Early-stage Phishing Detection on the Ethereum Transaction Network

Yun Wan (wyun@smail.swufe.edu.cn)
Southwestern University of Finance and Economics
Feng Xiao
Southwestern University of Finance and Economics https://orcid.org/0000-0003-3412-5816

Research Article

Keywords: Early-stage, phishing detection, cryptocurrency, network

Posted Date: June 9th, 2022

DOI: https://doi.org/10.21203/rs.3.rs-1233747/v1

Early-stage Phishing Detection on the Ethereum Transaction Network

Yun Wan, Feng Xiao *

School of Business Administration, Faculty of Business Administration,

Southwestern University of Finance and Economics, Chengdu, China

Abstract

As cryptocurrency is widely accepted and used, illegal activities based on it have also attracted attention, especially phishing scams, which bring great losses to both customers and countries. Therefore, early-stage detection of this behavior is of great significance, as it can minimize users' losses when a phishing scam is ongoing. Existing detection methods perform effectively with all available data; however, in the early stage of a phishing scam, the performance is not satisfactory. This paper proposes the Early-stage Phishing Detection framework, which contains data processing, feature extraction, and detection components. One main contribution of this paper is the design of features based on the local graph structure and time-series attributes of the transaction network. Moreover, the phishing scam is divided into early, middle, and late stages, according to the fraud amount. Finally, the proposed method is validated on a real dataset, and the experimental results show that it can achieve the best performance. Specifically, in the early stage, the proposed method performs far better than the embedding methods. In addition, with time, it can maintain a certain degree of robustness. Thus, this paper provides useful ideas for regulators and platforms to detect phishing scams in advance.

Keywords: Early-stage, phishing detection, cryptocurrency, network

* Corresponding author. E-mail: xiaofeng@swufe.edu.cn

1 Introduction

Cryptocurrencies, based on blockchain technology, have attracted attention globally. As of March 11, 2021, the total market value of global cryptocurrencies exceeded $1.69 billion. Among them, Bitcoin accounted for 61.35%, ranking first, and Ether accounted for 12.12%, ranking second. However, in recent years, many illegal behaviors based on cryptocurrency transactions have emerged. The "Crypto Crime Report 2020," issued by the blockchain organization Chainalysis, pointed out that phishing attacks account for a high proportion of total cybercrimes, becoming one of the main attack methods for fraudulent activities. Therefore, it is necessary to design methods to detect such crimes.

Phishing is the fraudulent attempt to obtain sensitive information or data, such as usernames, passwords, and credit card details, or other sensitive information, by impersonating oneself as a trustworthy entity in a digital communication (Ramzan, 2010; Van der Merwe et al., 2005). This term can be traced back to 1996 owing to social engineering attacks against America On-line (AOL) accounts by online scammers (Khonji et al., 2013). As time progressed, phishing attacks began to shift to other targets that could make a profit, such as online banking and e-commerce. To deal with these traditional phishing attacks, researchers have conducted much research on phishing identification. In the initial stage of phishing identification, the method of detecting phishing websites was mainly based on list recognition and comparison of similarity (Han et al., 2012; Sharifi & Siadati, 2008). However, the identification technology of the list-based phishing website needs to be updated in real time, and the life cycle of the phishing website is much shorter than that of a normal website, which limits the efficiency and accuracy of this method. Therefore, some scholars have proposed heuristic methods (Jain & Gupta, 2018), but the rules defined by these methods are relatively simple and easy to be circumvented by attackers. Finally, machine learning methods are widely used in the identification of phishing websites owing to the distinguishable features of phishing websites and legitimate websites (Xi et al., 2020).

In recent years, as the value of cryptocurrencies has risen, criminals have begun to change their targets to cryptocurrencies. Phishers on cryptocurrency platforms not only use traditional web pages and emails to obtain users’ private information but also use more sophisticated technical means to defraud. For example, in July 2020, hackers gained access to more than a dozen high-profile Facebook accounts, including those of Bill Gates and Elon Musk. After taking over the accounts, the hackers posted a message, using the double return as bait to allow users to send cryptocurrency funds to the designated account address. Phishers use technology to gain the trust of the users and allow victims to send cryptocurrency directly to their accounts, which makes traditional text-based phishing detection methods no longer suitable for cryptocurrency phishing detection.

Luckily, owing to the open and transparent characteristics of blockchain technology, fully accessible transaction information has brought us new ideas and possibilities. This is actually an anomaly detection problem (Cheng et al., 2019; Hosseinzadeh et al., 2020) with supervised learning.

Many researchers have conducted network modeling analyses on public transaction data, especially through the Ethereum transaction network (T. Chen et al., 2020; Ferretti & D'Angelo, 2019; Guo et al., 2019; Lee et al., 2020; Li et al., 2020; Victor & Lüders, 2019; Zheng et al., 2020). These studies have involved building a network model for the transaction flow and then analyzing the macro and micro attributes of the network, which have provided a certain reference value for later research on phishing detection problems. Additionally, the Xblock-ETH (Zheng et al., 2020) has introduced a well-processed up-to-date dataset of Ethereum, which has promoted the further study of phishing scams of cryptocurrencies.

Just as traditional phishing detection scenarios have phishing labels of websites, phishing detection on Ethereum can exploit the labels of accounts (Q. Yuan et al., 2020). Public transaction data and available user tags make the phishing detection task a supervised classification problem. The challenge is how to obtain node features through unstructured data. Naturally, scholars use graph embedding to solve such problems. Q. Yuan et al. (2020) built an Ethereum transaction network and used node2vec (Grover & Leskovec, 2016) to extract the latent features of accounts. Chen et al. (2021) sampled the subgraph by a random walk through the neighbor relationship of the largest connected component and then combined the structural features with the embedded features based on the Graph Convolutional Network. Owing to the importance of the timestamp and transaction amount (Lin et al., 2020), Wu et al. (2020) subsequently proposed a trans2vec embedding detection method, which considers these characteristics in a walking strategy to learn users' representations. W. Chen et al. (2020) proposed a graph-based cascade feature extraction method based on a high order network and then used an Ensemble algorithm to build the identification model. Wang et al. (2021) extracted the subgraph of each address in the transaction network and then adopted graph representation methods to obtain their features. After graph embedding, these studies trained classification models based on existing label data to detect phishing accounts, which to a certain extent made up for the gap in the detection of phishing scams of cryptocurrencies.

Although previous studies have detected phishing scams effectively, the methods they used still have the following limitations: 1) the graph embedding method assumes that the representations of the connected nodes are similar and does not take into account the influence of the interaction between abnormal nodes and normal nodes; 2) as the embedding method relies on the global network structure, the entire transaction network first has to be rebuilt when new users join the transaction network; 3) most of the features learned cannot be explained well, which cannot reveal the characteristics of phishing accounts and limits the possibility of practical applications; 4) existing research has not considered the time sequence of the node appearance when classifying phishing nodes, which means it uses future transaction information to predict past node categories.

To overcome the abovementioned shortcomings, this paper presents the EPD framework. **Fig. 1** shows the framework of our study. As can be seen, the EPD framework is composed of three components: a data processing component, a feature-extract method that captures features based on the local network structure and the time series of transactions (FELT), and a phishing detection model that detects suspicious nodes. The information derived from the FELT method can be used as input for the early-stage detection task.

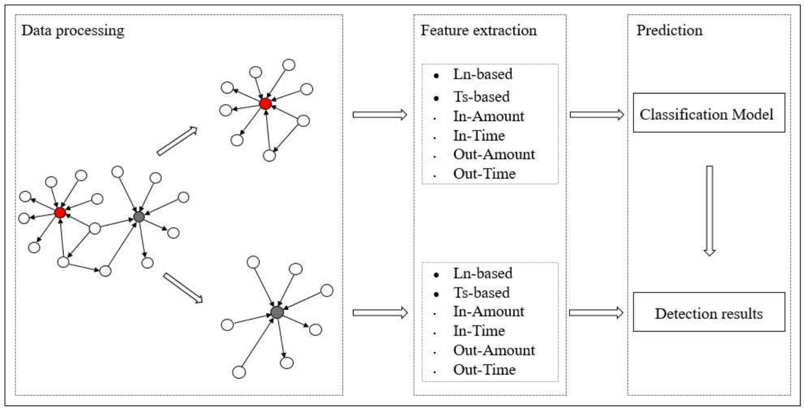

Fig. 1 An overview of the Early-stage Phishing Detection (EPD) framework

In summary, the major contributions of this study are as follows.

First, to the best of our knowledge, this is the first work that divides the phishing process into different stages, then studies the phishing detection problem.

Second, compared with existing embedding methods, the method we propose can achieve a better performance in early-stage phishing detection.

Third, as time progresses, phishing nodes may show some new behavioral characteristics. Our method has been proved to be able to adapt to new situations relatively quickly and obtain good detection results.

Fourth, the features extracted by FELT can be well explained, which can help us understand the behavior of phishing accounts and study their patterns.

The rest of this paper is organized as follows. Section 2 introduces the problem definition and data description. Section 3 describes the detection framework and baselines. The experiments conducted are detailed in Section 4. Finally, we conclude the paper in Section 5.

2 Problem definition and data description

2.1 Problem definition

On Ethereum, the transaction is the transfer of Ether from one account to another account. Therefore, we define a directed multigraph, $G(V, E, X, Y)$, of the transaction network, where V is the set of nodes, which represents the address or the account on Ethereum; E is the set of links, and $E = \{e_{ij}, i, j \in V, i \neq j\}$, which represents the transaction relationship between two addresses; X is the set of edge attributes; the edge e_{ij} has a transaction amount, a_{ij}, and a timestamp, t_{ij}, where a_{ij} refers to the transaction amount between two nodes, and $t_{ij}$is the timestamp of the transaction between them. The timestamp in cryptocurrency refers to the total number of seconds from 08:00 on January 1, 1970, Beijing time to the present. A timestamp is complete verifiable data that can indicate that a piece of data existed at a specific point in time. Y is the set of node labels, where $Y = \{y_i, i \in V\}$, $y_i = 1$ for the phishing node, and $y_i = 0$ for the normal node. Note that there may be multiple transactions between the same node pair.

Suppose there are many phishing scams at the current timestamp, T. Some accounts have been reported by victims as phishing nodes, while some continue to deceive other users; we mark them as finished phishing nodes and unfinished phishing nodes, respectively. Take **Fig. 2** as an example. There are three types of nodes in the transaction network G —the red node, gray node, and white node — representing the finished phishing node, the unfinished phishing node, and the normal node, respectively. As time progresses, the gray nodes will eventually evolve into red nodes. The purpose of this study was to detect these unfinished phishing nodes as soon as possible so as to effectively reduce users' losses.

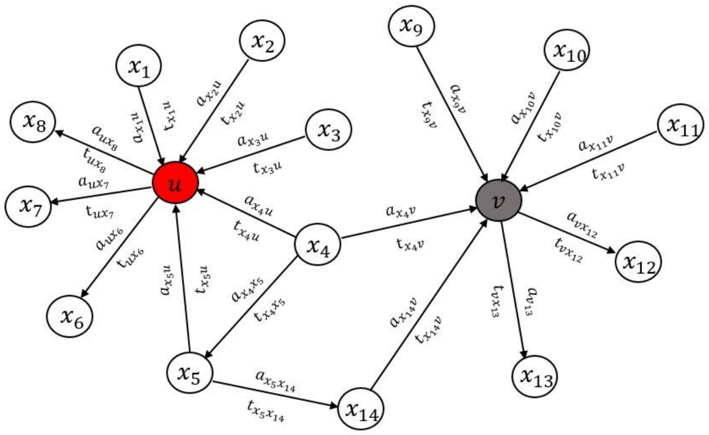

Fig. 2 Sample of transaction network

2.2 Data description

We used the dataset MulDiGraph provided by XBlock (Zheng et al., 2020), which crawls transaction data and credible label data on authoritative websites from 2016 to 2019. The network G contains 2,973,489 nodes, 13,551,303 edges, and 1,165 labeled phishing nodes.

As is well known, the characteristics and patterns of behavior are extracted from a certain number of behavior bases, and transaction behavior is no exception. Therefore, we selected the accounts whose in-degree was no less than five. In this study, 757 phishing accounts met the requirement, and the fraudulent amount accounted for 93.93% of all fraudulent amounts.

In **Table 1**, we summarize the basic statistics of these phishing accounts in detail. The total ether scammed by the phishing account reflects the user's losses, and 50% of the phishing accounts scam more than 17.64 ether each, with the largest scam even exceeding

42,000 ether, which further reveals the harmfulness of phishing scams. The number of transactions scammed by the phishing account refers to the number of transactions attracted by the phishing account. It can be seen that more than half of the phishing accounts commit fraud more than 17 times. A phishing account even scams more than 5,000 times. Fortunately, the duration of the phishing scam provides the possibility of early-stage phishing detection.

Table 1 Statistics of phishing scams

	Number of transactions scammed by the phishing account	Total ether scammed by the phishing account
Mean	47	150.4074
50%	17	17.6419
75%	32	46.2653
Max	5097	42492.7937

3 Methodology

In this section, we introduce the EPD framework and baselines of node2vec [16] and trans2vec [19], and then, we describe the evaluation criteria: accuracy, precision, recall, and F1 score.

3.1 The EPD framework

Fig. 1 shows the EPD framework, which consists of three components: data processing, feature extraction based on the FELT method, and the phishing detection process.

3.1.1 Data processing

As mentioned in Section 2.2, we select nodes with an in-degree of not less than five for research. There are 757 phishing nodes and 72,237 normal nodes that satisfy this requirement. Taking these 72,994 nodes as the central nodes, we extract the subgraph, G_s, of all the central nodes and their direct neighbors. In this study, the subnetwork contains 2,927,114 nodes, accounting for 98.44% of the original network, G.

3.1.2 Feature extraction

The feature extraction component consists of two parts: (i) extracting features based on the local network; (ii) extracting features based on the time series of the transaction amount and transaction timestamp.

Taking the target node, u , as an example, **Fig. 3** shows the whole process of the FELT method. The details are discussed in the following subsections.

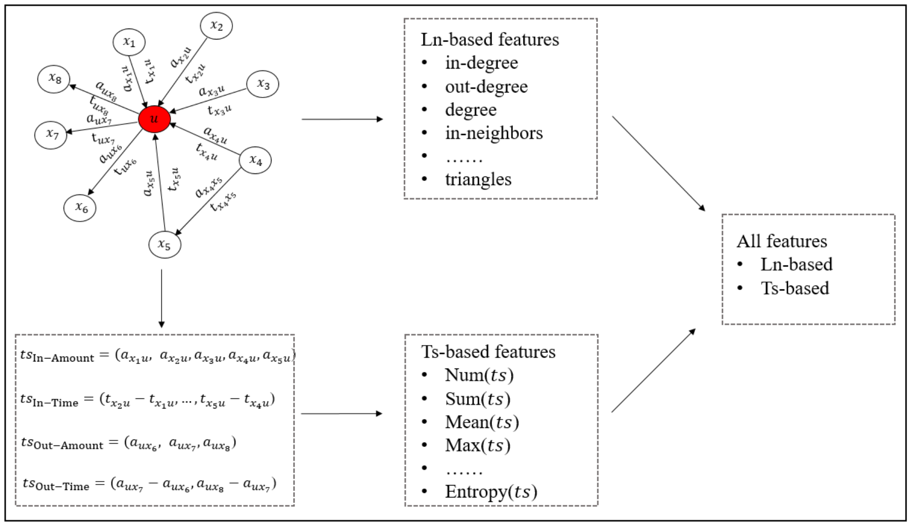
Ln-based features
in-degree
out-degree
degree
in-neighbors
……
triangles
All features
Ln-based
Ts-based
$ts_{\text{In-Amount}} = (a_{x_1u}, a_{x_2u}, a_{x_3u}, a_{x_4u}, a_{x_5u})$
$ts_{\text{In-Time}} = (t_{x_2u} - t_{x_1u}, \dots, t_{x_5u} - t_{x_4u})$
$ts_{\text{Out-Amount}} = (a_{ux_6}, a_{ux_7}, a_{ux_8})$
$ts_{\text{Out-Time}} = (a_{ux_7} - a_{ux_6}, a_{ux_8} - a_{ux_7})$
Ts-based features
Num(ts)
Sum(ts)
Mean(ts)
Max(ts)
……
Entropy(ts)

Fig. 3 The FELT method

i) Feature extraction based on the local network

In real life, a person's consumption characteristics can probably be reflected by their daily transaction flow, and if they commit a financial crime, then their trading behavior will be different from normal trading behavior. This is the same in cryptocurrency transactions. Therefor we consider neighbor nodes as the background to detect phishing users. This part extracts the typical features based on the local network of the target node.

With the target node u being the central node, we measure it in 14 dimensions in **Table 2** and mark these features as Ln-based (Local-network based) features.

Table 2 Descriptions of features based on the local network

Feature	Name	Description
Ln_1	In-degree	Number of transactions transferred to u
Ln_2	Out-degree	Number of transactions transferred by u
Ln_3	Degree	Number of transactions made by u
Ln_4	In-neighbors	Number of upstream neighbors of u
Ln_5	Out-neighbors	Number of downstream neighbors of u
Ln_6	Neighbors	Number of neighbors of u
Ln_7	In-strength	Total amount transferred to u
Ln_8	Out-strength	Total amount transferred by u
Ln_9	Strength	Total amount of transactions of u
Ln_{10}	Mean transactions of in-neighbors	Average number of transactions between u and its upstream neighbors
Ln_{11}	Mean transactions of out-neighbors	Average number of transactions between u and its downstream neighbors
Ln_{12}	Mean transactions of neighbors	Average number of transactions between u and its neighbors
Ln_{13}	Close neighbors	Number of nodes that have two-way transactions with u
Ln_{14}	Triangles	Number of triangles containing u

Ln_1, Ln_2, and Ln_3 measure the number of transactions. Note that the degree of multigraph refers to the number of transactions, including multiple transactions between the same node pair. They measure the number of in-going edges, out-going edges, and the total number of edges of the target node, respectively.

Ln_4, Ln_5, and Ln_6 measure the number of neighbors, which are different from the number of transactions. A neighbor may have multiple transactions with the target node. They measure the number of the upstream neighbors, downstream neighbors, and the total number of neighbors of the target node, respectively.

Ln_7, Ln_{18}, and Ln_9 consider the transaction amount. They measure the total transaction amount of upstream neighbors, downstream neighbors, and all neighbors of the target node, respectively.

Ln_{10}, Ln_{11}, and Ln_{12} measure the average number of transactions between the same node pair. The expressions are as follows:

$$Ln_{10} = \frac{Ln_1}{Ln_4} \tag{1}$$

$$Ln_{11} = \frac{Ln_2}{ln_5} \tag{2}$$

$$Ln_{12} = \frac{Ln_3}{Ln_6} \tag{3}$$

Owing to the specificity of phishing behavior, the phishing node may only conduct a single transaction with each of its neighbors, while transactions of normal nodes will have the characteristics of continuity and repetition, which makes their average number of transactions greater than 1.

We define Ln_{13} as "close neighbors," which represents the number of neighbors that conduct two-way transactions with the target node. It is believed that friends who interact with each other will be closer than those who do not; therefore, we define this feature to reveal the intimacy between the target node and its neighbors.

Ln_{14} is the number of triangles containing the target node in the transaction network. Essentially, the triangle means that there are edges between the neighbors of the target node. It is well known in sociology that the triangle is a symbol of a stable relationship, which reveals the stability between the target node and its neighbors. Here, we ignore the number of transactions and the direction between node pairs.

ii) Feature extraction based on time series of transactions

In addition to the network structure, the transaction network also has the characteristics of time series. In other words, the transactions of the target node are sequential. We define in-transactions as all transactions from other nodes to the target node, and out-transactions as all transactions from the target node to other nodes. Considering the transaction direction and amount, each target node has four sequences. Taking the target node u in **Fig. 2** as an example, in-transactions and out-transactions are arranged in time order. We assume that timestamp $t_{x_1u} < t_{x_2u} < t_{x_3u} < t_{x_4u} < t_{x_5u}$ and $t_{ux_6} < t_{ux_7} < t_{ux_8}$. The four time series, ts, are as follows:

$$ts_{In-Amount} = (a_{x_1u},\ a_{x_2u},\ a_{x_3u}, a_{x_4u},\ a_{x_5u})$$

$$ts_{In-time} = (t_{x_2u} - t_{x_1u}, \dots, t_{x_5u} - t_{x_4u})$$

$$ts_{Out-Amount} = (a_{ux_6},\ a_{ux_7}, a_{ux_8})$$

$$ts_{Out-Time} = (a_{ux_7} - a_{ux_6}, a_{ux_8} - a_{ux_7})$$

$ts_{In-Amount}$is the amount-based time series of in-transactions, and $ts_{In-time}$ is the time difference series of in-transactions. Similarly, $ts_{Out-Amount}$ is the amount-based time series of out-transactions, and $ts_{Out-time}$ is the time difference series of out-transactions.

After obtaining ts, we calculate their statistical properties to reveal the transaction characteristics of the target node, as shown in **Table 3**.

Table 3 Descriptions of features based on time series

Feature	Name	Description
$F_{ts,1}$	number	Number of ts
$F_{ts,2}$	sum	Sum of ts
$F_{ts,3}$	mean	Mean value of ts
$F_{ts,4}$	max	Max value of ts
$F_{ts,5}$	min	Minimal value of ts
$F_{ts,6}$	median	Median value of ts

$F_{ts,1}$ is the number of ts, for amount-based time series like $ts_{In-Amount}$ and $ts_{Out-Amount}$. $F_{ts,1}$ is the number of in-transactions and out-transactions, which equals the in-degree and the out-degree of the transaction network; for time-based time series like $ts_{In-Time}$ and $ts_{Out-Time}$, $F_{ts,1}$ is the number of in-transactions and out-transactions minus 1. Therefore, this feature already exists in the Ln-based features, but it is considered here because of the need for a separate comparison in the following experiments.

$F_{ts,2}$ is the sum of ts, for amount-based time series like $ts_{In-Amount}$ and $ts_{Out-Amount}$. $F_{ts,2}$ is the total amount of in-transactions and out-transactions, which equals the in-strength and the out-strength of the transaction network; for time-based time series like $ts_{In-Time}$ and $ts_{Out-Time}$, $F_{ts,2}$ is the time span of in-transactions and out-transactions.

$F_{ts,3}$, $F_{ts,4}$, $F_{ts,5}$, $F_{ts,6}$, and $F_{ts,7}$ are the mean value, the maximum value, the median value, the minimal value, and the standard deviation of ts, respectively.

Among them, the $F_{ts,8}$ skew reflects the asymmetry of the time-series distribution, and its calculation is as follows:

$$F_{ts,8} = \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2}{\left[\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2\right]^{\frac{3}{2}}}, \tag{1}$$

where x_i is the element of ts, $\bar{x}$ is the mean of samples, and n is the number of samples in ts. When the skewness is less than 0, most of the values are on the right side of the average value; when it is greater than 0, most of the values are on the left side of the average value; and when it is about 0, the values are evenly distributed on both sides of the average value.

$F_{ts,9}$ kurt measures the kurtosis of the probability distribution of a real random variable, and its calculation is as follows:

$$F_{ts,9} = \frac{\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^4}{\left[\frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})^2\right]^{2}} - 3\,. \tag{2}$$

When kurtosis is less than 0, the distribution is smoother than the normal distribution; when it is greater than 0, the distribution is steeper than the normal distribution; and when it is about 0, the distribution is close to the normal distribution. For some nodes that have a relatively small number of out-transactions, their kurtosis cannot be calculated, and we set the kurtosis to 0 in this case.

$F_{ts,10}$ entropy measures the uncertainty of ts. We perform equidistant binning operations on the value of ts, and it can be calculated as follows:

$$F_{ts,10} = -\sum_{k=0}^{\min(bin,len(ts))} p_k \, lnp_k, \tag{3}$$

where bin is the number of subintervals of ts; $len(ts)$ is the number of ts; and p_k represents the proportion of the value of ts falling in the k-th subinterval. In this study, we set bin to 5. The smaller the value of entropy, the more uniform the distribution of the time series.

As shown in **Fig. 3**, to further compare the contribution of these features to the detection task, we mark the features of $ts_{In-Amount}$, $ts_{In-Time}$, $ts_{Out-Amount}$, and $ts_{Out-Time}$ as

$In\_Amount, In\_Time, Out\_Amount,$ and Out_Time, respectively, and each feature as "$In/Out\_A/T$" plus the feature number. For example, the features of In_Amount are marked as In_A1, In_A2, …, In_A10. Moreover, all the features are denoted as Ts-based (Time-series based) features. Finally, we combine the Ts-based features with the Ln-based features.

3.1.3 Phishing detection process

As shown on the right side of **Fig. 1**, after data processing and feature extraction, we adopt the logistic regression (LR) method to detect the suspicious nodes. Unlike previous researchers, we divide the phishing node into different stages for further research.

3.2 Feature learning by graph embedding methods

Compared to the EFL method, our framework uses two graph embedding methods, node2vec [16] and trans2vec [19], as benchmarks. The reasons for choosing these two methods for comparison are as follows.

First, the transaction network has a natural graph structure, and most of the existing research is based on graph embedding algorithms. Second, node2vec is one of the classic graph embedding algorithms, which considers the weight and the walking direction at the same time. Third, based on the walk of the amount value and timestamp, trans2vec is an improved embedding method of node2vec, which achieves the best performance in the phishing detection task on Ethereum transaction network.

Note that both of these embedding methods should first process the original network G. Merging transactions between the same node pair, and retaining the sum of the transaction amount and the timestamp of the last transaction, we simplify the multigraph into a simple graph. Then, with all the target nodes being the central nodes, we extract their neighbors and all the connected edges between all of them to form a subnetwork. Finally, we learn the representations of the nodes in the subnetwork through graph embedding methods.

3.3 Performance evaluation metrics

To perform downstream phishing detection tasks after obtaining the features of target nodes, we consider four evaluation indicators in **Table 4** as follows:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}, \tag{4}$$

$$\text{Precision} = \frac{\text{TP}}{TP + FP}, \tag{5}$$

$$\text{Recall} = \frac{TP}{TP + FN}, \tag{6}$$

$$\text{F1} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}, \tag{7}$$

where TP is true positive, TN is true negative, FP is false positive, and FN is false negative.

Table 4 Confusion matrix

		Predicted Class	
		Phishing	Normal
Actual Class	Phishing	TP	FN
	Normal	FP	TN

4 Experiment

4.1 Experiment settings

Previous research (Wu et al., 2020; Z. Yuan et al., 2020) marked the phishing accounts as positive samples and then randomly selected normal nodes with the same number of positive samples as the negative samples. However, as mentioned in Section 1, they ignored the sequence of occurrence of phishing scams and thus used future transaction information to predict the types of nodes in the past. To fit our task, we modified the dataset through the following two steps:

First, given the current timestamp, T, we extracted the transaction network, G_T, where all transactions have happened before T. Then, we marked the finished phishing nodes as positive samples, randomly selected normal nodes with the same number of positive samples as negative samples, and then took 80% and 20% of these data as the training set and the validation set, respectively. Similarly, we marked the unfinished phishing nodes and the testing set.

In order to ensure the robustness of the experiment results, we repeated the random selection procedure of normal nodes 100 times in each experiment. Finally, the LR model was trained to classify the target nodes, and the average metrics were reported.

For embedding methods, we set the parameters according to the guidance given by Wu et al. (2020) as follows: embedding size $d = 64$, walks per node $r = 20$, walk length $l = 5$, and context size $k = 10$, $p = 0.25$, q= 0.75 for node2vec; $\alpha = 0.5$ for trans2vec.

4.2 Feature selection

Feature selection plays a huge role in building machine learning models. As some features are highly correlated, we used the filter method to remove these features. First, we computed the correlation coefficient of each feature pair and plotted them, as shown in **Fig. 4**. Note that we did not use the information of the testing set here. Then, we compared the correlation and removed one of the feature pairs whose correlation was higher than 0.9. We can observe that the adjacent features have a higher correlation. Finally, we chose 40 features to conduct the following experiments.

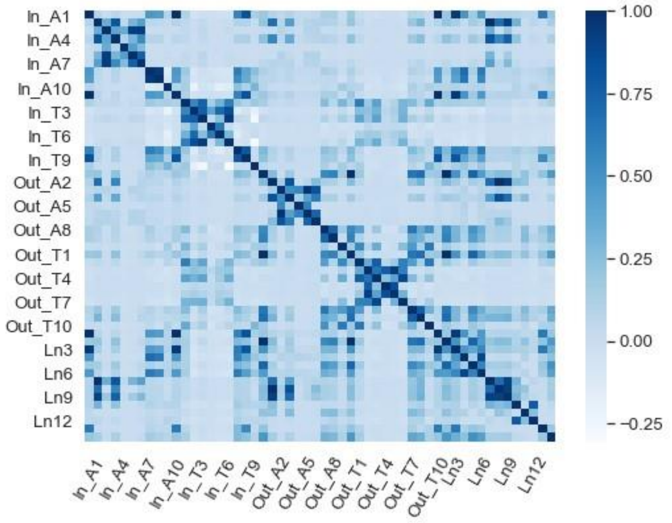

Fig. 4 Heatmap of correlation coefficient of features

4.3 Phishing detection

We assumed that the current time was 2018.07.08 23:59:59 and the corresponding timestamp was 1,531,065,599. There were 606 finished phishing nodes and 90 unfinished phishing nodes.

Table 5 presents an overview of the performance of different methods on phishing detection.

Table 5 Performance comparisons of the FELT method and embedding methods with LR

Method	Accuracy	Precision	Recall	F1	Time (s) [a]
n2v	0.7400	0.7662	0.6948	0.7277	9,351
t2v	0.7096	0.7547	0.6249	0.6825	13,330
Ts-based	0.6998	0.6830	0.7528	0.7152	/
Ln-based	0.7283	0.6866	0.8606	0.7541	/
FELT	0.7906	0.7687	0.8337	0.7987	6,755

[a] Time here refers to the time each method took to learn the features.

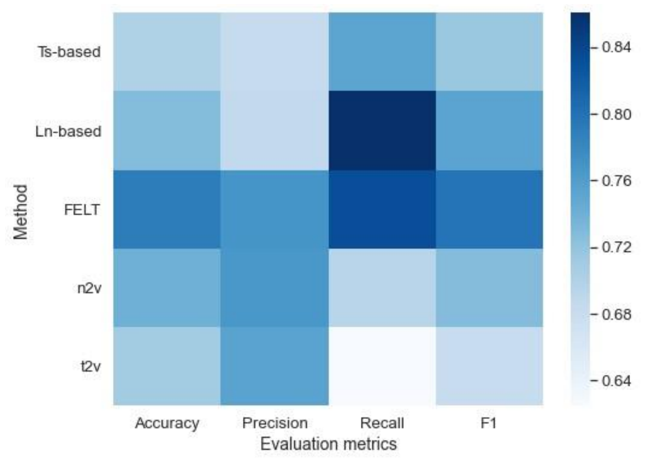

Fig. 5 Heatmap of performance comparisons of different methods

As shown in **Fig. 5**, the highlighted area of FELT is significantly darker than that of n2v and t2v, which indicates that the FELT method is superior to the embedding methods in all evaluation metrics. Moreover, the Ln-based features contribute more than the Ts-based features, which indicates the importance of the local structure.

Note that n2v performs better than t2v in all metrics. We speculate that this is because the random walk of t2v depends on the timestamp and amount, which depend on the global network. However, this paper focuses on using past information to predict unfinished phishing scams; so, t2v is unable to obtain enough transaction information. On the contrary, the FELT method focuses on the transaction behavior of the target node, which will be reflected during the fraud process. Finally, based on the random walk and skip-gram, embedding methods are more time-intensive than the FELT method. All of this proves that the FELT method is not only accurate but also efficient.

Furthermore, the selected classifier is also an important factor. We chose Naive Bayes, K-Nearest Neighbors (KNN), and Support Vector Machine (SVM) as baselines. Using the 40 features of the FELT method as input features, the detection performance of different classifiers was compared in

Table 6. According to the results, the LR model performs best in all metrics. Therefore, we selected it as the classifier in this study, with $C = 0.9$ and $solver = 'liblinear'$, where C is the inverse of regularization strength, and solver is the algorithm used in the optimization problem.

Table 6 Performance comparisons of different classifiers

	Accuracy	Precision	Recall	F1
KNN	0.7859	0.8117	0.7484	0.7779
SVM	0.7800	0.7872	0.7706	0.7783
Naive Bayes	0.7542	0.7661	0.7337	0.7486
LR	0.7906	0.7687	0.8337	0.7987

4.4 Early-stage phishing detection

To detect the unfinished phishing nodes as soon as possible, we divided the phishing scam into different stages according to the $process\_ratio$, which is the ratio of the accumulated fraud amount at the current timestamp to the total fraud amount. To ensure that there were as many abnormal nodes as possible in each stage, we divided the phishing scam into three stages, as shown in **Table 7.**

Table 7 The three stages of a phishing scam

$process\_ratio$	Phishing stage	Number of unfinished phishing nodes
(0, 33%]	Early	23
(33%, 66%]	Middle	11
(66%, 100%)	Late	56

Table 7 lists the number of unfinished phishing nodes in each stage. We performed separate detection following the experiment in the previous part. **Table 8** presents the performance comparisons, and **Fig. 6** shows the F1 metric of different methods.

Table 8 Performance comparisons of different stages of the phishing scams with LR

Stage	Method	Accuracy	Precision	Recall	F1
Early-stage	n2v	0.5957	0.6129	0.5330	0.5654
	t2v	0.5550	0.5760	0.4435	0.4961
	FELT	0.7398	0.7175	0.7991	0.7549
Middle-stage	n2v	0.7936	0.7867	0.8191	0.7986
	t2v	0.7950	0.8055	0.7882	0.7934
	FELT	0.7973	0.7854	0.8300	0.8043
Late-stage	n2v	0.7753	0.8069	0.7298	0.7646
	t2v	0.7472	0.7999	0.6641	0.7236
	FELT	0.8195	0.7882	0.8777	0.8295

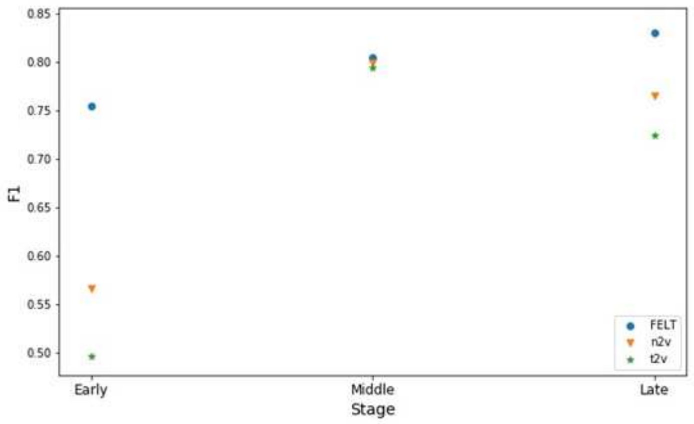

Fig. 6 Performance comparisons of different stages

We can observe that regardless of the stage, our method can achieve the best detection results, especially in the early stage, which shows that the behavioral characteristics of the phishing scam can be captured by the features we proposed. In addition, as the stage progresses, the detection performance based on our method improves, but the results based on the embedding methods experience great fluctuations. We speculate that the phishing nodes, in the late stage, may transfer fraudulent funds or conduct some normal transaction behaviors to camouflage themselves, which may make their local structures more complicated.

To further analyze these features on early-stage phishing detection, the recursive feature elimination (RFE) method was adopted to select the 10 most important features. We set the LR model as the external estimator. First, the estimator was trained on the initial set of features, and the coefficient of each feature was obtained; then, the least important feature was deleted from the set of features. This procedure was repeated until the number of remaining features was eventually reached. After filtering, the 10 most important features were obtained, as shown in **Table 9.**

Table 9 Top 10 important features

Rank	Feature
1	Mean transactions of in-neighbors
2	Mean of $ts_{In-Time}$
3	Mean transactions of neighbors
4	Number of $ts_{In-Amount}$
5	Min of $ts_{In-Time}$
6	Skewness of $ts_{In-Time}$
7	Sum of $ts_{In-Time}$
8	Min of $ts_{In-Amount}$
9	Skewness of $ts_{In-Amount}$
10	Mean of $ts_{In-Amount}$

It can be seen from **Table 9** that the 10 most important features are related to the local network and in-transactions. For Ln-based features, the mean transactions of in-neighbors and the mean transactions of neighbors are important. This proves our conjecture in Section 3.1.2 that the phishing nodes generally only conduct a single transaction with the victim, which makes it different from normal nodes. For Ts-based features, the features of $ts_{In-Time}$ are essential, such as the mean of $ts_{In-Time}$, the min of $ts_{In-Time}$, the skewness of $ts_{In-Time}$, and the sum of $ts_{In-Time}$, which indicates the importance of the transaction frequency to phishing detection. In addition, the distribution of $ts_{In-Amount}$ also contributes greatly to detection results, such as the number of $ts_{In-Amount}$, the min of $ts_{In-Amount}$, the skewness of $ts_{In-Amount}$, and the mean of $ts_{In-Amount}$.

These features may help regulatory authorities set thresholds for real-time monitoring and preliminary screening of suspicious nodes. However, we will not analyze this topic further here.

4.5 Early-stage phishing detection at different *T* values

As time, T, progressed, there were more finished phishing nodes. Would they provide more information to help us improve the detection effect? Would the newly emerging phishing scams show different behavior patterns, reducing the detection effect of our method?

To answer these questions, we set different T and defined the $size$ as the proportion of finished phishing nodes of G_T to all finished phishing nodes in G_s. **Table 10** summarizes the statistics. The date is the Beijing time corresponding to each Timestamp T, N_1 is the number of finished phishing nodes, N_2 is the number of unfinished phishing nodes at the early stage, and G_T is the number of nodes. We can find that N_1 grows rapidly, which reflects the urgency of early-stage phishing detection.

Table 10 Statistics of different T values

T	Timestamp	Date	$size$	N_1	N_2
T_1	1517327999	2018.01.30 23:59:59	20%	151	20
T_2	1523807999	2018.04.15 23:59:59	40%	303	23
T_3	1527004799	2018.05.22 23:59:59	60%	454	32
T_4	1531065599	2018.07.08 23:59:59	80%	606	23

The detection results are compared in **Table 11**. We can observe that the evaluation metrics of different methods do not show a fixed trend with the increase of T. Nevertheless, the FELT method performs better than other methods at different T values.

Table 11 Performance comparisons of different T values with LR at the early stage

T	Method	Accuracy	Precision	Recall	F1
	n2v	0.5988	0.6106	0.5500	0.5751
T_1	t2v	0.5717	0.5748	0.5555	0.5631
	FELT	0.7602	0.7212	0.8525	0.7795
	n2v	0.6183	0.6405	0.5504	0.5906
T_2	t2v	0.5652	0.5746	0.5130	0.5404
	FELT	0.6811	0.6676	0.7330	0.6971
	n2v	0.5623	0.5796	0.4700	0.5156
T_3	t2v	0.5481	0.5621	0.4572	0.5019
	FELT	0.7302	0.7104	0.7816	0.7428
	n2v	0.5957	0.6129	0.5330	0.5654
T_4	t2v	0.5550	0.5760	0.4435	0.4961
	FELT	0.7398	0.7175	0.7991	0.7549

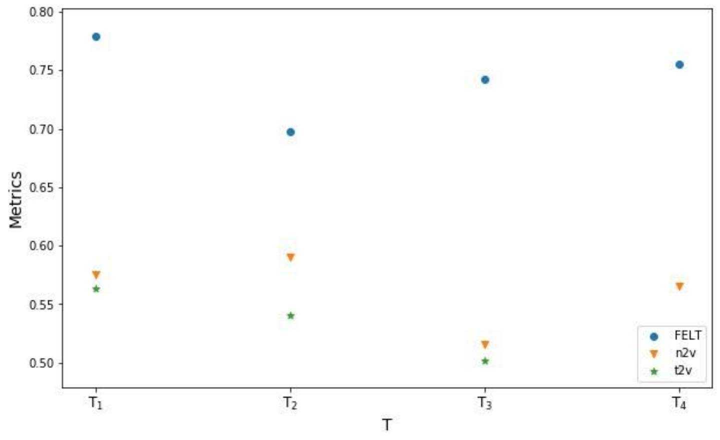

Fig. 7 Performance comparisons of different T values

It can be seen from **Fig. 7** that the F1 of the FELT method decreases at the timestamp T_2 and then increases at T_3 and T_4. A possible explanation for this decline may be that the newly phishing nodes show some new behavior characteristics at T_2, while the finished phishing nodes do not provide additional information about these behavior patterns. As pointed out in the report of “Spam and phishing in Q2 2018,” 1 provided by Kaspersky Lab, a Russian multinational cybersecurity and anti-virus provider, in the second quarter of 2018, criminals tried various new methods to entice a victim to willingly send them cryptocurrency, with one of the examples of this being cryptocurrency giveaways. The transaction patterns of this type of behavior are different from those caused by criminals obtaining private information from users, which increases the difficulty of detection. Luckily, these newly phishing nodes develop into finished phishing nodes at T_3, and their transaction features are learned to help us conduct detection. Therefore, the metrics rise significantly at T_3 and T_4.

However, it can also be seen from **Fig. 7** that the detection performance of the t2v method decreases linearly with time, and the detection performance of the n2v method also fluctuates greatly. We speculate that the reliance on the structure of the embedding method causes the learned representation to be inaccurate.

All of this proves the adaptability and robustness of our method.

5 Conclusion and future work

A three-component framework was proposed to detect phishing nodes using the features extracted from the local network and time series of transactions. Experiments on the Ethereum transaction network proved the effectiveness and robustness of the FELT method. Through these analyses, we obtained many new observations and findings, which can help us gain a deeper understanding of phishing behavior in a transaction network.

First, the features of the local network are more critical than that of time series, which indicates that we should pay more attention to the local structure of accounts. Among the features of time series, the features of transaction frequency play an important role. Second, the transactions of phishing nodes in the early stage will expose their transaction behavior patterns, which helps us to carry out early-stage phishing detection. Third, as time T progresses, new phishing scams may show different behavior patterns, but we can quickly capture their characteristics and effectively detect subsequent phishing nodes.

Finally, this research should continue to advance in at least the following two aspects: 1) in terms of data, the current research focused on Ethereum transaction data. With the

[1] https://securelist.com/spam-and-phishing-in-q2-2018/87368/

development of cryptocurrency, it is necessary to try to use more cryptocurrency data for further verification; 2) there are many illegal and criminal behaviors in cryptocurrency transactions. This paper only focused on phishing behavior. Follow-up research should pay attention to the similarities and differences of various illegal behaviors.

Declarations

Funding

The work described in this paper was supported by the National Natural Science Foundation of China (72001152), the Fundamental Research Funds for the Central Universities (JBK2103009), the fund of Financial Innovation Center, SWUFE and the achievements transformation projects of SWUFE Jiaozi Institute of Fintech Innovation.

Conflicts of interest

The authors have no relevant financial or non-financial interests to disclose.

Authors' contribution

The initial idea was first proposed by Yun Wan. Material preparation, data collection and analysis were performed by Yun Wan and Feng Xiao. The first draft of the manuscript was written by Yun Wan and all authors commented on previous versions of the manuscript. All authors read and approved the final manuscript.

Data Availability

The dataset analysed during the current study is available in the XBlock, one of the blockchain data platforms in the academic community. http://xblock.pro/tx/.

References

Chen, L., Peng, J., Liu, Y., Li, J., Xie, F., & Zheng, Z. (2021). Phishing Scams Detection in Ethereum Transaction Network. *ACM Transactions on Internet Technology*, *21*(1), 1-16. https://doi.org/10.1145/3398071

Chen, T., Li, Z., Zhu, Y., Chen, J., Luo, X., Lui, J. C.-S., Lin, X., & Zhang, X. (2020). Understanding Ethereum via Graph Analysis. *ACM Transactions on Internet Technology*, *20*(2), 1-32. https://doi.org/10.1145/3381036

Chen, W., Guo, X., Chen, Z., Zheng, Z., & Lu, Y. (2020). Phishing Scam Detection on Ethereum Towards Financial Security for blochchain Ecosystem. *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*.

Cheng, Q., Zhou, Y., Feng, Y., & Liu, Z. (2019). An unsupervised ensemble framework for node anomaly behavior detection in social network. *Soft Computing*, *24*(9), 6421-6431. https://doi.org/10.1007/s00500-019-04547-6

Ferretti, S., & D'Angelo, G. (2019). On the Ethereum blockchain structure: A complex networks theory perspective. *Concurrency and Computation: Practice and Experience*, *32*(12). https://doi.org/10.1002/cpe.5493

Grover, A., & Leskovec, J. (2016). node2vec: Scalable Feature Learning for Networks. *KDD*, *2016*, 855-864. https://doi.org/10.1145/2939672.2939754

Guo, D., Dong, J., & Wang, K. (2019). Graph structure and statistical properties of Ethereum transaction relationships. *Information Sciences*, *492*, 58-71. https://doi.org/10.1016/j.ins.2019.04.013

Han, W., Cao, Y., Bertino, E., & Yong, J. (2012). Using automated individual white-list to protect web digital identities. *Expert Systems with Applications*, *39*(15), 11861-11869.

Hosseinzadeh, M., Rahmani, A. M., Vo, B., Bidaki, M., Masdari, M., & Zangakani, M. (2020). Improving security using SVM-based anomaly detection: issues and challenges. *Soft Computing*, *25*(4), 3195-3223. https://doi.org/10.1007/s00500-020-05373-x

Jain, A. K., & Gupta, B. B. (2018). Two-level authentication approach to protect from phishing attacks in real time. *Journal of Ambient Intelligence and Humanized Computing*, *9*(6), 1783-1796.

Khonji, M., Iraqi, Y., & Jones, A. (2013). Phishing Detection: A Literature Survey. *IEEE Communications Surveys & Tutorials*, *15*(4), 2091-2121. https://doi.org/10.1109/surv.2013.032213.00009

Lee, X. T., Khan, A., Sen Gupta, S., Ong, Y. H., & Liu, X. (2020). *Measurements, Analyses, and Insights on the Entire Ethereum Blockchain Network* Proceedings of The Web Conference 2020,

Li, Y., Akcora, U. I. C., Smirnova, E., Gel, Y. R., & Kantarcioglu, M. (2020). Dissecting Ethereum Blockchain Analytics What We Learn from Topology and Geometry of the Ethereum Graph. *Proceedings of the 2020 SIAM International Conference on Data Mining*

Lin, D., Wu, J., Yuan, Q., & Zheng, Z. (2020). Modeling and Understanding Ethereum Transaction Records via a Complex Network Approach. *IEEE Transactions on Circuits and Systems II: Express Briefs*, *67*(11), 2737-2741. https://doi.org/10.1109/tcsii.2020.2968376

Ramzan, Z. (2010). Phishing attacks and countermeasures. *Handbook of information and* *communication security*, 433-448.

Sharifi, M., & Siadati, S. H. (2008). A phishing sites blacklist generator. 2008 IEEE/ACS international conference on computer systems and applications,

Van der Merwe, A., Loock, M., & Dabrowski, M. (2005). Characteristics and responsibilities involved in a phishing attack. Proceedings of the 4th international symposium on Information and communication technologies,

Victor, F., & Lüders, B. K. (2019). Measuring Ethereum-Based ERC20 Token Networks. In *Financial Cryptography and Data Security* (pp. 113-129). https://doi.org/10.1007/978-3-030-32101-7_8

Wang, J., Chen, P., Yu, S., & Xuan, Q. (2021). TSGN Transaction Subgraph Networks for identifying Ethereum Phishing Accounts. *arXiv preprint arXiv:2104.08767*.

Wu, J., Yuan, Q., Lin, D., You, W., Chen, W., Chen, C., & Zheng, Z. (2020). Who Are the Phishers? Phishing Scam Detection on Ethereum via Network Embedding. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 1-11. https://doi.org/10.1109/tsmc.2020.3016821

Xi, F., Hui, L., & Xingwen, Z. (2020). Survey on phishing detection research. *Chinese Journal of Netword and Information Security*. https://doi.org/10.11959/j.issn.2096−109x.2020062

Yuan, Q., Huang, B., Zhang, J., Wu, J., Zhang, H., & Zhang, X. (2020). Detecting Phishing Scams on Ethereum based onTransaction Records. *IEEE International Symposium on Circuits and Systems (ISCAS)*.

Yuan, Z., Yuan, Q., & Wu, J. (2020). Phishing Detection on Ethereum via Learning Representation of Transaction Subgraphs. In *Blockchain and Trustworthy Systems* (pp. 178-191). https://doi.org/10.1007/978-981-15-9213-3_14

Zheng, P., Zheng, Z., Wu, J., & Dai, H.-N. (2020). Xblock-ETH: Extracting and exploring blockchain data from Ethereum. *IEEE Open Journal of the Computer Society*, *1*, 95-106.