Ethereum Analysis via Node Clustering

Hanyi Sun, Na Ruan(✉), and Hanqing Liu

Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai, China
naruan@cs.sjtu.edu.cn

Abstract. As an open source public blockchain with the capabilities of running smart contract, Ethereum provides decentralized Ethernet virtual machines to handle peer-to-peer contracts through its dedicated cryptocurrency Ether. And as the second largest blockchain, the amount of transaction data in Ethereum grows fast. Analysis of these data can help researchers better understand Ethereum and find attackers among the users. However, the analysis of Ethereum data at the present stage is mostly based on the statistical characteristics of Ethereum nodes and lacks analysis of the transaction behavior between them. In this paper, we apply machine learning in Ethereum analysis for the first time and cluster users and smart contract into groups by using transaction information in existing blocks. The clustering results are analyzed by using the identity information of the available Ethereum users and smart contracts. Based on the clustering results, we propose a new way of user identity discrimination and malicious user detection.

Keywords: Blockchain · Ethereum · Network embedding

1 Introduction

As the second largest blockchain platform, Ethereum [1] has had a market value of nearly 20 billion since its inception in 2015. Different from traditional blockchain like Bitcoin [2], Ethereum is an emerging blockchain platform in which users can create smart contracts. This feature makes the data structure of Ethereum more complicated compared with other blockchains. A smart contract is a contract implemented in code that can be executed automatically after its creation [3]. In Ethereum, transactions between users are mainly done through direct Ether trading and invocation to smart contracts.

In previous Ethereum studies, researchers focused on the statistical characteristics of Ethereum nodes and lacks analysis of the trading behavior between them [4]. Therefore, based on the work of predecessors, this study focuses on the cluster of Ethereum nodes and analysis of the clustering result. Specifically, the Ether trading between users and users in Ethereum, the creation of smart contracts and the invocation of smart contracts were studied. The clustering algorithm of machine learning was applied for the first time in Ethereum data analysis. Based on the clustering results of experiment, we propose a new way of user identity discrimination and malicious user detection.

There are two types of accounts in Ethereum, external owned accounts (EOAs) and smart contract accounts. The EOAs represent the Ethereum users in the form of a hash value, and the smart contract accounts represent smart contracts in Ethereum. In this paper, we mainly foucs on three types of transaction relationships between EOAs and smart contract accounts, including Ether transactions between external owned accounts, smart contract creation between external owned accounts and smart contract account, and smart contract invocation between external owned accounts and smart contract accounts. The two different accounts types and three different transaction relationship types form the heterogeneous network of Ethereum.

For the feasibility of the experiment, we only used part of Ethereum transaction data due to huge amount of them. Among these data, we filter the three transaction types and two account types which form a Ethereum heterogeneous network. We learn about the eigenvector representation of the account nodes in Ethereum based on the heterogeneous network. And clustering algorithms are used to cluster the nodes eigenvector representation into groups. In the analysis of the clustering results, we got some interesting observations and findings. For example, the clustering results shows that the nodes in Ethereum have obvious clustering trend and there are some clusters led by nodes with known identities. By collecting the identity information about nodes in Ethereum, we can predict the identity of the clusters to which these nodes belong including exchange market and attackers. Moreover, we propose a new way of user identity discrimination and malicious user detection based on the clustering results. Although there are some methods for malicious user detection, they are only applicable to those nodes with large degree. In this paper, our new method use the nodes with known identities to predict the identity of other nodes in the clustering results. On generality, our method could be more adapted to the specificity of Ethereum.

The main contributions of this paper are as follows:

- In our research, this paper applies clustering algorithm in the analysis of Ethereum for the first time. We collect and filter the transactions from Ethereum blocks and cluster the accounts nodes included in them.
- We use node embedding algorithm to calculate the eigenvector representation for the external owned account nodes and smart contract account nodes in the Ethereum. Based on the eigenvector, we cluster those nodes into groups and visualize the clustering results.
- We obtain some new observations by the analysis of clustering result which makes us have a better understanding of Ethereum.
- We propose a new way of malicious user detection based on the clustering result and the nodes whose identities is already known.

The rest of paper is organized as follows. In Sect. 2 we introduce the background of our paper, including Ethereum and the machine learning algorithm we used. In Sect. 3, we introduce our system model. In Sect. 4, we analyze the clustering results and propose a new way of user identity discrimination and malicious user detection. We conclude the paper in Sect. 5.

2 Background

Ethereum is the second largest blockchain and has a market value of nearly 20 billion since its inception. And it has its own cryptocurrency Ether which can be traded or used to pay for smart contract operations. Different from traditional blockchain with currency trading as the main function, users in Ethereum can develop more complex functions by developing smart contracts. These smart contracts run according to the established code, and can be called by users in Ethereum after deployment [5]. Meanwhile, user have to pay some fees to perform the operation when calling smart contracts. There some research of user relationships in Bitcoin [6–9], but they are not suitable for the case of Ethereum due to the difference in structure.

2.1 Accounts and Transactions

The accounts in Ethereum has two types, external owned accounts (EOAs) and smart contract accounts.

External owned accounts represent accounts that belongs to external owner of Ethereum, and each external owned account has its own Ether balance. The owner of the external owned account can transfer information from his or her external owned account by creating and signing a transaction. If the owner's external balance is sufficient to cover the cost of the transaction, the transaction is valid. Then the originator account will deduct the corresponding Ether amount, and the recipient account will receive the amount. In addition to direct Ether trading between external owned accounts, Ether can also be traded by calling a smart contract and the transaction deduction is determined by the code written in advance within the smart contract.

The smart contract account represents the smart contract deployed in Ethereum which controlled by the code written in advance. Each smart contract account stores the hash value of the smart contract code. In the case of a invocation, the smart contract account receives a transaction message and activate the smart contract code stored in it, which allows it to read and write to the internal storage or send other messages such as creating another smart contract.

The various activities of Ethereum's external owned and smart contract accounts are realized through transactions which are packaged into blocks and then broadcast to the entire Ethereum network. A transaction is a message that is sent from one account to another. Transactions can contain binary data called payload and Ether coins. If the target account contains code, the code will execute, and payload is the input data. If the target account is a zero account, the transaction will create a new smart contract. And each transaction in Ethereum contains the sender's signature, the recipient and the number of Ethers sent.

At the same time, based on different transaction types, Ethereum's transaction information also contains several other types, such as gas, gasprice and other optional data items used as smart contract execution fees.

Due to the features of blockchain, different activities in Ethereum will form transactions such as Ether trading transactions, smart contract creation and smart contract invocation. At the same time, these transactions are packaged into blocks and spread throughout the Ethereum network.

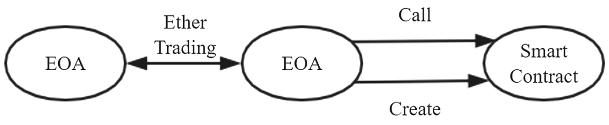


Fig. 1. Three transaction types in Ethereum

As shown in Fig. 1, we mainly focus three different trading relationships between external owned accounts and smart contract accounts:

– Direct Ether trading transactions between EOAs.
– One EOA creates a smart contract account. The external owned account firstly writes the code of the smart contract, and uploads the smart contract to the Ethereum by paying a certain amount of Ether.
– One EOA calls a smart contract account. Once a smart contract account is uploaded to the Ethereum, the users in the Ethereum can call the smart contract and realize the trading activity through the code logic in it.

In Ethereum, the Ether transaction may be done by calling a smart contract. In this case, we treat it as both smart contract invocation and Ether trading transaction in this paper.

2.2 Method of Node Embedding

In Ethereum, there are two different accounts types and three different transaction relationship types between them which form a heterogeneous network. SO in this paper, we learn from Metapath2vec [10] as the eigenvector representation learning method.

In the previous application, Metapath2vec mainly solved the problem of learning the eigenvector representation of scholars and conferences in the academic network. The main idea of Metapath2vec is to use the meta-path to guide the random walk [11] acquisition path in the academic network and learn these paths. Suppose there are three different nodes in the academic network, scholars, papers and venues. Meanwhile there are two different relationships papers written by authors and these papers are published on venues. The idea of Metapath2vec is to guide the way of random walks through a meta-path, A-P-C-P-A.

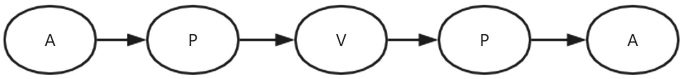


Fig. 2. Meta-Path in academic networks

As shown in Fig. 2, in the academic network we first select an author A and then randomly walk to a paper P written by the author A and the venue V which the paper published at. Then randomly walk to another paper P which published at this conference and the author A of the paper. By analogy, a path of author-paper-venue is formed. This Meta-Path guides random walks in the academic network, forming many different paths that preserve the concept of "word context". After that, each path obtained by random walk is considered as a sentence, the node is considered as a word and the adjacent nodes in the path are regarded as contexts. Then these paths are learned by using the skip-gram model [12]. In this way, the eigenvector representation of the nodes in the academic network can be obtained. There are two different nodes and three different relationships in the Ethereum network which is similar to academic networks. In this paper, we learn the eigenvector representation of the nodes in Ethereum with reference to the idea of Metapath2vec.

3 System Model

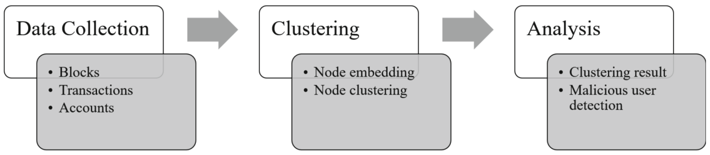


Fig. 3. Our system model with three phases

As shown in Fig. 3, our model can be divided into three phases: Data Collection, Clustering and Analysis. Data Collection collects the block data we used. Clustering part contains Node Embedding and Node Clustering. Node Emmbedding learns the eigenvector representation of the nodes in Ethereum. Node Clustering presents the clustering results and visualize [13] them. The Analysis part will analyze the clustering results and give some new opinions.

3.1 Data Collection

The dataset of Ethereum blocks we used is from previous works of others [4]. We choose the first 10 millions of transactions and detect the three main activities, Ether trading, smart contract creation and smart contract invocation included in them. At the same time, in order to ensure the validity of the analysis, we only pay attention to the transactions confirmed in the Ethereum block, and do not consider the transactions that failed for various reasons.

Among the transactions in the block, the types of transactions we need are only Ether trading, smart contract creation and smart contract invocation. In order to extract the required transaction types from the block data, we observed that the three behavior patterns in the transaction are related to the main trading activities we are concerned with.

- The Create behavior corresponds to the creation of a smart contract.
- The Call behavior corresponds to the invocation of a smart contract.
- When the number of Ethers in the transaction is greater than 0, the two accounts in transaction have made a Ether trading.

By detecting these three behaviors in the transaction, we filter out the three types of transactions and classify them. At the same time, we exclude four types of transactions that are unrelated to the three main activities. One is that the transaction between external owned accounts but the number of Ether traded is 0. The second is the transaction in which the number of gas in the smart contract that supports the contract is 0, which means the smart contract cannot run. The third type is the Ether trading transaction between accounts which fail for various reasons. The fourth is a transaction that fails when an external owned account creates a smart contract account.

By eliminating invalid transactions and classifying and counting the selected transactions, the Table 1 is obtained.

Table 1. Number of transactions

Transaction type	Number
Ether trading transaction	**8913083**
Smart contract creation	**119347**
Smart contract invocation	**1924918**

As can be seen from Table 1, the number of Ether trading transactions is 8,913,083. The proportion of Ether trading transactions in Ethereum exceeds the sum of smart contract creation and smart contract invocation. It can be known that ether trading occupy the vast majority of transactions in Ethereum. At the same time, the smart contract creation is the least of the three types of transactions and only has a number of 119,347. Therefore, in the analysis part, we will pay more attention to the type of Ether trading transaction in Ethereum.

The number of accounts we get from the selected transactions are shown in Table 2. Based on the transactions data, we obtained 406,774 external owned account addresses and 119,347 smart contract account addresses, a total of 526,121 account addresses. There are more account addresses in Ethereum, but

Table 2. Number of accounts

Account type	Number
External owned account	**406774**
Smart contract account	**119347**
Total	**526121**

we only consider the account addresses obtained from the transactions. In other word, for other external owned accounts in Ethereum that have never made any Ether trading transactions, smart contract creation and smart contract invocation, we do not consider them in our analysis. It can be seen that the number smart contract account is as same as the number of transactions of smart contract creation. At the same time, we observed that there are a few external owned accounts which included in many transactions have play a very important role in the trading of Ethereum. Therefore, we will give more weight to these important nodes in the node emmbedding part.

3.2 Node Embedding

Before node clustering, we need to learn the eigenvector representation of the nodes in Ethereum. In the network we built, there are two types of nodes external owned accounts and smart contract accounts. And there are three types of relationships between the two types of nodes, Ether trading transactions between external owned accounts, smart contract creation and smart contract invocation between external owned accounts and smart contract accounts. In order to learn the eigenvector representation of the nodes in the Ethereum network, we have improved the idea based on Metapath2vec.

An important problem when learning the eigenvectors of nodes in network is how to transform the structure of network into the form which skip-gram model can handle. To solve this problem, Metapath2vec capture the semantic and structural correlations between different types of nodes by using meta-path-based random walks to generate paths.

In our experiments, we used the idea of improved Metapath2vec by using a mixed meta-path to guide the generation of random paths. Since external owned accounts is the main body of Ethereum, we consider the following scenarios such as both two external owend accounts have Ether trading transactions with another account, or both external owned accounts have called a same smart contract or an external owned account have called a smart contract created by another external account. In these cases, we believe that these accounts may belong to the same category or have similarities. So we use the following mixed meta-path to guide the process of random walks to generate paths and use these paths to learn the eigenvector representation of the nodes.

As shown in Fig. 4, external owned accounts are expressed by node U, smart contract accounts are expressed by SC and the transactions are represented by a

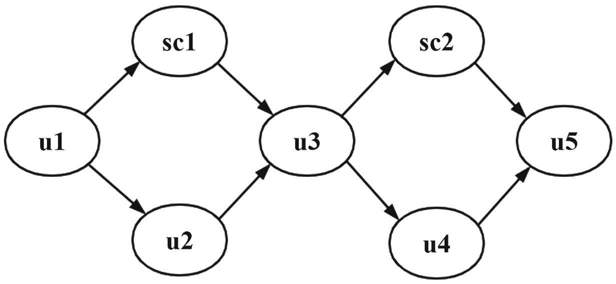


Fig. 4. Meta-path in Ethereum

line between nodes. In order to generate a path, we first select an external owned account node *u1* and then randomly walk to an external owned account node *u2* or smart contract account node *sc1* that has had a transaction relationship with it. After that, it randomly walk to another external owned account node *u3* which has a transaction relationship with the node obtained before. Then through this node, it continues to find a next external owned account node or smart contract account node. And so on, then it can walk randomly to get a fixed length path, which saves the context of the account nodes in the Ethereum network.

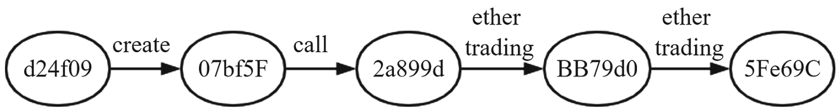


Fig. 5. A sample path in Ethereum

Figure 5 shows part of a sample path generated by randomly walk, actual accounts are represented by the node with the first six digits of their account addresses. In this path, d24f09, 2a899d, BB79d0 and 5Fe69C represent four external owned account nodes, and 07bf5F represents a smart contract account node. The lines between them represent the different types of relationships including contract creation, contract invocation and Ether trading. 2a899d calls smart contract account 07bf5F which is created by d24f09, so we can think that these two nodes d24f09 and 2a899d are closely related. At the same time, 2a899d and 5Fe69C may have a close relationship because they both have Ether trading with BB79d0.

In order to save the relationship between nodes in Ethereum network as much as possible by random walks, the specific settings of our experiment are as follows:

- Set the path length of the random walk to 100 during the random walk.

- Calculate the degree the nodes in Ethereum. For each node with a degree greater than 30, the number of paths that are randomly moved from this node as starting point is set to 300. For other nodes, the path with those node as starting point is set to 100.
- Set the eigenvector representation dimension of the nodes to 128.

In Sect. 3.1, we found that there are a small number of nodes with large accessibility have a very important influence on Ethereum. So we set more random paths which start from nodes with larger degrees. In the course of the experiment, we get millions of paths by randomly walk which save the information in Ethereum network. Then we input those paths into the skip-gram model for training and get the eigenvector representation of each node in 128 dimensions. We visualize and cluster the eigenvectors of these nodes and then analysis the clustering result.

3.3 Node Clustering

For the analysis of the node eigenvectors obtained by node embedding, we first reduce their dimensions and visualize them. T-distributed stochastic neighbor embedding abbreviated as TSNE is a machine learning algorithm used for dimensional reduction. And it is a nonlinear dimensionality reduction algorithm, which is very suitable for high-dimensional data dimensionality reduction to 2D or 3D for visualization.

First, we use the PCA dimensionality reduction algorithm [14] to initialize the eigenvector representation of the nodes, and then use the TSNE algorithm [15] to reduce dimension of the initialization result. After a period of training, the eigenvector representation of the nodes are reduced from 128 dimensions to 3 dimensions. Then we visualize the eigenvector of nodes after the dimension reduction.

In Fig. 6, each blue point represents an external owned account node or a smart contract account node in Ethereum. And the nodes in the figure are represented by three-dimensional vectors. It can be seen from the figure that after the dimensional reduction by the TSNE dimensionality reduction algorithm, the eigenvectors of the nodes show obvious clustering trends, and the boundaries between the clusters are also obvious. Based on the eigenvectors after dimension reduction, we cluster the nodes in Ethereum.

Since we have a large number of nodes and no fixed number of clusters, we have adopted the Birch algorithm that works better on larger data sets in the clustering of nodes. The Birch algorithm is an algorithm based on hierarchical clustering. It adopts a tree structure to perform fast clustering and has a good effect on large data sets. The clustering algorithm shows great results on our dataset.

Some important parameters of the Birch are as follows:

- threshold: we set the value to 0.6
- branching_factor: due to the large number of nodes, we set it to 100
- n_clusters: with no prior knowledge, we set it to None
- compute_labels: the default is True

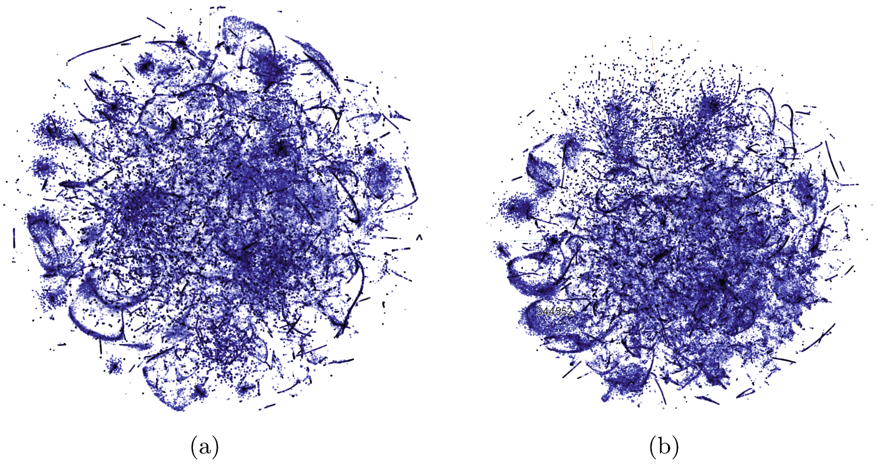


Fig. 6. Eigenvector visualization (Color figure online)

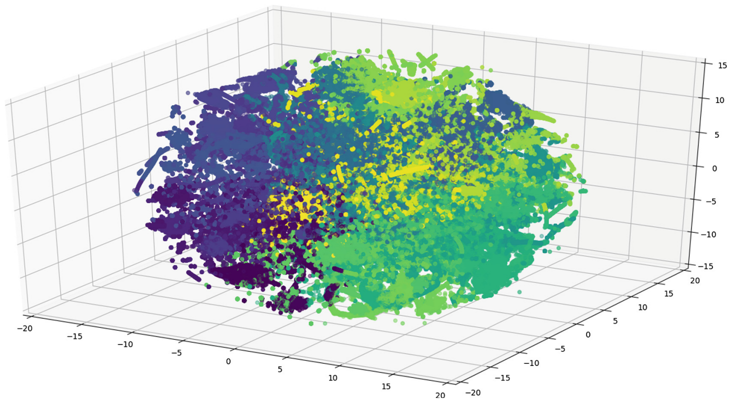


Fig. 7. Clustering result (Color figure online)

As shown in Fig. 7, we visualize the 3-dimension eigenvector of nodes in Ethereum. In Fig. 7, each colored point represents an external owned account node or a smart contract account node in Ethereum. And we distinguish the nodes in different clusters with different colors. There are many different clusters in the clustering result. In the Analysis Section, we will analyze the clustering results of the nodes and give our opinions.

4 Analysis

4.1 Nodes with Known Identity

Access to users' real identities in Ethereum is extremely difficult because users in Ethereum only need hash strings of addresses to make transactions. However, in the Ethereum tag function, users can choose to disclose their identities. In Ethereum, a small number of financial trading accounts has disclosed their identities in their tags. At the same time, some accounts were identified as attackers by Ethereum users or some previous papers due to malicious behavior. In addition, some real-life social groups will also open their Ethereum addresses on social media networks for use in fundraising and other purposes. Through these channels, we can get a small amount of identity information about external owned accounts and smart contract accounts in Ethereum.

Since the Ethereum account address is too long, we only list the shortest of their first six digits in the table to represent their address. Some of the accounts with known identities are shown in Table 3.

Table 3. Accounts with known identities

Account addr	Account identity	Account type	Organization
70faa2	Exchange market	External owned account	ShapeShift
209c47	Exchange market	Smart contract	Poloniex
fa5227	Exchange market	Smart contract	Kraken
aa1a6e	Security contract	Smart contract	ReplaySafe
1c39ba	Exchange market	Smart contract	ShapesShift
e94b04	Exchange market	Smart contract	Bittrex
9e6316	Exchange market	External owned account	ShapeShift
96fc45	Exchange market	External owned account	Changelly
9bcb07	Exchange market	Smart contract	ShapeShift
b42b20	Exchange market	Smart contract	Poloniex
42da8a	Exchange market	External owned account	YUNBI
7c2021	Attacker	Smart contract	/
3898d7	Attacker	External owned account	/
8b3b3b	Attacker	External owned account	/
29dfaa	Attacker	External owned account	/
bb9bc2	Fundraising organization	Smart contract	DAO
a74476	Exchange market	Smart contract	Golem

In Table 3, the first column represents the abbreviation of account addresses and the second column represents the known identity of the account address including exchange market, attacker and so on. The third column in the table represents the account type, including external owned account and smart contract account. The fourth column indicates the real organization to which the account address belongs.

As can be seen from the table, the nodes with known identities are mostly exchange markets which play important roles for Ethereum transactions, or attacker account addresses that are harmful to Ethereum. The accessibility of these nodes is generally high, and their identity information will be an important reference for the analysis in clustering results.

4.2 Clustering Result Analysis

In Sect. 3, we clustered the nodes vectors. Since the number of nodes in our experiment is more than 500,000, the number of clusters in the clustering result is also large. For effective analysis, we selected the top ten clusters for analysis.

Table 4. Some known accounts

Cluster ID	Number of nodes
4	14710
12	14060
29	13798
24	13597
15	13431
19	13173
9	12794
10	12215
3	11521
17	9047

In Table 4, There are some of these clusters containing nodes with known identities:

- Among the known identity nodes, there are four account addresses 70faa2, 1c39ba, 9e6316, and 9bcb07 who belong to the cluster 4. In Table 3, all four account addresses belong to the exchange market organization ShapeShift. And it can be speculated that the nodes in the cluster are mostly related to the exchange market ShapeShift.
- Similarly, it can be speculated that cluster 12 is dominated by another exchange market, Poloniex. Account addresses b42b20 and 209c47 are both in this cluster, and they belong to exchange market Poloniex according to Table 3.
- At the same time, we observed that two of the four known attackers, 3898d7 and 29dfaa belong to cluster 17. It can be speculated that the nodes in this cluster may be related to malicious attack nodes.

Among the clusters of clustering results, we find that several well-known exchange markets are the core of several large clusters. At the same time, we also found a cluster that is suspected to be related to malicious users. Among the four attacker-related account addresses of known identities, two account addresses of attackers exist in this cluster. Based on this cluster, we propose a new way for malicious user detection.

4.3 Malicious User Detection

There are already some works in malicious user analysis [16] in blockchain. However, the complexity of Ethereum structure makes these methods inadequate. We propose a more extensive and convenient malicious user detection strategy for Ethereum based on our experimental results.

In Table 3, we list four nodes associated with the attackers, and two of them 3878d7 and 29dfaa are in a same cluster. In the previous Ethereum activities, due to the large number of spam smart contracts creation (these smart contracts are usually similar in code and rarely have been called since their creation), the two nodes were identified as malicious users who manufacture junk smart contracts to consume the storage space of Ethereum. Base on the nodes with known identities, we propose a strategie for the detection of malicious users in Ethereum.

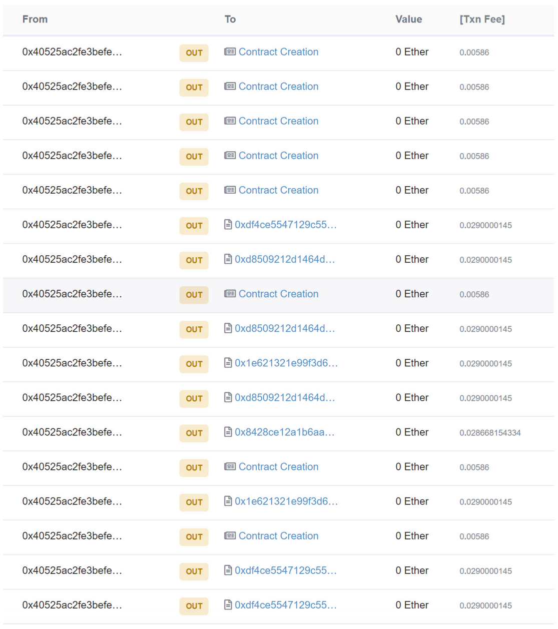
Our malicious user detection method is based on the vector space distance of the nodes. Based on the clustering results, the main steps are as follows:

- Given the known malicious users a1 and a2 and their cluster T.
- Mark the two nodes as malicious nodes, and then calculate the distance of all the nodes in the cluster T from a1 and a2 respectively.
- Select the n points closest to a1 to form N1 and the n points closest to a2 to form N2. The intersection of the two sets is taken, and the nodes in the intersection are marked as potentially malicious users.
- Find out all the marked potentially malicious users to form set A. Then do the same for every two node in A.
- The method will iterate t times to mark possible malicious users in the cluster.

In the actual analysis, we mark the two known nodes 3988d7 and 29dfaa in cluster 17 as malicious user nodes. At the same time, set n to 200 and iterate the entire detection strategy twice. By analysis of the nodes included in the obtained potentially malicious users set A, we find a node 40525a has similar malicious user behavior.

Part of the transactions of the external owned account 40525a is shown in Fig. 8. We look up all the transactions made by the node account 40525a in Ethereum and find that the node currently creates 2,504 smart contracts. We detected the smart contract created by the account and found that the smart contract code created by it is same. It proves that the node is not a developer of smart contracts but a malicious user node like the previous two external owned account nodes who consumes Ethereum's storage space by creating a large number of identical and unattended smart contract. Since the frequency and number of smart contracts created by this node is not as obvious as the previous two smart contracts, the node has not been identified as an attacker by the Ethereum Community Forum or other papers about Ethereum. But according to our analysis, this node has a great possibility of being a malicious user.

In this way, we can quickly find potential attackers without having to check the accounts in the entire Ethereum.



From		To	Value	[Txn Fee]
0x40525ac2fe3befe...	OUT	Contract Creation	0 Ether	0.00586
0x40525ac2fe3befe...	OUT	Contract Creation	0 Ether	0.00586
0x40525ac2fe3befe...	OUT	Contract Creation	0 Ether	0.00586
0x40525ac2fe3befe...	OUT	Contract Creation	0 Ether	0.00586
0x40525ac2fe3befe...	OUT	Contract Creation	0 Ether	0.00586
0x40525ac2fe3befe...	OUT	0xdf4ce5547129c55...	0 Ether	0.0290000145
0x40525ac2fe3befe...	OUT	0xd8509212d1464d...	0 Ether	0.0290000145
0x40525ac2fe3befe...	OUT	Contract Creation	0 Ether	0.00586
0x40525ac2fe3befe...	OUT	0xd8509212d1464d...	0 Ether	0.0290000145
0x40525ac2fe3befe...	OUT	0x1e621321e99f3d6...	0 Ether	0.0290000145
0x40525ac2fe3befe...	OUT	0xd8509212d1464d...	0 Ether	0.0290000145
0x40525ac2fe3befe...	OUT	0x8428ce12a1b6aa...	0 Ether	0.028668154334
0x40525ac2fe3befe...	OUT	Contract Creation	0 Ether	0.00586
0x40525ac2fe3befe...	OUT	0x1e621321e99f3d6...	0 Ether	0.0290000145
0x40525ac2fe3befe...	OUT	Contract Creation	0 Ether	0.00586
0x40525ac2fe3befe...	OUT	0xdf4ce5547129c55...	0 Ether	0.0290000145
0x40525ac2fe3befe...	OUT	0xdf4ce5547129c55...	0 Ether	0.0290000145

Fig. 8. Suspected malicious behavior

5 Related Work

In recent years, there are some studies in Ethereum Analysis. Researchers have adopted different methods such as graph analysis and complex networks modeling framework.

Chen et al. conducted the first systematic study on Ethereum [4]. They applied graph analysis to characterize the three main activities on Ethereum, Ether transaction, smart contract creation and smart contracts invocation. They devised a new way to collect the transaction data and construct three graph from the data to make analysis. In their next work, they designed a systematic data exploration framework with high-fidelity for Ethereum [17].

Ferretti et al. employed the modeling techniques of the complex network in Ethereum analysis [18]. They represented the flow of transactions happened in the blockchain as a network, where nodes are the Ethereum accounts. It has been observed that the wider the network, the greater the likelihood of a hub in the network, which means that some nodes in the blockchain are more mobile. It also can be seen how the use of blockchains changes over time.

6 Conclusion

In this paper, we analyze the behavior of users in Ethereum by using the method of node embedding and node clustering. We filter out the transactions and addresses of accounts related to the three transaction types from Ethereum blocks. Then we construct the Ethereum network to learn the eigenvectors of the account nodes based on the filtered data. And we cluster the eigenvectors of the account nodes in Ethereum. The clustering result is analyzed and draw some new opinions. At the same time, it also proves the clusterability of account nodes in Ethereum. We propose a malicious user detection method based on clustering results and the nodes with known identities. In the future work, we will continue the research and detect potential malicious users through our method and verify them.

Acknowledgments. This work is supported by: Chinese National Research Fund (NSFC) No. 61702330.

References

1. Wood, G.: Ethereum: a secure decentralised generalised transaction ledger. Ethereum Project yellow paper **151**, 1–32 (2014)
2. Nakamoto, S.: Bitcoin: a peer-to-peer electronic cash system (2008)
3. Buterin, V.: A next-generation smart contract and decentralized application platform. white paper (2014)
4. Chen, T., Zhu, Y., Li, Z., et al.: Understanding Ethereum via graph analysis. In: IEEE INFOCOM 2018-IEEE Conference on Computer Communications, pp. 1484–1492. IEEE (2018)
5. Bartoletti, M., Pompianu, L.: An empirical analysis of smart contracts: platforms, applications, and design patterns. In: Brenner, M., et al. (eds.) FC 2017. LNCS, vol. 10323, pp. 494–509. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-70278-0_31
6. Meiklejohn, S., Pomarole, M., Jordan, G., et al.: A fistful of bitcoins: characterizing payments among men with no names. In: Proceedings of the 2013 Conference on Internet Measurement Conference, pp. 127–140. ACM (2013)
7. Zhao, C., Guan, Y.: A graph-based investigation of bitcoin transactions. In: Peterson, G., Shenoi, S. (eds.) DigitalForensics 2015. IAICT, vol. 462, pp. 79–95. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24123-4_5
8. Maesa, D.D.F., Marino, A., Ricci, L.: An analysis of the bitcoin users graph: inferring unusual behaviours. In: Cherifi, H., Gaito, S., Quattrociocchi, W., Sala, A. (eds.) Complex Networks & Their Applications V. COMPLEX NETWORKS 2016. Studies in Computational Intelligence, vol. 693, pp. 749–760. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-50901-3_59
9. Reid, F., Harrigan, M.: An analysis of anonymity in the bitcoin system. In: Altshuler, Y., Elovici, Y., Cremers, A., Aharony, N., Pentland, A. (eds.) Security and Privacy in Social Networks, pp. 197–223. Springer, New York (2013). https://doi.org/10.1007/978-1-4614-4139-7_10
10. Dong, Y., Chawla, N.V., Swami, A.: Metapath2vec: scalable representation learning for heterogeneous networks. In: Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 135–144. ACM (2017)
11. Perozzi, B., Al-Rfou, R., Skiena, S.: Deepwalk: online learning of social representations. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 701–710. ACM (2014)

12. Mikolov, T., Sutskever, I., Chen, K., et al.: Distributed representations of words and phrases and their compositionality. In: Advances in Neural Information Processing Systems, pp. 3111–3119 (2013)
13. Smilkov, D., Thorat, N., Nicholson, C., et al.: Embedding projector: interactive visualization and interpretation of embeddings. arXiv preprint arXiv:1611.05469 (2016)
14. Jolliffe, I.: Principal Component Analysis. Springer, Heidelberg (2011). https://doi.org/10.1007/978-1-4757-1904-8
15. Maaten, L., Hinton, G.: Visualizing data using t-SNE. J. Mach. Learn. Res. **9**, 2579–2605 (2008)
16. Liu, H., Ruan, N., Du, R., et al.: On the strategy and behavior of bitcoin mining with N-attackers. In: Proceedings of the 2018 on Asia Conference on Computer and Communications Security, pp. 357–368. ACM (2018)
17. Chen, T., Li, Z., Zhang, Y., et al.: DataEther: data exploration framework for Ethereum. In: Proceedings of the 39th IEEE International Conference on Distributed Computing Systems (2019)
18. Ferretti, S., D'Angelo, G.: On the Ethereum blockchain structure: a complex networks theory perspective. Pract. Exp. Concurrency Comput., e5493 (2019)
19. Bok. https://www.bokconsulting.com.au/blog/ethereum-network-attackers-ip-address-is-traceable/. Accessed 25 Oct 2016
20. Latetot. https://www.reddit.com/r/ethereum/comments/55rd3j/attacker_is_gearing_up_again_for_new_spam_deluge/. Accessed Nov 2016