either a change address, if it belongs to the same actor as the input transaction link addresses, or a payment address if it belongs to a third party.*

Identifying whether a transaction has a change address, and which of the output addresses is the change address is one of the main challenges of the bitcoin address clustering, known as *change address detection*. When a change address is detected, this address can naturally be added to the address aggregate containing the transaction's input address(es), thus improving the *recall*. The challenge of this operation is that any error can have a catastrophic effect: if the output address belongs to a different actor, then the aggregates of the two actors will be merged, and they will be wrongly considered as a single aggregate.

Various methods have been proposed in the literature to tackle the change detection task. We can classify them into two categories: heuristics and unsupervised learning approaches.

2.2.1. Change address detection heuristics

The first approaches to detect change addresses were based, as H1, on ad-hoc heuristics, leveraging knowledge of the Bitcoin implementation and observation of actors behaviors. Found in the literature in different flavors (e.g., [12, 2]), it uses a system of rules, such as : IF there are exactly two outputs AND that one of the output addresses has never been seen before AND that the other address has been seen before, THEN we identify that address as a typical *single used address*: a new address created specifically to receive a change. This heuristic **H2** is known to succeed in identifying many correct change addresses. However it is also well known that it tends to make critical errors merging major actors together, leading to a collapse of the precision [6]. Other rules can be added to make the detection more permissive or more restrictive. When used in applications, manual corrections are usually done to limit these errors [2]. An

empirical study comparing these heuristics and their variants can be found in [13].

2.2.2. Change address detection using unsupervised machine learning

A purely unsupervised method has been proposed in [6], using community detection (e.g., node clustering) on a graph such that the nodes correspond to the aggregates found by H1 and the edges connect two nodes if they contain addresses found in the inputs and the outputs of the same transaction. The authors propose to use the popular Louvain algorithm for this task to group H1 aggregates that are connected by many edges. The node clusters yielded by the method are considered the new aggregates. They validate their results based on a few hundred manually labeled addresses, and found that the results were more reliable than the ones obtained by H2 heuristic.

Shao et al. [14] propose to use deep neural network and clustering methods to identify change addresses. In input of their framework, they describe each address by some statistical features describing the characteristics of the transaction history in which the address appears. The feature vector associated with an address is based on transaction history and is represented as a variable length sequence where each element is a concatenation of features related to a transaction. The length of the sequence corresponds to the size of the transaction history. From these sequences, they build a representative vector in Euclidean space to ease the comparisons between sequences. To that end, they use Word2Vec algorithm where each transaction in the training set is treated as a word. Skip-gram is about predicting context words appearing near the input word. Combined with Continuous Bag of Words (CBOW) model, which does the opposite of skip-gram by guessing a word based on its context words, it learns dense vectors that preserve the contextual meanings of addresses. Note that this method does not require a labeled dataset, but instead take only into account the similarity of the feature description of addresses. The resulting vectors are used to recognize an unknown address from a given test set using k-Nearest Neighbors (k-NN) method. They also cluster the addresses using K-Means on vectors pre-processed by PCA.

3. ML for change address identification via *a posteriori* interpretation

Our contribution is to learn a predictive model to recognize that a transaction output corresponds to a change address. As in state-of-the-art methods, we use descriptors calculated from the Blockchain. The novelty of our work lies in the fact that we propose an original approach to generate a large quantity of train and test examples based on ground truth built from an *a posteriori* interpretation scheme. In this section, we first describe the data we used for the analysis. We then introduce our machine learning approach, which consists in defining a supervised learning task on this dataset by introducing labels and features.

3.1. Dataset

Since Blockchain data is available to everyone, we collected all data using a Bitcoin node, following a process comparable to [15]. We used data from block 0 (January 3, 2009) to block 667542 (January 25, 2021), representing a total of approximately 600 million transactions. For each transaction, we have access to the input and output addresses, the amounts associated with each of them, the transaction fees, the block and its timestamp.

We have augmented this information from an external source, provided by the `walletexplorer.com` website, which provides the list of known addresses for a variety of popular actors in the Bitcoin economy. First, this allows us to know who particular actors are, helping us to identify some interesting actors such as Exchange platforms. Second, it makes it possible to identify some known weaknesses of the H1 heuristic, especially when a same actor is known to possess multiple H1 aggregates.

3.2. Two Machine Learning tasks

We consider in the following the use of our supervised machine learning scheme to solve two different problems:

Automatic discovery of enhanced address aggregates. This task (cf. Section 4) aims to predict change addresses using an original approach to generate a large amount of training examples and ground truth examples, thanks to an interpretation scheme *a posteriori*.

User specialized change detection. In some cases, we are not interested in discovering all possible aggregates, but rather we have one or more aggregates of interest (typically discovered using H1), and we want to improve it by detecting change addresses related to these addresses. We propose a method (see Section 5) to perform this detection by specializing the learning model on the actor of interest, and show the relevance of this approach.

3.3. Label assignation

The methods we propose are the first to use supervised machine learning do detect change outputs. The challenge of the machine learning approach is that there are no labels in the original Bitcoin data to know if an output is or not a change output. To overcome this limit, we propose to add such labels to the dataset, by using a two step approach. In a first step, we apply heuristic H1 to discover a first level of address aggregates.

Definition 4 (H1 aggregates). *Considering a set of transactions $\mathcal{T}$, the set of H1 address aggregates constructed from $\mathcal{T}$ and denoted by $\mathcal{A} = \{A_1, \ldots, A_p\}$ is such that $\forall i = 1 \ldots p$, and $\forall$ two distinct addresses $a, b \in A_i, \exists$ a sequence of pairs of addresses $P = (a, c_1), (c_1, c_2), \ldots, (c_n, b)$ such as $\forall (p, q) \in P$, $\exists t \in \mathcal{T}$ such that $p, q \in t.input$.*

In the second step we use this information to attribute partial labeling to the address outputs. In practice, for each transaction output, if we observe that one of the output addresses belongs to the same aggregate as the transaction input addresses, then we label that output as the change, while the other outputs are labeled as non-change. We call this approach **H1-labeling**.

Definition 5 (H1-labeling). *Considering H1 aggregates* $\mathcal{A} = \{A_1, \ldots, A_p\}$, *if there exists a transaction* $t \in \mathcal{T}$ *with* $t.input \subseteq A_i$ *and* $o \in t.output$ *then*

$$\begin{cases} \text{label(o)} = \text{change } \textit{if } \text{o} \in \text{A}_\text{i} \\ \text{label(o)} = \text{payment } \textit{otherwise} \end{cases}$$

H1-labeling has two limits: 1) When an output is labeled as payment, we do not know if it is in fact a genuine payment, or just a limit of the H1 heuristic, which failed to associate the address in output with the aggregate in input. 2) The method works only *a posteriori*, i.e., H1 heuristic is based on the principle that users eventually tend to combine their addresses in input of the same transaction. But such combination might occur days, months or years after the address was used for the first time. We propose different mitigation for these issues depending on the task.

3.4. Transaction output features

For each transaction output, we compute several features, that we use as descriptive features in the supervised machine learning approach. These features are presented in Table 1.

4. Automatic Discovery of Improved Address Aggregates

The H1 and H2 heuristics, as well as methods such as [6, 14], have a common objective which is to discover aggregates of addresses supposed to belong to the same actor. One possible application of change address detection is to solve this problem. More formally, if we consider the graph defined such that each node is an aggregate discovered by H1, with edges corresponding to change addresses detected between these nodes, then the improved aggregates are defined by the connected components of the graph. In other words, if an aggregate sends the change of at least one of its transactions to another aggregate, then the two aggregates must in fact be considered as one, belonging to the same actor.

To improve these address aggregates using change detection, the method we propose consists in creating an induced graph in which the nodes are H1 aggregates, and the edges correspond to the changes detected between them. Improved aggregates are defined as connected components of this induced graph.

4.1. Ground truth aggregates definition using a posteriori approach and external source

While H1-labeling allows to constitute a large training corpus, it raises the question of the ground truth used for validation. The usual machine learning

N°	Description
1	Value associated to the considered output in Satoshi[1].
2	The number of prior usages of this address as an output.
3	Total value of the inputs of the transaction (Satoshi).
4	Total value of the output of the transaction (Satoshi).
5	Number of non-zero digits among the 8 last digits of the value (Satoshi) of the considered. It captures the property of being or not a round number.
6	The number of decimals of the Bitcoin value.
7	The fraction of the considered output value compared to all outputs.
8	The index of the output.
9	The number of inputs of the transaction.
10	The fees paid for this transaction.
11	The output value (Satoshi)
12	The output value (USD)
13	A Boolean indicating if the remaining of the output value, after removing all zeros equals '1'
14	A Boolean indicating if the remaining of the output value, after removing all zeros equals '5'
15	A Boolean indicating if the remaining of the output value, after removing all zeros equals '2', '3', '4', '6', '7', '8', '9', '15', '25', '55' or '99'
16	A Boolean being True if this output is smaller than the smallest input minus transaction fee, False otherwise. In principle, the change value should be smaller than the smallest input, otherwise this input was not needed to do the transaction;
17	Shannon entropy of values distribution (for input and for output). Such diversity indexes allow to differentiate cases were most of the values come from a single input from these were it is spread more evenly;
18	Shannon entropy of distribution of different addresses (for input and for output).
19	The year.
20	The month.
21	The day in the month.
22	The timestamp
23	The day of the week.

Table 1: Transaction output features.

approach would consist in splitting all outputs in two parts, a training set and a test set, and evaluate the quality of the change recognition in the test set. However, this approach is not satisfying, for two reasons: 1) At best, it proves only that the machine learning approach can do *as well as* the H1 heuristic, since all positive change examples have been labeled using H1 itself, 2) A very efficient

[1]A Satoshi is the smallest unit of a Bitcoin, worth one hundred millionth (10^{-8}) of a Bitcoin.

method that would recognize all change addresses, including these missed by H1, would in fact have a *lower* score than a method only able to reproduce H1 change recognition. To circumvent these weaknesses, we propose two solutions.

A posteriori test set. To mitigate the first problem, we define our ground truth *a posteriori*. More precisely, we split our dataset in two parts: the studied dataset **D-2017**, composed of transactions up to bloc 501950 (December 31, 2017), and the *a posteriori* ground truth dataset **D-2021**, composed of all transactions up to the end of the collected dataset. We use D-2017 for training and for evaluation, and we use D-2021 to compute the ground truth aggregates, that we call **H1-AP** for *a posteriori*. The principle is that H1 applied to D-2017 will incorrectly label some changes as payment, and thus several different aggregates according to H1 will in fact be merged into a single aggregate in our ground truth, thanks to the additional information found in D-2021.

External ground truth. To mitigate the second problem, we improve our ground truth using an external source. We used data extracted from the website WalletExplorer[2], commonly used in the literature to recognize known actors (e.g., [16, 17]). The website provides for a few hundred aggregates the name of the actor to which this aggregate corresponds to. Although the exact details of the process are not known, this information is said to be obtained through manual collection: since it is enough to know the identity of one address of an aggregate to label the aggregate, one transaction with a known actor (Exchange platform, Gambling service, etc.) usually allows to label its whole aggregate. The information we are interested in from this website is that, for several known entities, the website provides several associated aggregates. We leverage this information to improve our ground truth: instead of using H1 aggregates in our ground truth, we create Ground Truth aggregates **GT-A** by merging H1 aggregates found on D-2021 according to the actors defined by WalletExlorer. We chose to split the dataset in 2017 because 1) we want to have enough data after the end the study dataset to discover change addresses, 2) WalletExplorer is known to be less reliable for actors after 2017 (missing information). The ground truth and study datasets are summarized in Figure 2. The types of merges that we can observe, and the consequence on our evaluation score are summarized in Figure 3.

4.2. Training and testing protocols

To constitute the training set, we have retained only the transactions composed of two outputs, one being a change and the other not, according to H1-D-2017. These transactions are the most likely to be correctly labeled because Bitcoin's most common behavior is to make a payment in one output and send the rest to a change address. It also has the advantage of maintaining a balanced dataset between changes and payment training examples.

[2] `https://www.walletexplorer.com`

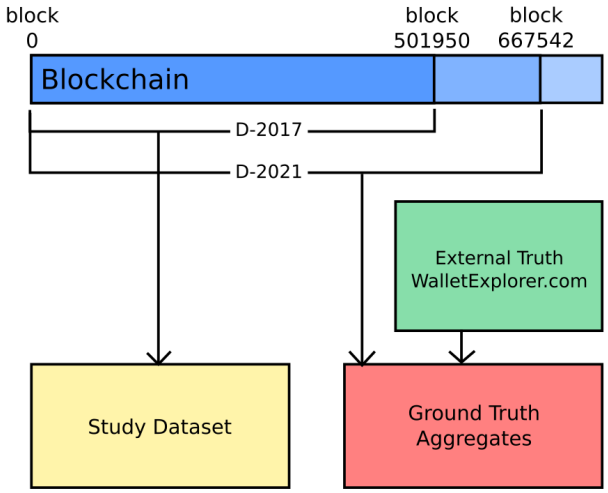


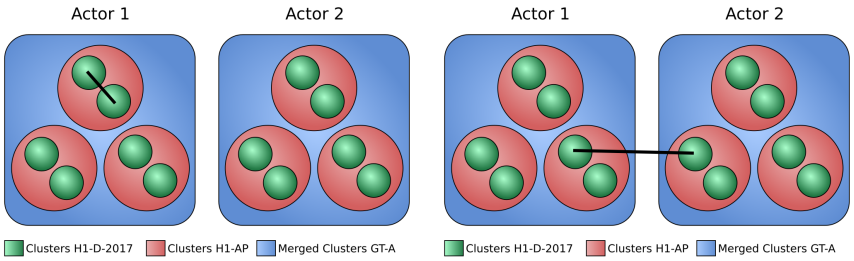
Figure 2: Generation of the studied dataset **D-2017** and of the Ground Truth Aggregates **GT-A** based on D-2021 and WalletExplorer data.

Our test set (or rather predict set in our case, since it is used for making predictions but not directly for evaluation) is conversely composed of transactions having exactly two outputs, none of them being labeled as change according to H1-D-2017. These transactions are the most likely to have a missing change that we could detect. We ignore transactions with more than two outputs, as these may correspond to different and more complex cases. They could be addressed in future work.

We restricted our analysis to six actors[3] from WalletExplorer, for performance and reliability of the ground truth. We have collected all transactions such as the sender of the transaction is among the chosen actors. These actors have been chosen according to their size (number of transactions), for having multiple known aggregates, for their diversity (Mining Pool, Exchanges, Gambling services), and for their period of activity, compatible with our a posteriori approach (activity before and, if possible, after 2017). Having six actors only is a limit, but reflect our choice of having less data of greater quality, instead of taking the risk of evaluating our method by comparing it with a biased ground truth composed of a large number of unknown aggregates.

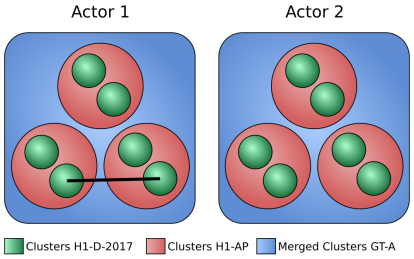
We finally have at our disposal 520 578 addresses belonging to the six chosen actors, 354 006 training examples (outputs addresses, 50% change outputs, 50% paying outputs), and 2 557 002 transaction outputs to evaluate as being or not

[3]Bter.com, PrimeDice.com, BitcoinVideoCasino.com, FaucetBOX.com, BTCCPool, BitZino.com.



(a) Merging of two H1-D-2017 aggregates belonging to a same H1-AP aggregate. Beneficial Merge.

(b) Merging of two aggregates H1-D-2017 belonging to two different actors according to WalletExplorer. Detrimental Merge.



(c) Merging of two H1-D-2017 aggregates belonging to two different H1-AP aggregates belonging to the same ground truth. Beneficial Merge.

Figure 3: Different types of possible merges.

change addresses.

4.3. Machine learning models

We chose to use a decision tree for its ability to learn nonlinear relationships, its simplicity and its interpretability. More expressive methods such as Random forest or XGBoost could have been used interchangeably, but the relative performance of different classifiers is beyond the scope of this article. A *Grid Search* approach was used to optimize the maximum number of leaves and the minimum number of items per leave, maximizing accuracy through cross-validation. We used the implementation available in scikit-learn[4].

To decide if an output should be considered a change output, we do not directly use the class provided by the decision tree to avoid the risk of **catastrophic merge**. Indeed, a major pitfall of aggregate discovery is that, since a single occurrence of one aggregate being identified as receiving the change of another is enough to merge the two in a single aggregate, a single false positive (a payment being wrongly identified as a change) can have a catastrophic ef-

[4] `https://scikit-learn.org/`

fect on the final partition, since two potentially large distinct aggregates would merge. Conversely, a false negative (a change output being wrongly assumed to be a payment) is less critical, in particular since a different change transaction between the two same clusters can be rightfully detected and will be enough to correct this missed detection.

We propose and test three variants of change detection methods to minimize catastrophic merges:

M1 – Classification with variable confidence threshold. The reference approach is to use a variable threshold on the confidence probability of the classification. By imposing a high confidence value, we detect fewer change transactions, but reduce the probability of a catastrophic merge. The right threshold is the best compromise between increasing false negatives and decreasing false positives.

M2 – Limit to one change per transaction. As mentioned when describing the train and test sets, we focus on transactions that have two outputs. It is known that these transactions tend to have a single payment and a single change output. Therefore, if two outputs are classified as a change with confidence above the threshold, and there is a significant difference ($> 1\%$) between the two, only the one with higher confidence is classified as a change. If there is less than 1% difference, we classify both as payment.

M3 – Allowing multiple change outputs for aggregate merges. To avoid that a single false positive merges two aggregates, we add the constraint that several change outputs must be observed between two H1 aggregates to merge them. In practice, we have defined improved aggregate detection as the search for connected components of the induced graph in which the nodes are H1 aggregates and the edges (u_1, u_2) represent the existence of a transaction with an address of u_1 as input and an address of u_2 as change output. In this variant, we add an edge in this induced graph only if we observe $x > k$ such transactions between the two H1 aggregates, with k a threshold. In the following experiments, we use $k = 2$, a sufficient limit to observe significant changes.

4.4. Results

To evaluate the performance of our approach, we compare aggregates obtained by the different variants with our ground truth. We use the most common cluster comparison metrics, namely *Homogeneity*, *Completeness*, *v-score/NMI*, *aNMI* and *Rand Index*.

We first report the *Homogeneity* scores obtained using H1-AP (Table 2) or GT-A (Table 3) as ground truth. We observe that the M3 method obtains a value below 1 with H1-AP, while it obtains a score of 1 using a threshold of 0.8 or more using GT-A as ground truth. This clearly confirms the validity and relevance of our original validation approach: the M3 variant merges some aggregates which are considered different using only the information from the Blockchain (H1 a posteriori), but which are recognized as correct when they are validated using external ground truth. It thus confirms that the process is able

to recognize change addresses that could not have been detected simply by the H1 heuristic. This has never been shown before. Note that H1 alone also has a homogeneity score of 1 because it never wrongly merges aggregates.

Since observations are similar for other scores, we will report only the comparison with the GT-A ground truth.

Heuristic H1	**1.000**		
Threshold	M1	M2	M3
0.7	0.593	0.903	0.780
0.75	0.593	0.903	0.799
0.8	0.644	0.903	0.948
0.85	0.644	0.903	0.948
0.9	0.799	0.903	0.948
0.95	0.920	**1.000**	0.978

Table 2: Homogeneity Score - Ground truth H1 on D-2021 (H1-AP).

Heuristic H1	**1.000**		
Threshold	M1	M2	M3
0.7	0.562	0.890	0.732
0.75	0.562	0.890	0.761
0.8	0.623	0.890	**1.000**
0.85	0.623	0.890	**1.000**
0.9	0.761	0.890	**1.000**
0.95	0.908	**1.000**	**1.000**

Table 3: Homogeneity Score - Ground truth enriched with actors from WalletExplorer (GT-A).

The *Completeness* measures the gain obtained by rightfully merging the H1-D-2017 aggregates. We observe in Table 4a that 1) the M1 method fails to obtain scores higher than the reference H1 score, 2) M2 only succeeds for the maximum threshold chosen, and 3) M3 makes it possible to improve significantly relatively to the baseline.

To confirm these results, we use three commonly scores that synthesizes both aspects captured by Completeness and Homogeneity. The first one is the v-score (or NMI), which is defined as the harmonic mean of Completeness and Homogeneity. Given the previous observations, it naturally confirms the superiority of the results of method M3 (Table 4b).

To make these results more robust, we use two randomly adjusted scores commonly used in clustering and community detection assessment: the aNMI (Table 4c) (random-adjusted version of the NMI), and the Adjusted Rand Index (ARI) in Table 4d. The two scores confirm the observation that the results obtained with the M3 method are superior both to the other methods and to the reference heuristic H1.

Heuristic H1	0.626		
Threshold	M1	M2	M3
0.7	0.593	0.617	0.588
0.75	0.593	0.617	0.597
0.8	0.606	0.617	**0.661**
0.85	0.606	0.617	**0.661**
0.9	0.597	0.617	**0.661**
0.95	0.618	0.627	0.641

(a) Completeness Score - GT-A Ground truth.

Heuristic H1	0.770		
Threshold	M1	M2	M3
0.7	0.577	0.729	0.652
0.75	0.577	0.729	0.669
0.8	0.614	0.729	**0.796**
0.85	0.614	0.729	**0.796**
0.9	0.669	0.729	**0.796**
0.95	0.736	0.770	0.781

(b) V-Score/NMI - GT-A Ground truth.

Heuristic H1	0.770		
Threshold	M1	M2	M3
0.7	0.577	0.729	0.652
0.75	0.577	0.729	0.669
0.8	0.614	0.729	**0.796**
0.85	0.614	0.729	**0.796**
0.9	0.669	0.729	**0.796**
0.95	0.736	0.770	0.781

(c) aNMI - GT-A Ground truth.

Heuristic H1	0.481		
Threshold	M1	M2	M3
0.7	0.286	0.432	0.345
0.75	0.286	0.432	0.351
0.8	0.299	0.432	**0.532**
0.85	0.299	0.432	**0.532**
0.9	0.351	0.432	**0.532**
0.95	0.446	0.481	0.501

(d) Rand Index - GT-A Ground truth.

Table 4: Scores obtained with the different scores. We observe that the M3 method yield the best scores overall according to all metrics, both among tested variants and compared to H1 baseline, in most contexts.

5. User specialized change detection

In the previous section, we have proposed an original supervised machine learning based method to solve a common problem in Bitcoin: the detection of

address aggregates. The problem was posed in a way comparable to the usual task tackled by state-of-the-art methods, i.e., the discovery of a set of address aggregates from a set of transactions.

However, we observed that in the literature, previous methods have in general not been reused after their publication, later articles on Bitcoin user analysis still relying on H1 heuristic to identify address aggregates considered as actors. From our own experience, we think that there are two main reasons for this mitigated results:

1. The difficulty to apply and reproduce these results: given the scale of the Bitcoin data, even computing the simple H1 heuristic is a challenging task. Computing on the whole dataset a much more complex approach, requiring to compute various features, and/or computationally expensive models such as Deep Neural Networks is an even greater challenge.
2. Despite the safeguards added to most methods, including ours, the risk of catastrophic merges (and in practice the little information we have about unwanted merges occurring on address aggregates other than the large known ones) makes the use of these methods a risky choice when analyzing behaviors in the Bitcoin Blockchain.

This analysis does not contradict the good results obtained by these methods, as demonstrated in the previous section. However, we think that change detection would be more actionable for a slightly different task that we define here: the change detection of individual actors. We argue that in various applications, researchers and practitioners are mostly interested in analyzing one particular actor, or a subset of actors of interest (e.g., malicious actors [18, 19], Mining Pools [20], Major exchanges[21], etc.). In this section, we investigate how our supervised machine learning approach, contrary to unsupervised ones, could be used for the change detection on a particular actor, with the objective to better detecting the activity of this actor in particular.

5.1. Training and testing protocols

To identify the change addresses of a target actor, we first build a dataset of all the transactions from which this actor has addresses used as input. We then compute the same features as defined in Section 3.4 for each output of these transactions. Contrary to the previous section, we do not restrict ourselves to transactions with two outputs, since we are less concerned about catastrophic merges, that would be more easily identified with a single actor than with the millions of aggregates obtained by an aggregate detection on the whole Bitcoin dataset. Furthermore, we are using all the transactions with at least one known change output, instead of just one, for the training and testing steps. This means that all the transactions having multiples known change outputs that were rejected in the previous section are now present in the dataset.

In addition to the features used in the first part of this work, we used the number of outputs. This feature was irrelevant before but applicable here, as now we are using transactions with any number of outputs.

To build our training and testing set, we can now rely on the commonly used supervised machine learning approach: we randomly split all outputs in two sets: the train and test sets. The quality evaluation process can be framed as any classification evaluation problem, using the ROC-AUC score. In this scenario, this score can be interpreted as the probability for a randomly chosen change output in the test set to have a higher confidence probability to be a change output than a randomly chosen payment output (according to the trained classifier). A score of 0.5 thus means that the classifier is doing no-better than random prediction, while a score of 1 means that all change addresses have a higher confidence score (probability of class *change*) than all payment addresses.

We have selected different Walletexplorer actors, from different categories and sufficiently active in terms of number of transactions to have statistically significant results. For each of them, we trained a model to recognize their change outputs based on their own transactions, using a division of two-thirds for training and one-third for testing. These models are called *targeted models* by opposition to a model learnt from a random sample of 500 000 transactions, called hereafter *un-targeted model*. The objective of this comparison is to see if it is more reliable to train with less data on a single actor, or to have more training data to better generalize the definition of a change address.

5.2. Results

ROC-AUC	
Dataset	Score
BTCC.com	0.998672
BitPay.com	0.998158
CoinRoyale.com	0.996111
Bter.com	0.995005
Bitfinex.com	0.992843
Banx.io	0.992029
Cryptsy.com	0.991124
PrimeDice.com	0.981156
500,000 random transactions	0.931392

Table 5: ROC-AUC score for each actor and random transaction datasets.

The ROC-AUC scores for change detection for the targeted models and for the un-targeted one are shown in Table 5. We can observe that the method yields very convincing results in most cases, confirming that a machine learning approach can be learnt to recognize change outputs.

Another obvious point is the difference between the actor's targeted scores and the random one. All scores based on an actor activity scores are above 0.98, while the random transaction one is a little above 0.93. This confirms that the

results obtained with previous methods are certainly sub-optimal and could be widely improved by using targeted learning.

Additionally, we generated a heatmap (see Figure 4) based on the calculated importance of each feature for each experiment, as well as a visualization of the learned decision trees with depth limited to 3, to facilitate their interpretation (Figures 5, 6 and 7).

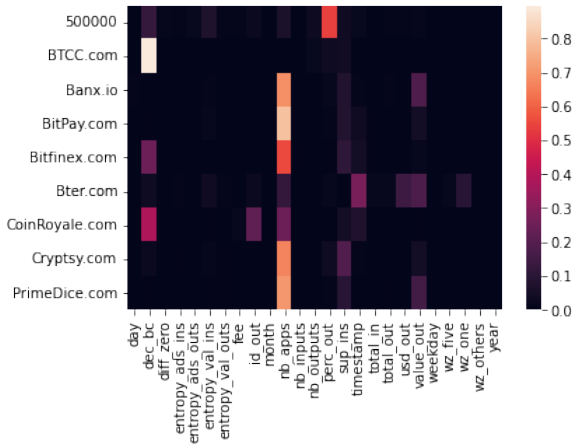


Figure 4: Heatmap of feature importance for different actors (targeted datasets) and for the transactions collected randomly (un-targeted dataset). We can observe clear differences in feature importance both between one or more targeted datasets and between targeted and un-targeted datasets.

When observing the feature importance heat-map (Figure 4) we can notice some interesting results:

- Different actors seem to manage differently their transactions: the features importance vary from one actor to another;
- One of the features (the number of previous appearances of the output address) is very important in many of the datasets. This element was known from previous heuristics such as H2, but we can see here that it is not reliable enough to be used as a single criteria, and that its importance depends on the actor;
- Contrary to the previous observation, *BTCC.com* does not rely on the number of previous appearances of the output address at all. Another characteristic is very important: the number of decimals of the output value in BTC;
- The timestamp feature is the most important one for the *Bter.com* actor. This can probably be explained by a change in transactions management at some time;

- The most important feature for the un-targeted model (Perc_out) seems to be of little importance for targeted models.

Beyond this global overview of the results, we can analyze the learned decision trees to study some actor's models individually. In this way, we can find other important information, as well as understand some of the previous observations. In the following trees, a darker color tone indicates that the node contains mostly elements of one class (orange for 0, blue for 1). On the contrary, lighter colored nodes have more mixed examples. Some important findings are:

- When examining *Banx.io* tree (Figure 5), we can learn that every address used as a change output is a newly generated address;
- The *BTCC.com* tree (Figure 6) shows us the importance of the decimal digits in the output value, as at the first branch it can separate many payments from change outputs. This shows us that most of the payments have more "rounded" values;
- Looking at the non-targeted model's tree (Figure 7), we can see right away that it has more trouble classifying the outputs, as we found more light colored nodes than for the others datasets. The absence of a common way in managing the transactions in the dataset makes it more difficult to classify the outputs.

These results validate our previous observation that the un-targeted model is not able to correctly classify the outputs. On the contrary, models trained with a specific actor approach obtain much higher scores.

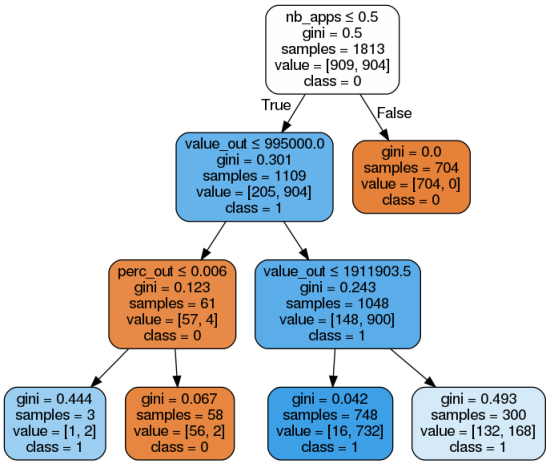


Figure 5: Banx.io decision tree.

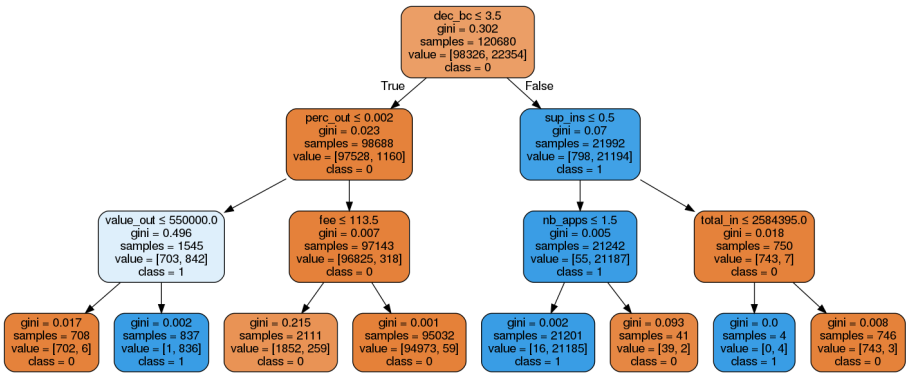


Figure 6: BTCC.com decision tree.

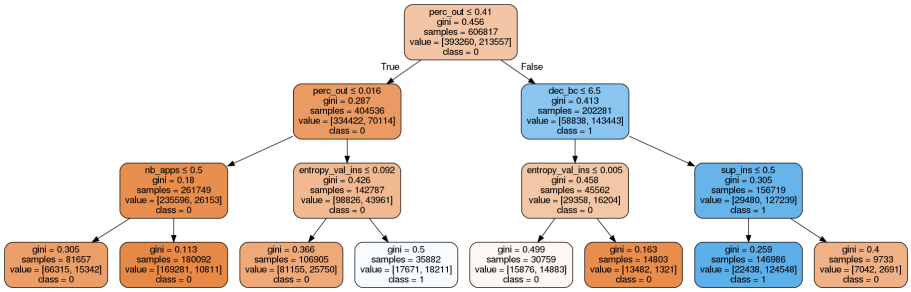


Figure 7: 500,000 random transaction dataset decision tree.

6. Conclusion

In this article, we propose a novel approach for identifying Bitcoin actors using supervised machine learning. We come up with an original procedure for building training set, and a validation with a much larger number of addresses than the previous methods. We have proposed two different uses: 1) one comparable to the task addressed in the state of the art, in which we provide a large set of transactions as input, and provide address aggregates as output, 2) the other restricted to the discovery of change addresses of a single target actor. We have shown the interest of the two methods with an empirical evaluation.

The key results of our paper are the following. First, unlike previous approaches, we used external sources to show that the aggregates we found are not only able to rediscover the same results as H1 heuristic, but is really improving on this result. Previous works were either proposing H1 improvement, but without qualitative evaluation (e.g., H2 heuristics as in [2]), or validating only using H1 or manual validation (e.g., [6, 14]. Second, we have shown that training a change detection model for a single actor is more efficient that training a model for recognition of change addresses in general. In our opinion, such a targeted application is more actionable than the traditional *all-in-one* approach.

Of course, our work has limits. First, we validated the first approach only with 6 actors. Although there is no restriction on the number of actors we could detect, we restricted ourselves to these actors due to the limitations of ground truth, as we would not have had any relevant ground-truth for an evaluation on thousdands of aggregates, which, in our opinion, is unreliable: if the ground truth is built only on H1, then showing that the output of the algorithm is similar to the ground truth is not a satisfactory approach. For the single-actor target method, we have proven that it is more efficient to detect change outputs than to train on the entire dataset, but we have not offered a comprehensive workflow to automatically discover all addresses of an actor according to this change detection: to do so, the method must be applied recursively after having discovered new addresses of the same actor, and, as was done with the M3 variant of the first method, add safeguards to avoid the risk of catastrophic merges. However, we believe that such a method is beyond the scope of this article.

We think nevertheless that the work proposed here clearly demonstrates the feasibility and the interest of supervised machine learning for Bitcoin actor identification.

References

[1] S. Nakamoto, Open source p2p money (2008).
URL `https://bitcoin.org/`

[2] S. Meiklejohn, M. Pomarole, G. Jordan, K. Levchenko, D. McCoy, G. M. Voelker, S. Savage, A fistful of bitcoins: characterizing payments among men with no names, in: Conference on Internet measurement, 2013, pp. 127–140.

[3] D. Kondor, M. Posfai, I. Csabai, G. Vattay, Do the rich get richer? an empirical analysis of the bitcoin transaction network, PloS one 9 (2) (2014) e86197.

[4] S. Delgado-Segura, C. Perez-Sola, G. Navarro-Arribas, J. Herrera-Joancomarti, Analysis of the bitcoin utxo set, in: International Conference on Financial Cryptography and Data Security, Springer, 2018, pp. 78–91.

[5] F. Reid, M. Harrigan, An analysis of anonymity in the bitcoin system, in: Security and privacy in social networks, Springer, 2013, pp. 197–223.

[6] R. Cazabet, R. Baccour, M. Latapy, Tracking bitcoin users activity using community detection on a network of weak signals, in: Complex Networks and Applications, 2018, pp. 166–177.

[7] M. Harrigan, C. Fretter, The unreasonable effectiveness of address clustering, in: Intl IEEE Conferences on Ubiquitous Intelligence Computing, 2016, pp. 368–373. doi:10.1109/UIC-ATC-ScalCom-CBDCom-IoP-SmartWorld.2016.0071.

[8] M. Möser, R. Böhme, The price of anonymity: empirical evidence from a market for bitcoin anonymization, Journal of Cybersecurity 3 (2) (2017) 127–135.

[9] M. Möser, R. Böhme, Anonymous alone? measuring bitcoin's second-generation anonymization techniques, in: 2017 IEEE European Symposium on Security and Privacy Workshops (EuroS&PW), IEEE, 2017, pp. 32–41.

[10] M. Harrigan, C. Fretter, The unreasonable effectiveness of address clustering, in: 2016 Intl IEEE Conferences on 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), IEEE, 2016, pp. 368–373.

[11] D. Kondor, N. Bulatovic, J. Steger, I. Csabai, G. Vattay, The rich still get richer: Empirical comparison of preferential attachment via linking statistics in bitcoin and ethereum, arXiv preprint arXiv:2102.12064 (2021).

[12] E. Androulaki, G. O. Karame, M. Roeschlin, T. Scherer, S. Capkun, Evaluating user privacy in bitcoin, in: Financial Cryptography and Data Security, 2013, pp. 34–51.

[13] J. D. Nick, Data-driven de-anonymization in bitcoin, Master's thesis, ETH-Zürich (2015).

[14] W. Shao, H. Li, M. Chen, C. Jia, C. Liu, Z. Wang, Identifying bitcoin users using deep neural network, in: Int. Conf. on Alg. and Arch. for Parallel Proces., 2018, pp. 178–192.

[15] J. A. Emery, M. Latapy, Full bitcoin blockchain data made easy, in: Advances in Social Networks Analysis and Mining, 2021.

[16] D. Ermilov, M. Panov, Y. Yanovich, Automatic bitcoin address clustering, in: IEEE International Conference on Machine Learning and Applications (ICMLA), IEEE, 2017, pp. 461–466.

[17] M. Möser, A. Narayanan, Resurrecting address clustering in bitcoin, arXiv preprint arXiv:2107.05749 (2021).

[18] A. Yazdinejad, H. HaddadPajouh, A. Dehghantanha, R. M. Parizi, G. Srivastava, M.-Y. Chen, Cryptocurrency malware hunting: A deep recurrent neural network approach, Applied Soft Computing 96 (2020) 106630.

[19] S. Dalal, Z. Wang, S. Sabharwal, Identifying ransomware actors in the bitcoin network, arXiv preprint arXiv:2108.13807 (2021).

[20] N. Tovanich, N. Soulié, N. Heulot, P. Isenberg, An empirical analysis of pool hopping behavior in the bitcoin blockchain, in: 2021 IEEE International Conference on Blockchain and Cryptocurrency (ICBC), IEEE, 2021, pp. 1–9.

[21] M. Jourdan, S. Blandin, L. Wynter, P. Deshpande, Characterizing entities in the bitcoin blockchain, in: 2018 IEEE international conference on data mining workshops (ICDMW), IEEE, 2018, pp. 55–62.