Address Clustering Heuristics for Ethereum

Chapter · July 2020

DOI: 10.1007/978-3-030-51280-4_33

Friedhelm Victor
TRM Labs
22 PUBLICATIONS **823** CITATIONS

Address clustering heuristics for Ethereum

Friedhelm Victor[0000−0001−8329−3133]

Technical University of Berlin, Straße des 17. Juni 135, 10623 Berlin, Germany
friedhelm.victor@tu-berlin.de

Abstract. For many years, address clustering for the identification of entities has been the basis for a variety of graph-based investigations of the Bitcoin blockchain and its derivatives. Especially in the field of fraud detection it has proven to be useful. With the popularization and increasing use of alternative blockchains, the question arises how to recognize entities in these new systems. Currently, there are no heuristics that can directly be applied to Ethereum's account balance model. This drawback also applies to other smart contract platforms like EOS or NEO, for which previous transaction network analyses have been limited to address graphs. In this paper, we show how addresses can be clustered in Ethereum, yielding entities that are likely in control of multiple addresses. We propose heuristics that exploit patterns related to deposit addresses, multiple participation in airdrops and token authorization mechanisms. We quantify the applicability of each individual heuristic over the first 4 years of the Ethereum blockchain and illustrate identified entities in a sample token network. Our results show that we can cluster 17.9% of all active externally owned account addresses, indicating that there are more than 340,000 entities that are likely in control of multiple addresses. Comparing the heuristics, we conclude that the deposit address heuristic is currently the most effective approach.

Keywords: Blockchain · Accounts · Ethereum · Network Analysis.

1 Introduction

Since the introduction and popularization of Bitcoin [22] in 2009, blockchain and cryptocurrency analysis has gained a foothold in science as well as in business. A number of established companies and startups are investigating blockchain data for purposes related to cryptoasset assessment, insights for financial institutions and the support of law enforcement [7]. In most of these networks, an individual can participate with several pseudonymous addresses, the creation of which is virtually cost-free. For outsiders, it is not necessarily obvious that they belong to the same entity. Cryptocurrencies are also used for criminal activities where the perpetrators hope to cover up their traces. To hide their identity, extortionists do not use the same address for every victim [25], and money laundering is carried out using a large number of addresses [21]. In blockchain-based voting systems, where currency balance determines voting power, equality could be faked when a user distributes their assets to multiple addresses. Therefore, a core component of many investigations is the detection of single entities that interact through multiple addresses. To detect such entities, a number of address clustering heuristics have been proposed for Bitcoin, that have also been reused in derivatives like Litecoin and ZCash [14, 11]. Most of the existing heuristics are based on Bitcoin's UTXO model which allows a single transaction to have multiple inputs and outputs. However, a growing number of blockchain implementations have not adopted this model. A prominent example is Ethereum, which instead employs an account model where a regular transaction has one source and one destination account address. Apart from Ethereum, this account model is also present in other popular smart contract platforms such as EOS or NEO. Existing address clustering heuristics based on multiple inputs or outputs cannot be used for transactions with single inputs and outputs.

However, performing entity identification on account model blockchains such as Ethereum is of great interest, as it forms the basis for entity graph analysis, which allows for better assessment of network properties related to usage, wealth distribution and fraudulent activity. For example, Ether payments are also accepted in darknet marketplaces [15], and ponzi schemes exist through smart contracts [2, 6]. It is likely that money laundering also exist on Ethereum, and the emergence of decentralized finance services like on-chain derivatives, loans and the use of decentralized exchanges are likely targets for manipulation. The underlying schemes may rely on the idea of creating the illusion of interaction between supposedly distinct participants.

Our contribution. In this work, we propose several novel address clustering heuristics for Ethereum's account model, derived from the analysis of phenomena surrounding deposit addresses, multiple participation in airdrops and self-authorization. We explore each heuristic in detail and quantify their applicability over time. Our results show that we can cluster 17.9% of all active externally owned account addresses, indicating that there are more than 340,000 entities likely in control of multiple addresses. Comparing the heuristics, we conclude that the deposit address heuristic is currently the most effective approach. To allow for the heuristics to be used in practice, we published an implementation of them on GitHub[1].

The remainder of this paper is structured as follows: In Sections 2 and 3, we provide an overview of the background on Ethereum, Tokens and Airdrops, as well as existing research results on address clustering for entity identification. In Section 4, we describe the data that forms the basis of our analyses and provide a set of high-level statistics of our data set. In Section 5 we study the heuristics of exchange deposit address reuse, airdrop multi-participation and token transfer authorization. We analyze the heuristics over time in Section 6, before discussing (Section 7) and summarizing the results of our paper in Section 8.

[1] https://github.com/etherclust/etherclust

2 Background

After the creation of Bitcoin in 2009 [22], many alternative blockchains and associated cryptocurrencies have been proposed. By market capitalization in 2019, Ethereum [33] is the second most popular blockchain after Bitcoin. Both systems are open-source, public, distributed and rely on a Proof-of-Work-based consensus algorithm. To interact with the transaction network, users typically use a wallet software. They can create and manage multiple public/private key pairs, which can be used to sign transactions. For each key pair, an address is derived from the public key, serving as a pseudonymous identifier.

While both Bitcoin and Ethereum share the basic notion of an address, they differ in their abstraction of currency transfer. In Bitcoin, each transaction on the ledger must have one or multiple Unspent Transaction Output (UTXO) as input, which may be used by the corresponding holders of the private keys. Each UTXO contains a certain amount of Bitcoin. With each transaction, the inputs are spent, and the outputs are new UTXO.

In Ethereum, each regular transaction has one sender and one receiver account address. An account can either be an Externally Owned Account (EOA), where the private key is owned by an external user, or a smart contract account. Smart contract accounts contain executable code and don't have a private key. Their address is determined by the deployer's address and nonce, and the code can be executed by sending transactions to them, optionally with parameters.

2.1 Tokens

Smart contracts are frequently used to create token systems. A token can represent a variety of transferable and countable goods such as votes, memberships, loyalty points, shares or other utility [3]. To create a new token that is compatible with popular wallet software, developers can follow implementation standards such as ERC20[2] for fungible tokens, or ERC721[3] for non-fungible tokens. Similar standards exist on other smart contract platforms.

2.2 ICOs, Bounties and Airdrops

Startups have embraced the idea of tokens in order to raise funds in an Initial Coin Offering (ICO), and distribute tokens in return for investment. Apart from distributing tokens only for investment, some token creators also offer so-called bounties, in which social-media engagement, translation and other activities are rewarded with tokens. This idea can also be found in several so-called Airdrops, in which a large number of participants can obtain tokens either for free or for similar online activities such as re-tweeting or following an online presence. By giving out tokens to a large number of addresses, the airdrop operators hope to kickstart their project. If the value of the tokens increases at a later stage, the founders can sell some of their retained tokens.

[2] https://eips.ethereum.org/EIPS/eip-20
[3] https://eips.ethereum.org/EIPS/eip-721

3 Related Work

In the context of distributed ledgers, address clustering heuristics determine a one-to-many mapping of entities to addresses [9]. While the addresses are likely to be controlled by the same entity, some addresses could be clustered incorrectly. Due to a lack of ground truth, quantifying the error rate is very difficult.

Notwithstanding, a long line of research has examined the anonymity properties of Bitcoin [27, 24, 1, 18, 31], frequently using address clustering to identify entities. Therefore, they can study transaction graphs between entities. This is in contrast to Ethereum, where the existing studies focus on the address graph [32, 30, 8, 5], as no entity identification heuristics have been proposed so far.

3.1 Address clustering methods

The most frequently used approaches to cluster addresses in Bitcoin and other UTXO based ledgers are the multiple input heuristic, and the change heuristic. The multiple input heuristic is based on the idea that multiple UTXOs which are used as input for a transaction are most likely controlled by the same entity [26, 18]. Similarly, the change heuristic assumes that a previously unused one-time change address created by a transaction is likely controlled by the same entity that created the transaction [1, 18, 31]. The effectiveness of these heuristics has been studied [9] and are implemented in open source analysis software like BlockSci [13] and GraphSense [11], which enable a range of features, including the tagging of entire address clusters given a label of one of its members.

By exploiting Airdrops based on existing wallets on Bitcoin, the reuse of addresses in newly created blockchains has enabled cross-ledger address clustering [10]. Related, and as an example of heuristics proposed for an alternative blockchain, Moreno-Sanchez et al. have developed clustering heuristics for the Ripple platform [20]. They exploit exchange gateways that allow exchanging Ripple with Bitcoins and other altcoins and are thus able to link wallets across cryptocurrencies. However, the approach it is not based on deposit address reuse, which is introduced in this paper.

Considering network-level information, Neudecker and Hartenstein associate IP addresses to transactions and exploit correlations with clusters [23]. Apart from these heuristics, Bitcoin users have been identified based on features derived from their transaction behavior [19]. By similar means, Jourdan et al. have characterized Bitcoin entities [12]. To the best of our knowledge, no clustering heuristics have been proposed for Ethereum's account model so far.

3.2 Address clustering countermeasures

To complicate the analysis of currency flows and disguise existing entities, a number of coin mixing services have been developed. These include CoinJoin [16] which lets separate entities create transactions jointly, causing the standard multiple-input heuristic to produce false results, as well as XIM [4] and CoinShuffle [28]. Coin mixing services have also been proposed for Ethereum, through smart contract-based solutions like Möbius [17] and Mixeth [29].

4 Data Collection

To perform our analyses, we have collected all blocks, transactions and event data up until block number 8,500,000 on the Ethereum blockchain, which appeared on September 7th, 2019. The following highlights the data parts that are relevant for the present work.

- **Transaction** data consists of a source and a target account address, as well as the amount of Ether transferred or smart contract function called. This data also includes *internal* transactions, that originate from smart contracts but are originally triggered by an EOA.
- **Event** data consists of a list of topics, that characterize the event, and a data field carries some value. This lets us extract any type of event a smart contract has triggered. Therefore, we extract all token `Transfer` events and the token minting events `Mint`, `Distr`, `Airdrop` and `Tokendrop`, that are sometimes used for initial token distributions. Finally, we retrieve `Approval` events, which state that an owner approves another address to spend some of his tokens. The type and number of extracted events are listed in Table 1.

4.1 Account types

For the following heuristics and analyses, we make extensive use of knowledge about the characteristics of addresses on the Ethereum blockchain. We categorize each address into whether it is an EOA or a smart contract, if it has mined blocks, and whether any transactions originate from it. If an address was never source of a transaction, we define it as inactive. One such inactive address is `0x0000000000000000000000000000000000000000`, which is commonly used to *burn* cryptoassets. Ether or tokens that are sent to this address become inaccessible because in all likelihood no one has the private key to this account.

Finally, we also obtained a list of addresses that are known to belong to exchanges. To do so, we have extracted all exchange addresses as listed by Etherscan[4], adding additional addresses manually, which we identified through our own exchange deposits and research on public discussion forums. Table 2 shows the number and type of accounts in our dataset.

Table 1. Event types and counts

Event type	**Count**
Transfer	255,931,124
Mint	3,528,933
Distr	7,978,077
Airdrop	156,131
Tokendrop	19,036
Approval	7,325,925

Table 2. Account address types and counts

Account characteristic	**Count**
EOA, active	53,291,969
EOA, inactive	22,641,698
Smart contract	17,970,742
Miner address	4,922
EOA, exchange	186
Smart contract, exchange	28

[4] https://etherscan.io/accounts/label/exchange

5 Heuristics

In the following subsections, we illustrate three entity identification heuristics: deposit address reuse, airdrop multi-participation and self-authorization. Each of the heuristics are based on usage patterns that can be observed on the Ethereum ledger. This means they are not inherent to the protocol, so that their effectiveness could change over time.

5.1 Deposit address reuse

The fact that the reuse of exchange deposit addresses provides a way to link addresses to each other is practically known, but has not yet been systematically exploited. In order to sell Ether or other cryptoassets, a user has to send them to an exchange. To credit the assets to the correct account, exchanges typically create so-called deposit addresses, which will then forward received funds to a main address. As these deposit addresses are created per customer, multiple addresses that send funds to the same deposit address are highly likely to be controlled by the same entity. This concept is illustrated in Figure 1. The key challenge lies in identifying these deposit addresses. Their characteristic property is that they forward received amounts to a major exchange account. The forwarded amount is often slightly less than what was received, as the exchange has to pay for the transaction costs. In most cases, deposit addresses are EOAs, but they can also be smart contracts. When depositing tokens on the cryptocurrency exchange Kraken for example, users are instructed to send them to a given smart contract address, identical versions of which have been mass deployed in advance. This makes it trivial to identify all identical token deposit contracts deployed by Kraken. They are designed to forward received tokens automatically, thereby passing on the transaction costs to the user. Here, we focus on the forwarding principle.

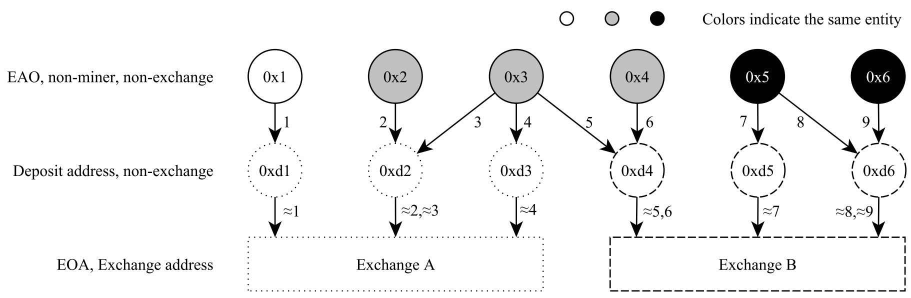

Fig. 1. Deposit address reuse: if 0xd1 to 0xd6 are exchange controlled deposit addresses that forward what is received, we cluster addresses that use the same deposit address. We can see 5 entities: 2 exchanges (dotted/dashed) and 3 potential users (colored).

Identifying deposit addresses relies on two parameters: the maximum amount difference between what was received and forwarded: a_{max} and the maximum time difference between receiving and forwarding: t_{max}. While the timing restriction ensures the forwarding characteristic, avoiding coincidental matches, a_{max} frequently corresponds to the transaction fees that are paid in the forwarding process. However, if a deposit address is a smart contract, the fee can be 0, as the EOA initiating the transaction pays for the fee. Secondly, if a sufficiently small amount of Ether is transferred to a forwarding deposit address, the exchange may wait for more deposits to make it worth the transaction fees. In the case of tokens, a_{max} is typically 0, as transaction fees cannot be paid with tokens.

Sometimes exchanges send funds to one another. As these could accidentally appear as a deposit address in a forwarding trace, we exclude known exchange addresses. We also require that the deposit address only forwards to a single exchange address. In practice however, an exchange may change their main wallet address. By imposing this restriction, we avoid accidentally linking major exchanges to the same entity. Finally, we exclude addresses using a deposit address that are either a known exchange address or have mined blocks. The former case appears frequently when users send funds directly between exchanges, the latter is frequent in mining pools, where participants request their share to be sent to a deposit address directly. For the full process see Algorithm 1.

Algorithm 1: Deposit address reuse heuristic

```
Input  : $G(V, E)$, $V_{exch} \subset V$, $V_{miner} \subset V$, $a_{max}$, $t_{max}$
         $V$: addresses, $E$: Ether transactions and token transfers
Output: Mappings $M_e$ and $M_u$ of addresses for each entity
1  foreach path $v_u \rightarrow v_d \rightarrow v_e$,
2            where $v_u \notin V_{exch} \cup V_{miner}$, $v_d \notin V_{exch}$, $v_e \in V_{exch}$ do
3      $e_1 = v_u v_d$;   $e_2 = v_d v_e$;
4      if $e_1.type = e_2.type$ and
5        $e_1.amount - e_2.amount \in [0, a_{max}]$ and
6        $e_2.blockNumber - e_1.blockNumber \in [0, t_{max}]$ then
7          depositAddresses.add($v_d$);
8          exchangeEntities.addPath($v_d \rightarrow v_e$);          // builds a graph
9          userEntities.addPath($v_u \rightarrow v_d$);              // builds a graph
10 // find weakly connected components as address clusters
11 $M_e$ = getWCC(exchangeEntities) ;                        // for exchanges
12 // remove deposit addresses as they belong to exchanges
13 $M_u$ = getWCC(userEntities) \ depositAddresses ;          // for users
```

Parameter estimation. We initially identify Ether and token forwarding traces in a time window t_{max} of 10,000 blocks, and an amount difference a_{max} of 1 Ether. In the result, the empirical a_{max} in non-contract forwards is 0.0083 Ether at the 95th percentile, and t_{max} at the 95th percentile is 3,185 blocks, corresponding to approximately 13 hours. Hence we rerun the extraction with thresholds a_{max}=0.01 Ether and t_{max}=3,200 blocks. As a result, we identify 13,104,448 traces that forward Ether or tokens to an EOA exchange address.

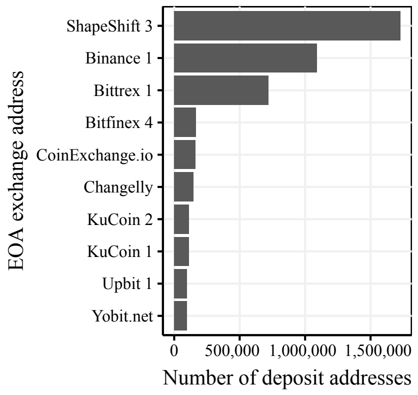

Fig. 2. Top 10 EOA exchange addresses by number of deposit addresses that only forward Ether and tokens to them. About 1.7 million belong to Shapeshift.

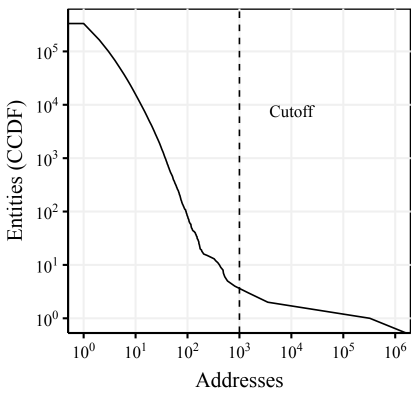

Fig. 3. CCDF showing how many entities each map to a minimum number of addresses. For example, about 10,000 entities consist of 10 or more addresses.

Results. Clustering the deposit addresses with the exchanges provides insight into how large the exchange clusters are. Figure 2 illustrates the top 10 exchange addresses by cluster size. We can see that Shapeshift and Binance form some of the largest clusters, with the former covering more than 1.7 million deposit addresses. In total, we can associate 6,670,392 deposit addresses to 186 EOA exchange addresses. Out of these, 5,671,405 are EOA, which means relative to all active EOA accounts, exchange deposit addresses account for 10.6%.

With respect to the accounts that have sent transactions or tokens to deposit addresses, we can make the following statements: Out of the 3,261,091 addresses that have used a deposit address, 1,446,715 (44.3%) have reused a deposit address with more than one account. In total, there are 333,107 entities that consist of more than one address. We can explore the full distribution with a complementary cumulative distribution function (CCDF), which is illustrated in Figure 3. There, we can also see that we find 4 entities with each more than 1,000 addresses (indicated by the cutoff). While not impossible, we believe such large address clusters are unlikely, and therefore ignore them.

Limitations. To consider how this heuristic could lead to false positives, we assume the role of an adversary. As soon as we receive a transaction from an arbitrary address, we send the same amount to one of the known Exchange wallets. This would result in our account being considered a forwarding deposit address. In this way, the sending address cluster could be extended to include our own address. Furthermore, we've only investigated one layer of forwarding. With this approach, we also can't capture which major exchange addresses belong to each other, as we've limited deposit addresses to only have one target.

5.2 Airdrop multi-participation

Airdrops are a popular mechanism to distribute tokens. On the Ethereum blockchain, they are performed through smart contracts. The owners of the smart contract may choose recipients randomly, based on past activity, or ask users to sign up through online forms. Some of these registration processes require users to perform certain actions on social media, such as posting articles or becoming a follower. The amount of tokens given to each user is either fixed, or based on existing account balances. If the amount is fixed, there is an incentive to cheat the system. A single user could sign up with multiple email addresses and perform actions with multiple social media accounts. Once the airdrop is performed, the user will receive the tokens on all of his registered addresses. Since it is impractical to manage the tokens on all of them, they are usually collected and aggregated to one address.

We can exploit this pattern to identify single entities that receive tokens multiple times. The concept is illustrated in Figure 4. We identify Airdrops where a fixed amount of tokens is distributed to many recipients. Then we search for addresses that have been forwarded the same amount from the initial recipients. It is important to ensure that these second hop recipients are not exchange wallets or Decentralized Exchange (DEX) contracts, as several honest recipients may transfer their tokens there directly. Furthermore, they must not be inactive accounts, as this could indicate many recipients burning the token.

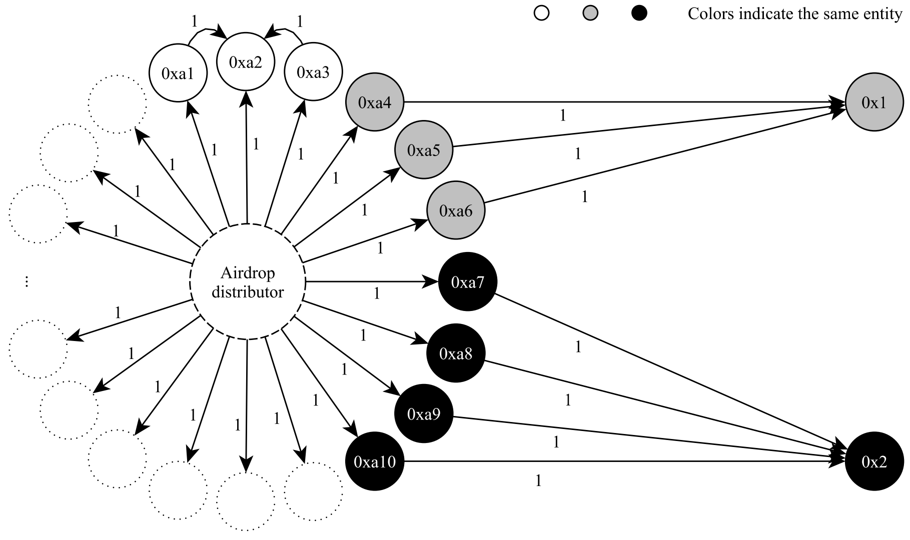

Fig. 4. In a token airdrop, where a large number of addresses (0xa1, ..., 0xa$n$) receive the same token amount (in this case 1), we cluster addresses that forward the exact received amount to a single address. Receiving addresses should be active EOAs, and should not be an exchange or a smart contract, such as a DEX.

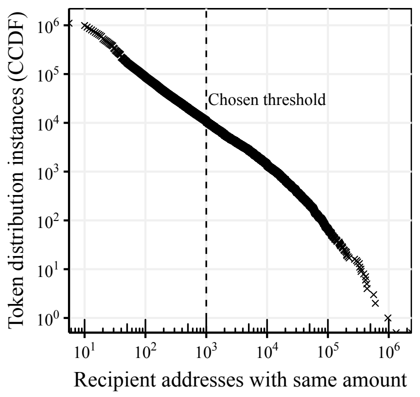

Fig. 5. CCDF illustrating fixed amount token distribution sizes. At the threshold, there are about 10,000 distribution events with at least 1,000 recipients.

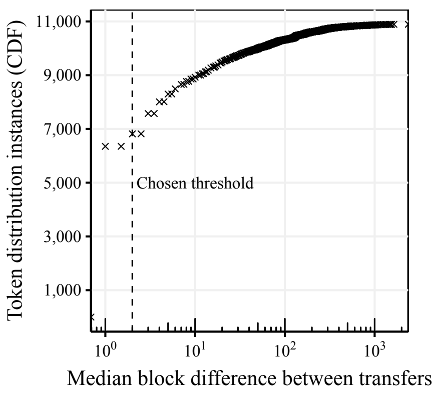

Fig. 6. CDF illustrating median block difference between airdrop distribution transactions. At a difference of less than 2, there are 6,819 distribution events.

The heuristic depends on two inputs. First, a set of airdrops with equal amounts, characterized by a signature of a distributing address, a token network and an amount. Second, the minimum number of token aggregations agg_{min} into a single address. The second parameter is trivial to choose, as multi-participation in its smallest form consists of two airdrop recipient addresses forwarding their tokens to a third address ($agg_{min} = 2$). In this case a single entity would be in control of at least 3 addresses.

Input and parameter choice. The main challenge lies in identifying airdrops. As we have no ground truth on airdrops, we first examine all same-source, fixed amount token distribution events. Figure 5 shows the CCDF of same amount token distributions. We can observe that there are about 10,000 distribution events with at least 1,000 recipients. Manual inspection reveals that this also includes token transfers within the EOS token network, which was an ICO, not an airdrop. Therefore we must further filter the set of token distribution events. As airdrops are frequently distributed in an automated fashion, we can inspect the temporal domain of such a distribution event. We calculate the block difference between the individual airdrop token transfers and calculate the median block difference. If it is very low, a large number of addresses received their tokens in a short time frame, so we assume it to be an airdrop. Figure 6 shows a cumulative distribution function (CDF) of how many distribution events fall into a maximum median block difference. The fastest same-amount EOS transfers with at least 1,000 recipients occur with a median block difference of 4. Therefore, we only select distributions where this difference is less than 2. This means at least 500 recipients have received their tokens in consecutive time steps of at most one block, corresponding to about 15 seconds on average.

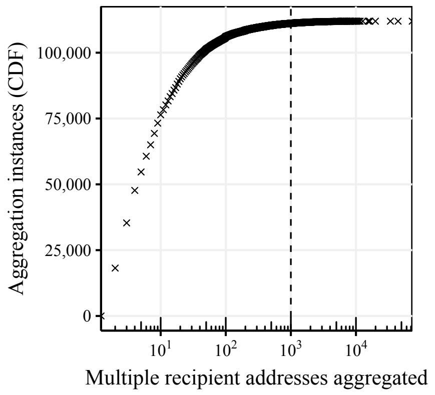

Fig. 7. CDF illustrating recipient aggregation instances. For example, there are 111,174 instances where between 2 and 1,000 recipients are aggregated.

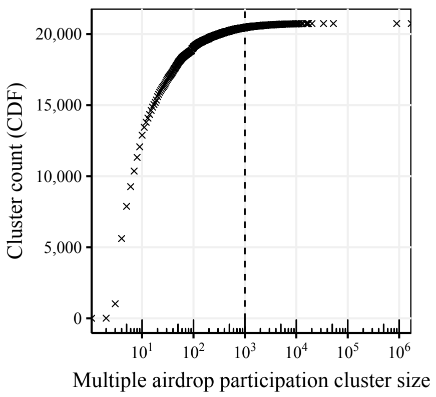

Fig. 8. CDF illustrating the final cluster sizes after joining. At the chosen threshold, there are 20,453 clusters containing between 2 and 1000 addresses.

Secondly, we need to determine what constitutes a suspicious aggregation process. Figure 7 shows the CDF of aggregation instances by maximum number of addresses collected from. Already two airdrop recipients forwarding their tokens to a single address can constitute multi-participation. Visible in the plot, the CDF reaches a plateau from about 1,000 token receiver aggregations. There are aggregations with more addresses participating, but only very few of them.

Results. Retrieving all aggregations results in 4,880,118 traces from airdrop source to final collecting address. The median time between airdrop and collection is 10 days, with the lower quartile at 40 hours. One user likely participates in multiple airdrops, where each multi-participation may slightly differ. Depending on the requirements for airdrop participation, users may add additional addresses, or not use all of them. As such, address clusters can merge. Once the joining is performed, we obtain our final entity clusters. The corresponding distribution is illustrated in Figure 8. Due to the merging, the number of entities we can extract is lower than the number of aggregation instances depicted in Figure 7. Some very large clusters have formed, which are unlikely to exist. This could be due to a collecting address that is actually a service used by many users. Secondly, some token transfers may have been falsely identified as airdrops. To reduce such issues, we only consider entities consisting of at most 1,000 addresses. Using this threshold, we count 675,512 addresses, likely controlled by 20,453 entities.

5.3 Self-authorization

The ERC20 token standard requires an `approve` function to allow another address to spend tokens on behalf of the actual owner. Through the execution, a spender address gains access to a limited amount of tokens. This functionality is mainly used in connection with smart contracts, especially with decentralized exchanges. Although smart contract use is the main purpose, this type of authorization can also be used for regular EOA addresses.

In this section, we exploit this functionality under the assumption that there are users that approve another address they own. We call this process *self-authorization*. Reasons for such self approval might include test purposes or risk distribution over several addresses with partial accessibility. Successful function calls typically emit an `Approval` event, which contains the owner, spender and permitted amount. As stated in section 4, we have obtained 7,325,925 such events. Out of these, 338,510 (≈4.6%) are between active EOA addresses. As there may still be exchange addresses among the approved spenders, we remove them accordingly. Finally, we extract all unique pairs of owners and spenders, disregarding the type of token or the amount.

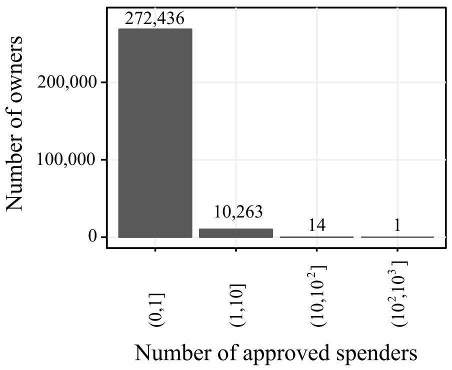

Fig. 9. Most EOA owners approve exactly one EOA spender. More than 100 approved spenders appears once.

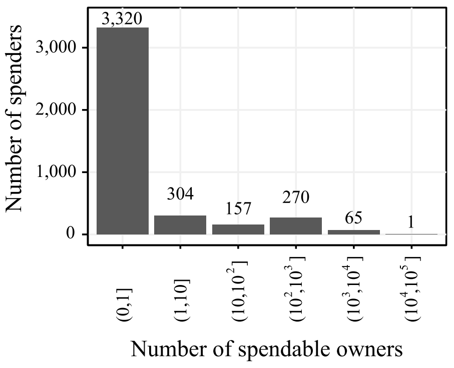

Fig. 10. Most spenders have been approved by one owner, one spender is approved by more than 10,000 owners.

We can then study the relationship between these owners and spenders. Figure 9 illustrates that the vast majority of owner addresses only approve one spender address. However, it appears that this single spender address is frequently the same across many owner addresses: On the far right side of Figure 10, we can observe that there is one spender address, that has been approved by more than 10,000 owners, and 65 addresses with more than 1,000 owners. For these, it is unlikely that they belong to the same entity. To extract entities, we believe a limit of up to 10 owners approving the same spender and up to 10 spenders approved by the same owner is a plausible. Doing so, lets us extract 4,599 entities from 7,107 addresses.

6 Analysis

In this section, we study the applicability of each clustering heuristic over time. Secondly, we apply the heuristics on a sample token network which highlights how the results allow for an interpretation of the interactions in the network.

Figure 11 illustrates how many newly seen addresses are clustered with an existing entity per block range of 100,000 blocks. It clearly illustrates that the deposit address clustering heuristic is the most effective by number of captured addresses. Most of these however, are the exchange deposit addresses themselves. Both deposit address reuse and multiple airdrop participation decrease in number of captured addresses. Even though the number of addresses captured by multi-airdrop participation is much lower, they appear consistently relative to the total number of addresses captured by all heuristics. The self-authorization heuristic however, only captures a very small number of addresses. In fact, there are so few of them, that they are not visible in the chart. With all clustering heuristics combined, we can cluster 10,561,143 addresses into 343,467 entities. The majority of these addresses belong to the exchange entities, which include smart contract deposit addresses. The number of EOA addresses we were able to cluster is 9,562,153, which equates to a share of 17.9% relative to all active EOAs.

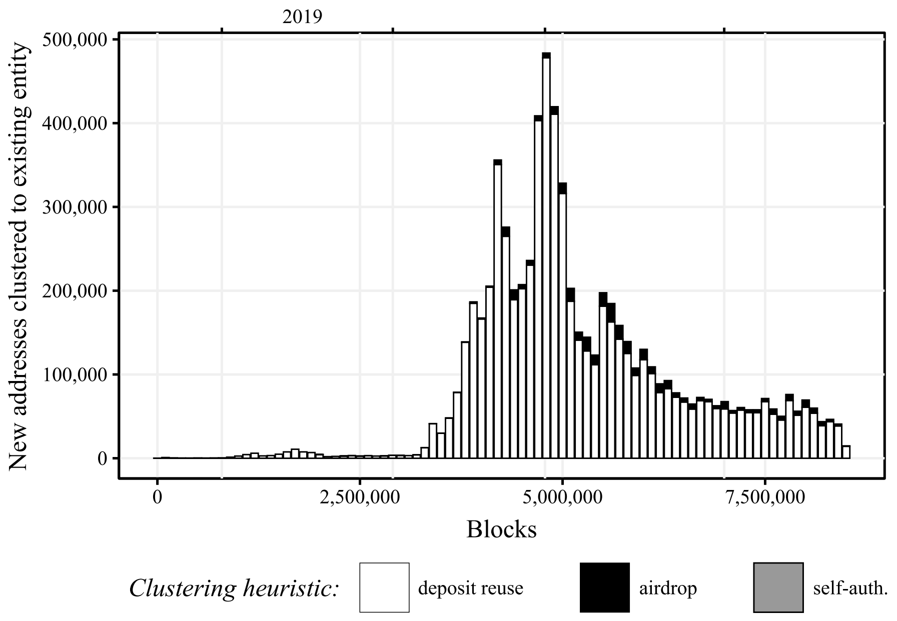

Fig. 11. Newly seen addresses that are clustered with previously seen addresses. The exchange deposit address clustering heuristic is responsible for most address clusters.

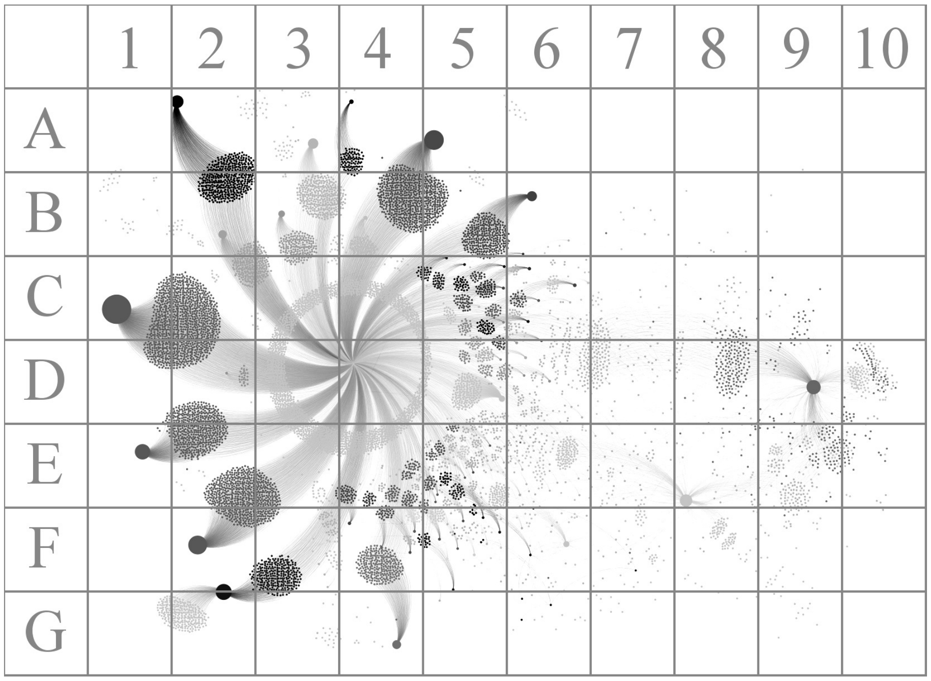

Fig. 12. The Bionic token network (0xef51c9377feb29856e61625caf9390bd0b67ea18). Nodes close to each other with the same shade of gray indicate the same entity. Node size corresponds with their indegree. The Bionic token network contains an airdrop source at D4, the circle surrounding it are recipients that received tokens, but never did anything with them. But there are many airdrop recipients that appear to belong to the same entity, as they aggregate their received tokens. At D9, the HotBit exchange is visible. Deposit addresses belonging to HotBit are visible in C8.

In Figure 12, we illustrate the airdrop and deposit heuristics applied to the token transfers of only the Bionic token network, and highlight entities with shades of gray. In the token network, we can see that an airdrop has been performed originating from D4. The airdrop itself is responsible for a large part of all transfers. Many recipients did not forward their received tokens, but some of them trade them on exchanges like IDEX (E8) or Hotbit (D9). Airdrop recipients in C7-D7 forward tokens to Hotbit's deposit addresses in D8. Addresses in D10 have received tokens from Hotbit, and some have been sent back.

Surrounding the airdrop, there are 170 clusters of entities that likely control multiple addresses. They have received airdropped tokens and forwarded them to a single address, indicated by a larger node size. The majority of these entities have then forwarded tokens to the decentralized exchange IDEX, most likely in order to sell them. Due to the many transfers involved in collecting from multiple addresses, the token network appears to have significant activity, when in reality, a large portion of this activity originates from a few entities.

7 Discussion

Due to a lack of ground-truth labels on which addresses actually belong to the same entity, it is very difficult to assess the quality of the clustering heuristics. This same issue is prevalent in existing UTXO-based clustering heuristics. In comparison to them, the proposed approaches in this paper have the drawback that they are not parameter-free. They require lists of previously known addresses or thresholds. Nevertheless, in the case of deposit address reuse, the advantage lies in the fact that the usefulness can be improved when provided with more labels of major exchange addresses. Some of the very large cluster formations could be due to unknown exchange addresses. In the case of airdrop multi-participation, the main challenge is identifying airdrops correctly. We have chosen the path of counting same amount recipients, as well as considering the temporal domain. As a result, some very large clusters have formed which we had to exclude. The quality of the results would benefit from more sophisticated airdrop detection methods. With respect to the utility of each of the heuristics, we can state the following: whereas deposit address reuse and self-authorization may provide insightful links for future analysis surrounding fraudulent behavior, we expect that the clusters around airdrop multi-participation are mostly limited to the particular use case of multi-participation.

8 Conclusion and Future Work

This paper is the first to propose clustering heuristics for Ethereum's account model, including an analysis of their applicability. We have explored deposit address reuse, airdrop multi-participation and self-authorization. For each heuristic, we have analyzed and selected parameters as inputs. We have shown that the exchange deposit address reuse heuristic captures the majority of addresses, whereas the airdrop multi-participation heuristic can provide fewer but additional address clusters. The self-authorization heuristic however, has only provided very few results. Overall, we are able to cluster 17.9% of active addresses on the Ethereum blockchain, which may form the foundation of future entity graph analyses related to usage assessments or fraud detection.

8.1 Future work

As part of future work, we believe the detection of exchange wallets is important to improve the clustering results. Further usage patterns on the Ethereum blockchain can be studied. They may provide insight into how entities use them, which in turn allows for clustering heuristics. Examples include online wallets, identity management solutions like ERC 725, smart contracts related to games, gambling or services in the realm of decentralized finance.

Another challenge is the question of how to treat smart contract accounts when identifying entities. A smart contract could act as a regular wallet, in which case the owner is likely the creator. But it is also possible that the smart contract merely forwards currency, in which case an owner is not important.

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. Bartoletti, M., Pes, B., Serusi, S.: Data mining for detecting bitcoin ponzi schemes. In: 2018 Crypto Valley Conference on Blockchain Technology (CVCBT). pp. 75–84. IEEE (2018)
3. Bartoletti, M., Pompianu, L.: An empirical analysis of smart contracts: platforms, applications, and design patterns. In: International conference on financial cryptography and data security. pp. 494–509. Springer (2017)
4. Bissias, G., Ozisik, A.P., Levine, B.N., Liberatore, M.: Sybil-resistant mixing for bitcoin. In: Proceedings of the 13th Workshop on Privacy in the Electronic Society. pp. 149–158. ACM (2014)
5. Chen, T., Zhu, Y., Li, Z., Chen, J., Li, X., Luo, X., Lin, X., Zhange, X.: Understanding ethereum via graph analysis. In: IEEE International Conference on Computer Communications. pp. 1484–1492. IEEE (2018)
6. Chen, W., Zheng, Z., Cui, J., Ngai, E., Zheng, P., Zhou, Y.: Detecting ponzi schemes on ethereum: Towards healthier blockchain technology. In: Proceedings of the 2018 World Wide Web Conference. pp. 1409–1418. International World Wide Web Conferences Steering Committee (2018)
7. Fanusie, Y.J., Robinson, T.: Bitcoin laundering: An analysis of illicit flows into digital currency services. A memorandum by the Center on Sanctions and Illicit Finance and Elliptic (January 2018)
8. Ferretti, S., D'Angelo, G.: On the ethereum blockchain structure: A complex networks theory perspective. Concurrency and Computation: Practice and Experience p. e5493
9. Harrigan, M., Fretter, C.: 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). pp. 368–373. IEEE (2016)
10. Harrigan, M., Shi, L., Illum, J.: Airdrops and privacy: A case study in cross-blockchain analysis. In: 2018 IEEE International Conference on Data Mining Workshops (ICDMW). pp. 63–70. IEEE (2018)
11. Haslhofer, B., Karl, R., Filtz, E.: O bitcoin where art thou? insight into large-scale transaction graphs. In: SEMANTiCS (Posters, Demos) (2016)
12. Jourdan, M., Blandin, S., Wynter, L., Deshpande, P.: Characterizing entities in the bitcoin blockchain. In: 2018 IEEE International Conference on Data Mining Workshops (ICDMW). pp. 55–62. IEEE (2018)
13. Kalodner, H., Goldfeder, S., Chator, A., Möser, M., Narayanan, A.: Blocksci: Design and applications of a blockchain analysis platform. arXiv preprint arXiv:1709.02489 (2017)
14. Kappos, G., Yousaf, H., Maller, M., Meiklejohn, S.: An empirical analysis of anonymity in zcash. In: 27th USENIX Security Symposium, USENIX Security 2018. pp. 463–477 (2018)
15. Madore, P.H.: Crypto Market OpenBazaar Confirms Upcoming Support for Ethereum. https://www.ccn.com/openbazaar-adding-support-ethereum-soon/ (2019), [Online; accessed 12-September-2019]

16. Maxwell, G.: CoinJoin: Bitcoin privacy for the real world. bitcointalk.org/index.php?topic=279249 (2013), [Online; accessed 12-September-2019]
17. Meiklejohn, S., Mercer, R.: Möbius: Trustless tumbling for transaction privacy. Proceedings on Privacy Enhancing Technologies pp. 105–121 (2018)
18. 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. Proceedings of the Internet Measurement Conference - IMC '13 (6), 127–140 (2013)
19. Monaco, J.V.: Identifying bitcoin users by transaction behavior. In: Biometric and Surveillance Technology for Human and Activity Identification XII. vol. 9457, p. 945704. International Society for Optics and Photonics (2015)
20. Moreno-Sanchez, P., Zafar, M.B., Kate, A.: Listening to whispers of ripple: Linking wallets and deanonymizing transactions in the ripple network. Proceedings on Privacy Enhancing Technologies (4), 436–453 (2016)
21. Möser, M., Böhme, R., Breuker, D.: An inquiry into money laundering tools in the bitcoin ecosystem. In: 2013 APWG eCrime Researchers Summit. pp. 1–14. IEEE (2013)
22. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system (2008)
23. Neudecker, T., Hartenstein, H.: Could network information facilitate address clustering in bitcoin? In: International conference on financial cryptography and data security. pp. 155–169. Springer (2017)
24. Ober, M., Katzenbeisser, S., Hamacher, K.: Structure and anonymity of the bitcoin transaction graph. Future internet (2), 237–250 (2013)
25. Paquet-Clouston, M., Haslhofer, B., Dupont, B.: Ransomware payments in the bitcoin ecosystem. Journal of Cybersecurity (1), tyz003 (2019)
26. Reid, F., Harrigan, M.: An analysis of anonymity in the bitcoin system. In: Security and privacy in social networks, pp. 197–223. Springer (2013)
27. 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)
28. Ruffing, T., Moreno-Sanchez, P., Kate, A.: Coinshuffle: Practical decentralized coin mixing for bitcoin. In: European Symposium on Research in Computer Security. pp. 345–364. Springer (2014)
29. Seres, I.A., Nagy, D.A., Buckland, C., Burcsi, P.: Mixeth: efficient, trustless coin mixing service for ethereum. IACR Cryptology ePrint Archive p. 341 (2019)
30. Somin, S., Gordon, G., Altshuler, Y.: Network analysis of erc20 tokens trading on ethereum blockchain. In: International Conference on Complex Systems. pp. 439–450. Springer (2018)
31. Spagnuolo, M., Maggi, F., Zanero, S.: Bitiodine: Extracting intelligence from the bitcoin network. In: International Conference on Financial Cryptography and Data Security. pp. 457–468. Springer (2014)
32. Victor, F., Lüders, B.K.: Measuring ethereum-based erc20 token networks. In: International Conference on Financial Cryptography and Data Security. pp. 113–129. Springer (2019)
33. Wood, G.: Ethereum: A secure decentralised generalised transaction ledger, https://github.com/ethereum/yellowpaper