A Graph Learning based Approach for Identity Inference in DApp Platform Blockchain

Xiao Liu, Zaiyang Tang, Peng Li, *Member, IEEE*, Song Guo, *Fellow, IEEE*, Xuepeng Fan, Jinbo Zhang,

Abstract—Current cryptocurrencies, such as Bitcoin and Ethereum, enable anonymity by using public keys to represent user accounts. On the other hand, inferring blockchain account types (i.e., miners, smart contracts or exchanges), which are also referred to as blockchain identities, is significant in many scenarios, such as risk assessment and trade regulation. Existing work on blockchain deanonymization mainly focuses on Bitcoin that supports simple transactions of cryptocurrencies. As the popularity of DApp platform blockchains with Turing-complete smart contracts, represented by Ethereum, identity inference in blockchain faces new challenges because of user diversity and complexity of activities enabled by smart contracts. In this paper, we propose I^{2}GL, an identify inference approach based on big graph analytics and learning to address these challenges. Specifically, I^{2}GL constructs a transaction graph and aims to infer the identity of nodes using the graph learning technique based on Graph Convolutional Networks. Furthermore, a series of enhancement has been proposed by exploiting unique features of blockchain transaction graph. The experimental results on Ethereum transaction records show that I^{2}GL significantly outperforms other state-of-the-art methods.

Index Terms—blockchain, anonymity, identity inference, graph learning

1 Introduction

Blockchain is essentially an immutable public ledger that is maintained by a number of distributed nodes connected by a peer-to-peer network. Blockchain has gained massive momentum in recent years, mainly due to the success of cryptocurrencies like Bitcoin. A key feature of cryptocurrencies is the anonymity. Public keys are used as user account IDs, so that transactions can be done pseudonymously, without exposing their real identities (i.e., miners, smart contracts or exchanges) [1]. On the other hand, deanonymization, i.e., inferring identities of blockchain users, also attracts wide attention. By identifying and tracking these anonymous users, people like traders, law-makers, financial security officers, etc., can have a better understanding of what is occurring on the blockchain. Due to its significance, identify inference on Bitcoin has been extensively studied, ranging from usual trading detection [2] to exchange pattern mining [3].

New challenges of identity inference arise as the popularity of Ethereum, which is a blockchain-based software platform that enables smart contracts [4]. With Turing-complete smart contracts, users can perform not only transactions of cryptocurrencies but also arbitrary behaviors on the blockchain. Various decentralized applications (DApps), e.g., CryptoKitties [5] and IDEX [6], based on smart contracts have been developed and they generate new roles in the blockchain ecosystems. Therefore, identity inference becomes more challenging in Ethereum because of the diversity of users and the complexity of their activities implemented by smart contracts. For example, there are more than 2500 accounts labeled as "Phish/Hack" in Etherscan [7], which are fraud addresses related to phishing or hacking in Ethereum. Many of these accounts are disguised as ICO (Initial Coin Offering) wallets or DApps like casino [8], and it is hard to detect them without source code analysis [9]. Existing work [2], [3], [10] have studied discerning transaction flows for currency-trade blockchains, like Bitcoin, which is far from achieving successful identity inference in identity-rich blockchains like Ethereum. A recent work [11] infers the account identity by analyzing source codes (e.g., comments or keywords) of smart contracts to acquire information of internal transactions. However, it relies on experts to exhaustively enumerate all related features, which is tedious, time-consuming and inefficient.

In this paper, we propose an *Identity Inference approach by Graph deep Learning (I^{2}GL)* for Ethereum and other similar DApp platform blockchains. We define the blockchain identities as the account types, e.g., miners, smart contracts, exchanges, and phish/hack. The basic idea of I^{2}GL is to construct a transaction graph, where nodes denote user accounts and edges describe transactions among them. The identities of some nodes in the transaction graph are unknown. The goal of I^{2}GL is to infer the identities of these unknown nodes by exploiting graph features.

Graph analytics has been extensively studied for social media networks [12] and knowledge graphs [13]. However, transaction graphs of blockchains have many unique features that make the graph analytic more challenging. First, the Ethereum transaction graph can be huge. For example, a graph used in our experiments contains 16,599,825 nodes and 116,293,867 edges while the *Citeseer*, a typical dataset of citation network, contains only 3,327 nodes and 4,732 edges. Thus, most existing methods suffer from high computation and storage overhead on the analytics of blockchain transaction graphs. Second, there exist multiple types of activities, e.g., money transfer, contract creation, and invocation,

- *X. Liu and J. Zhang are with Peking University, China. E-mail: {lxlisa, jinbozhang}@pku.edu.cn.*
- *Z. Tang and X. Fan are with ASResearch, China. E-mail: {longinus41, athrunarthur}@gmail.com.*
- *P. Li is with the University of Aizu, Japan. E-mail: pengli@u-aizu.ac.jp.*
- *S. Guo is with the Hong Kong Polytechnic University, China. E-mail: song.guo@polyu.edu.hk.*

among nodes in Ethereum. Such a kind of activity diversity is critical for identity inference and should be considered in graph analytics. Third, we observe that even though two types of accounts conduct the same number of transactions during a period, the transaction density over time is different. This fact indicates that the temporal feature of transactions is also important for identity inference, which unfortunately cannot be exploited by existing methods.

In I^{2}GL, graph learning has been used to address the challenges by converting large transaction graphs into a low dimensional space, in which the graph information is still preserved [14]. Specifically, I^{2}GL adopts Graph Convolutional Network (GCN), a powerful learning technique based learning method designed to work directly on graphs by leveraging the graph structure as convolutional layers. To describe diverse activities among nodes, we create multiple edges between each pair of nodes in the transaction graph, each of which represents a type of activities, instead of using a single weighted edge to describe node relationship in traditional graphs. Furthermore, I^{2}GL enhances GCN by integrating the transaction density feature into the graph learning process. Other features, such as second-order proximity and asymmetric proximity, in the cryptocurrency transaction graph are also considered by I^{2}GL. The main contributions of this paper are summarized as follows.

- To the best of our knowledge, we are the first to apply graph analytics and learning techniques for identity inference on the blockchain enabling smart contracts.
- Motivated by unique features of blockchain transaction graphs, we propose several novel designs to enhance the performance of traditional GCN.
- Experiments on Ethereum transaction records are conducted to evaluate the performance of I^{2}GL. The results show that our proposal significantly outperforms existing approaches.

The rest of the paper is organized as follows. Section 2 introduces the background knowledge of cryptocurrencies and raises the identity inferring problem. In Section 3, an overview of our analysis framework is first given, and then detailed designs are presented. Section 4 presents experimental results. Related work is reviewed in Section 5, followed by Section 6 that concludes this paper.

2 Preliminary

In this section, we present some preliminaries related to smart contracts and user identities in Ethereum, which are necessary for understanding the challenges of identity inference and our proposed approach.

2.1 Smart Contracts

Although Bitcoin is the most striking example of realizing the concept of a decentralized cryptocurrency system, it does not support complex programs. With the development of blockchain technology, competitive successors have emerged and extended blockchain functions to support various decentralized applications. The most significant one is Ethereum, providing Turing-complete smart contracts that are essentially a collection of codes and data residing at specific account addresses on the Ethereum blockchain [4].

Different from Bitcoin using the *unspent transaction output* (UTXO) model, Ethereum and other Ethereum-like blockchains are designed based on the *Account/Balance* model to support smart contracts. Due to this difference, Ethereum has the concept of accounts and balances, which do not exist in Bitcoin.

There are two types of accounts in Ethereum: *Externally Owned Accounts* (EOAs), which are also known as Normal Accounts in some Ethereum-like blockchains, and *Contract Accounts* (CAs). Both EOAs and CAs have unique addresses and associated cryptocurrency balances. Their major difference is that EOAs are controlled by people who hold the public-private key pairs, whereas CAs are ruled by executable codes of smart contracts. For each CA, code execution is triggered by transactions from EOAs or messages (calls) received from other CAs.

Accounts are essential for users to interact with the Ethereum blockchain via transactions. The term transaction is used in Ethereum to refer to the data to be sent from an account to another on the blockchain. A transaction is called the *external one* if it is sent by an EOA, or the *internal one* if it is generated by executing a smart contract. Transactions can be categorized into different types, including `CALL`, `CREATE`, `REWARD` and `SUICIDE`. The most common type is the `CALL` transactions used for money transferring or contract invoking. The `CREATE` transactions are used to deploy smart contracts, and they can be destroyed at the end of their cycles via the `SUICIDE` method. The `REWARD` transactions appear on the head of each block, which depicts the reward that block miners obtained from special system accounts. Based on the these transactions, various activities can be enabled on Ethereum, such as money transfer, contract creation, and contract invocation [11].

2.2 Identities in Blockchain

Smart contracts can enable various blockchain applications, beyond simple cryptocurrency trading. A typical example is ERC-20, which is a technical standard on the Ethereum for implementing tokens [15]. It defines a common list of rules that an Ethereum token has to implement, giving developers the ability to program new tokens. With ERC-20, many new identities emerge, such as primary market investors and ICO fundraisers [8]. Besides, there are many different kinds of smart contracts. For example, there are contracts for non-fungible tokens (also known as deeds) based on ERC-721 standard and contracts for Ethereum Name Service based on ERC-137 standard [16]. Thanks to their Turing-completeness, arbitrary smart contracts are expected, and correspondingly, users with arbitrary identities are involved in the blockchain.

It is worthwhile to mention that it is very hard, if possible, to make a perfect identity categorization. Generally, these accounts can be classified into different categories according to user roles on the blockchain. A well-recognized identity categorization [17] including prominent ones, including mining pools, exchanges, token wallets, and investors, is studied in this paper, whose details will be given later. However, Ethereum is still in the development stage. The implementation of DApps is being extended and deepened, so various identities generated by DApps emerge inevitably.

TABLE 1
Typical Account Identities

Identity	Type	Description
Miners/Mining Pools	EOA/CA	Node who takes part in the block validation process
Pool	EOA/CA	The pooling of resources by miners, who share their processing power over a network
Token Contract	CA	Smart contract that allows customers to transfer ERC-20 tokens
Investor	EOA	Large holder of ETH, who usually got in early on ICOs
ICO Wallet	EOA/CA	ETH holding of token team, typically raised from ICOs
Exchange Deposit	EOA/CA	Address for user to deposit ETH at exchange
Exchange Root	EOA/CA	Address collects ETH from deposit addresses and withdraw ETH to users
Phish/Hack	EOA/CA	Fraud address related to phishing and hack

TABLE 2
NOTATIONS

Symbol	Explaination
G	The transaction graph
V	Set of nodes in the graph
E	Set of edges in the graph
Y	Embedding of the graph
$\vec{y_i}$	Embedding of the node v_i
(v_i, v_j, w, h, r)	Directed edge from node v_i to v_j, with weight w, block height h and relation r
E_i^+	Set of edges that point to node v_i
E_i^-	Set of edges that initiated from node v_i
X	Feature matrix of all nodes
$\vec{x_i}$	Feature vector of the node v_i
A^r, a_{ij}^r	Adjacency matrix of relation r, and the element is defined as first-order proximity between node v_i and v_j
T^r, t_{ij}^r	Transaction density matrix of relation r, and the element is defined as time variance of edges between node v_i and v_j

3 I^{2}GL Design

In this section, we propose the design of I^{2}GL. We first give an overview of I^{2}GL, and then present technical details including graph construction, graph learning and node classification.

3.1 Overview

The basic idea of I^{2}GL is to present blockchain accounts and associated transactions as a graph, in which accounts are represented as nodes with low dimensional feature vectors by graph learning techniques. Then, nodes identities are inferred according to a labeled training set. The account categorization is important for identity inference. In blockchains that support smart contracts, such as Ethereum, the perfect categorization does not exist because they are still in the developing stage. We use a taxonomy illustrated in TABLE 3, which is also advocated by [17]. Specifically, I^{2}GL consists of three phases: graph construction, graph learning, and node classification, as shown in Fig. 1. The main notations used in this section are summarized in TABLE 2.

Phase 1: Graph construction. We construct the blockchain transaction graph as a directed graph $G = (V, E)$, where each node $v \in V$ represents an account that can be an EOA or CA. We use the terms of account and node interchangeably in the rest of this paper. The total number of accounts is $N = |V|$.

The set E contains edges, each of which can be represented as a quintuple, i.e., $E = \{(v_i, v_j, w, h, r)|v_i, v_j \in V, w \in \mathbb{R}^+ \cup \{0\}, h \in \mathbb{Z}, r \in R\}$, where (v_i, v_j) indicates the direction of the transaction (e.g., assets transfer or smart contract invocation) from v_i to v_j, weight w represents the amount of transferred cryptocurrencies, and h is the block height of the transaction. Note that block height is defined as the number of blocks between the block containing the current transaction and the very first block, which is also called genesis block. We usually use block height, instead of the physical clock time, as the timestamp in blockchain. The final parameter r is the edge type that indicates different kinds of transactions, e.g., transferring money, creating or invoking smart contracts.

Phase 2: Graph learning. Since the blockchain transaction graph is too large to be handled by traditional graph analytics methods, we use graph learning, which is also called graph embedding, techniques [18] to convert the original graph into a low-dimensional one. The learning process can be defined as follows: given $G = (V, E)$, we represent each node v_i in a low-dimensional vector space $\vec{y_i}$. Based on such vectors (represented as a matrix Y), node classification can then be computed efficiently with high accuracy.

Traditional graph embedding techniques, such as random walk, map the pure graph structures into the embedding space [19]. Our model is primarily motivated as an extension of GCN (Graph Convolutional Network) [20], [21], which applies the convolution operators to the non-Euclid graph structural space. It has been shown that GCN is effective in object classification in large-scale applications with relational data.

Phase 3: Node classification. The final objective of I^{2}GL is to infer node identities on the transaction graph. Given a labeled node set, supervised or semi-supervised methods can be used for entities classification. I^{2}GL creates a GCN with layers using the softmax $(\cdot)$ activation [21]. A loss function, such as cross-entropy error function, can be used as the training objective.

3.2 Graph Construction

A transaction graph can be constructed based on original transaction records synchronized from the blockchain. In I^{2}GL, the transaction graph is represented by three kinds of matrices: (a) *node representations*, (b) *adjacency matrices*, and (c) *time density matrices*.

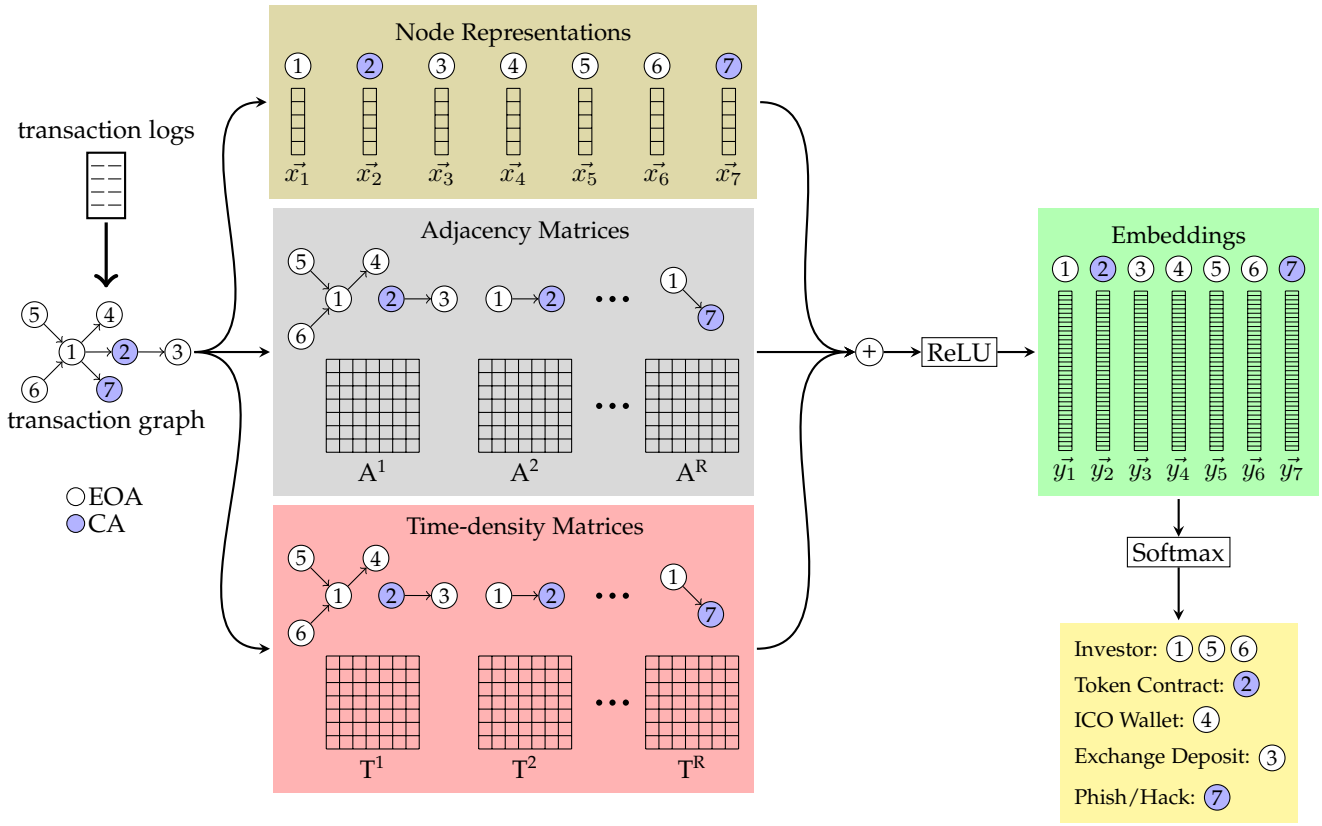


Fig. 1. Overall architecture.

3.2.1 Node representations

To encode the overall structural and additional information of accounts, we denote the node representations as a feature matrix $X \in \mathbb{R}^{N \times d}$, where each account is depicted as a vector $\vec{x_i}$ containing the following d features.

- *In-degree*: the number of incoming edges of a node, i.e., the number of transactions received by a node. The in-degree of node v_i can be counted as $|E_i^+|$, and $E_i^+ = \{(v_j, v_i, w, h, r) | \forall v_j \in V\}$.
- *Out-degree*: the number of outgoing edges of a node, i.e., the number of transactions issued by a node. The out-degree of node v_i can be counted as $|E_i^-|$, and $E_i^- = \{(v_i, v_j, w, h, r) | \forall v_j \in V\}$.
- *Weighted in-degree*: the sum of values of incoming edges of a node, which represents the total amount of cryptocurrencies received by a node. The weighted in-degree of node v_i is $\sum_{e \in E_i^+} w$.
- *Weighted out-degree*: the sum of values of edges outgoing to a node, which represents the total amount of cryptocurrencies sent from a node. The weighted out-degree of node v_i can be computed as $\sum_{e \in E_i^-} w$.
- *Node type*: whether the node is a smart contract or a normal user account.

3.2.2 Adjacency matrices

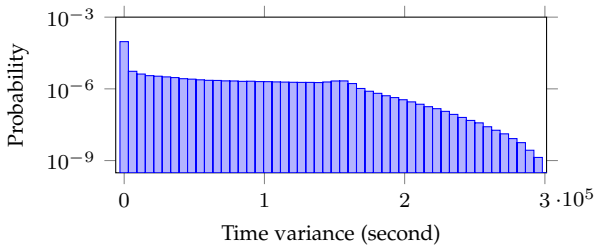
Different from social networks, edges in the transaction graph stand for various activities such as transferring money, creating smart contracts, and invoking smart contracts. Obviously, such activities should be represented as different edge types instead of being measured in uniformly weighted graph.

To address the challenge, we divide the original transaction graph into several sub-graphs and capture the relations of neighboring nodes using adjacent matrices. We use a set of adjacency matrices, $\{A^1, A^2, \ldots, A^R | A^r \in \mathbb{R}^{N \times N}\}$, to describe R edge types among N nodes in the transaction graph, respectively.

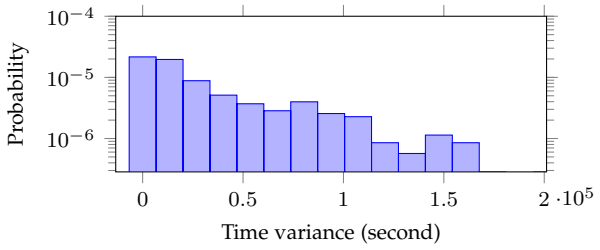
Specifically, four major types of transactions are considered for Ethereum, including (I) `CALL` transactions with weight (e.g., cryptocurrency transferring), (II) `CALL` transactions without weight (e.g., smart contract invocation), (III) `CREATION` transactions (e.g., smart contract deployment) and (IV) `REWARD` transactions (e.g., mining rewarding). For example, a transaction is regarded as type-I if it involves some ETH (cryptocurrency in Ethereum) transferring. And the transactions of ERC-20 token transferring are regarded as type-II since they are invocation of smart contracts without any ETH.

For types-II, -III and -IV, we calculate the frequency of repeated edges between a pair of nodes as the adjacent weight in the corresponding adjacent matrix. On the other hand, cryptocurrency transferring, i.e., type-I, is not sensible to use the assets amount directly as the adjacent weight. That is because these accounts transfer enormously varied amounts of assets, which would lead to underflow or overflow in the training process.

Therefore, the amounts of assets transferring are discretized into three types: (1) small transfers whose transaction payloads are less than 1 ETH; (2) medium transfers



(a) Time variance histogram of all nodes.



(b) Time variance histogram of Phish/Hack nodes.

Fig. 2. Time variance histogram of accounts in Ethereum. Phish/Hack nodes have smaller variances, no greater than 2 seconds, than other nodes, which indicates that they are more active in a short period.

with transaction payloads between 1 ETH and 10 ETH; and (3) large transfers whose transaction payloads are more than 10 ETH. Note that such a discretization is heuristic and it can be adjusted during performance tuning.

3.2.3 Transaction density matrices

In addition to adjacent matrices, I^{2}GL exploits more information on transaction graphs to further increase the accuracy of identity inference. To have a better understanding of transaction density, we study the distribution of transaction time and show the variances in Fig. 2. Note that transactions can be associated with Unix timestamps, which are attached to blocks when they are created. The transaction timestamps can be obtained using API `eth.getBlock(blockNumber).timestamp`. We observe that Phish/Hack nodes have smaller variances, no greater than 2 seconds, than other nodes, which indicates that they are more active in a short period. This observation motivates us to consider the density of transaction time in identity inference.

Specifically, we use a set of matrices $\{T^1, T^2, \ldots, T^R | T^r \in \mathbb{R}^{N\times N}\}$ to describe transaction density. Given a sequence of $\{h^r_{ij1}, h^r_{ij2}, \ldots, h^r_{ijm} | h^r_{ijk} > 0\}$ as the block heights of m transactions of type r between node v_i and v_j, the transaction density $t^r_{ij} \in T^r$ can be computed as

$$t^r_{ij} = g\left(\sqrt{Var\left[\frac{1}{m}\sum_{k=1}^{m} h^r_{ijk}\right]}\right) \quad (1)$$

where $g(\cdot)$ is the function of squash, which can be logarithmic function.

3.3 Graph Learning

3.3.1 GCN propagation rule

Based on the above matrices, we propose a multi-layer Graph Convolutional Network with the following layer-wise propagation rule:

$$H^{(l+1)} = \delta\Big(\sum_{r\in R}(T^r \odot (D^r)^{-1}A^r)H^{(l)}W^{(l)}_r\Big), \forall r \quad (2)$$

where $H^{(l)}$ is the matrix of activations of the l-th layer, and here the 0-th layer is set as the node representations, that is, $H^{(0)} = X$. The W^l_r is a weight matrix for the l-th neural network layer belonging to the transaction type r. We use $\delta(\cdot)$ to denote an activation function such as the ReLU$(\cdot)$ = max$(0, \cdot)$. Note that A^r is the adjacent matrix of type r, and D^r is a diagonal matrix of type r, i.e., $d^r_{ii} = \sum_j a^r_{ij}$, where $d^r_{ii} \in D^r$ and $a^r_{ij} \in A^r$. The symbol $\odot$ means the point-wise multiplication.

Specifically, the input of the l-th layer of the GCN model is $H^{(l)} = \{h^{(l)}_1, h^{(l)}_2, ..., h^{(l)}_N | h^{(l)}_i \in \mathbb{R}^{N\times d^{(l)}}\}$. Hence, the layer-wise propagation can be treated as a special case of the forward updating process as

$$h^{(l+1)}_i = \delta\Big(\sum_{r\in R}\sum_{j\in N^r_i}\frac{t^r_{ij}}{\hat{c}_{i,r}}W^{(l)}_r h^{(l)}_j\Big) \quad (3)$$

where $h^{(l)}_i$ is the hidden state of node v_i in the l-th layer of the neural network, N^r_i denotes the set of neighbors of node v_i associated with transaction type r, t^r_{ij} is the transaction density of type r from node v_i to v_j, as shown in Eq. (1), and $\hat{c}_{i,r}$ is the coefficient representing asymmetric proximity, which will be explained later.

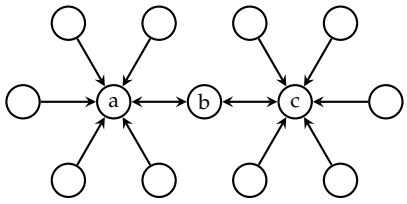
3.3.2 Second-order proximity

In existing work, the edge weight a^r_{ij} in adjacency matrix A^r is usually defined as the first-order proximity [22] that represents similarity between nodes v_i and v_j. However, I^{2}GL exploits the second-order proximity that evaluates the similarity of two nodes by comparing their neighborhood [23]. In the cryptocurrency transaction graph, we find that second-order proximity also plays an important role in preserving the local structure. As shown in Figure 3(a), v_a is not adjacent to v_c, but they have a similar neighbor structure. First-order proximity may miss such a similarity of nodes that are separated away, but second-order proximity can capture it to increase inference accuracy.

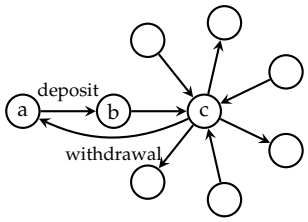
3.3.3 Asymmetric proximity

The asymmetric proximity, which has been ignored by many existing work, is also considered by I^{2}GL. For example, as shown in Fig. 3(b), suppose node v_a is a cryptocurrency trader, and nodes v_b and v_c are cryptocurrency exchange accounts. The exchange may move user deposits among several internal accounts, e.g., v_b and v_c in our example. We suppose that edge (v_a, v_b) stands for a deposit process and edge (v_c, v_a) is a withdrawal process. Generally, these edges have equal weights, i.e., $w^r_{ab} = w^r_{bc} = w^r_{ca}$ since deposit and withdrawal come in pairs in the symmetric model. However, we need to distinguish the edge (v_a, v_c) from (v_c, v_a) because they represent different activities.

Different from the asymmetric proximity preserving approach based on random walk [24], I^{2}GL uses a non-probability graph learning model. To preserve the asymmetric proximity, the coefficient $\hat{c}_{i,r}$ is introduced as:



(a) Second-order proximity



(b) Asymmetric proximity

Fig. 3. Examples of second-order and asymmetric proximity. (a) Nodes a and c have similar neighboring structure, which can be captured by second-order proximity. (b) Suppose node v_a is a cryptocurrency trader, and nodes v_b and v_c are cryptocurrency exchange accounts. We need to distinguish the relationship between (v_a, v_c) and (v_c, v_a) by asymmetric proximity because they represent different activities.

$$\hat{c}_{i,r} = \frac{1}{g(d_i^r) \cdot |N_i^r|} \tag{4}$$

where $d_i^r = \sum_j a_{ij}^r$, g is a squash function, and $|N_i^r|$ is used for normalization.

For a better understanding, we use an example in Fig. 4 to show how I^{2}GL exploits both second-order proximity and asymmetric proximity. In this example, we construct a 3-layer GCN for a graph consisting of 7 nodes. We show the feature aggregation processes of nodes 1 and 2. Specifically, in the layer 1, node 1 aggregates feature data from neighbors 2, 5, 6, who have direct links to node 1. Node 2 gets feature data from node 3. In the next-layer aggregation, node 2 continues to share its aggregated feature data to node 1, so that node 2 can obtain information from node 3 that is two-hop away.

3.4 Node Classification

We use the cross-entropy loss function as the training objective:

$$L = -\sum_{i=1}^{T}\sum_{j=1}^{m} y_{i,j} \log p_{i,j} + \lambda ||\theta||^2 \tag{5}$$

where T is the number of training samples, $y_{i,j}$ is the true probability of node v_i belonging to category j, θ is the set of all parameters and λ is the coefficient for L_2 regularization. The output of the last GCN layer is a probability matrix $P = \{\vec{p_1}, \vec{p_2}, ..., \vec{p_N} | \vec{p_i} \in \mathbb{R}^{1\times m}\}$, where the vector $\vec{p_i}$ includes the probabilities of classifying node v_i into m categories, respectively.

4 Experiments

In this section, we conduct experiments to evaluate the performance of I^{2}GL. Note that I^{2}GL can be applied for not only Ethereum, but also other blockchains that support smart contracts.

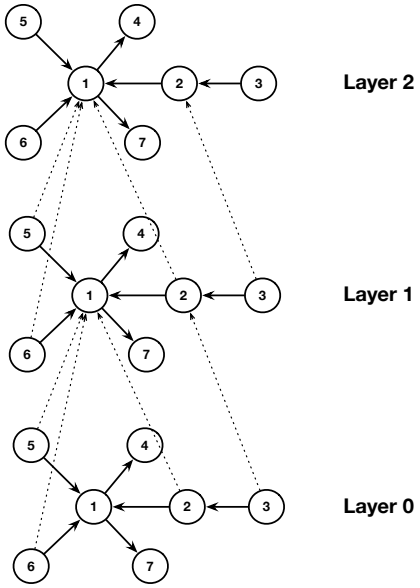


Fig. 4. A toy example of exploiting second-order proximity and asymmetric proximity.

TABLE 3
The number of labeled accounts

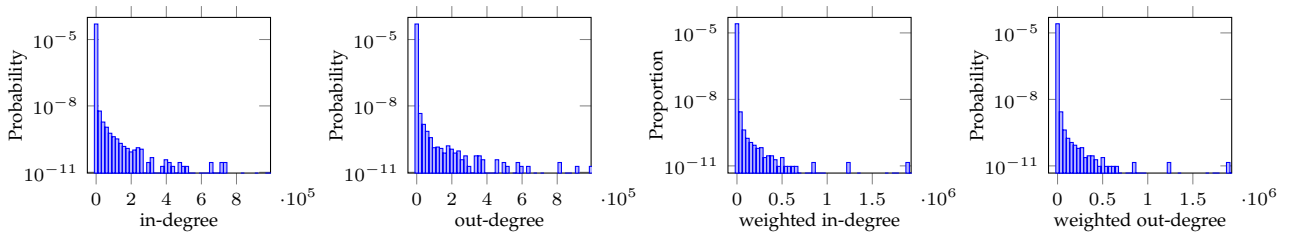
Miners/Mining Pools	100	Pool	99
Token Contract	82	Investor	100
ICO Wallet	100	Exchange Deposit	100
Exchange Root	99	Phish/Hack	100

4.1 Data Collection

We collect data by running an Ethereum client[1], which synchronizes all historical transaction records from the Ethereum blockchain. We choose the transaction records during January 1, 2018, and March 31, 2018, including 116,293,867 external transactions and internal transactions, as the input of graph construction. This is the most active period with various activities. By parsing these transactions, we obtain 16,599,825 active accounts, including 14,450,993 EOAs, 2,148,831 CAs, and a system account in charge of paying ETH bonuses to miners. Fig. 5 shows degree distributions of Ethereum transaction graph, which follow the power law. The results indicate that a few accounts make most of transactions with a large amount of ETH.

Since I^{2}GL uses a semi-supervised learning method, a small set of labeled accounts should be prepared for training. We obtain these labeled examples from *Etherscan*[2] and *Searchain*[3]. The address types can be acquired using the web3 API `web3.eth.getCode(address)`. If the return value is `'0x'`, the address is an externally owned account. If the return value is a code in hexadecimal representation, the address is a smart contract. The numbers of labeled accounts are shown in Table. **??**.

1. Parity Ethereum Client, https://www.parity.io/ethereum/
2. Etherscan LabelCloud, https://etherscan.io/labelcloud
3. Searchain, http://www.searchain.io/



(a) in-degree distribution (b) out-degree distribution (c) weighted in-degree distribution (d) weighted out-degree distribution

Fig. 5. Degree distributions in Ethereum transaction graph.

4.2 Identity Categorization

TABLE 3 shows the taxonomy of Ethereum accounts used in our experiments. Even though new account types emerge as the development of Ethereum, I^{2}GL can be easily extended to handle them. The details of some major account types are depicted as follows.

1) *Miners/Mining Pools.* The existing implementation of Ethereum blockchain uses the *Proof-of-Work (PoW)* protocol, similar to Bitcoin. The *miners* are the individuals or groups who validate transaction information and package transactions by solving cryptographic puzzles. The node who is the first to find a valid hash gets the reward in the form of ETH that is paid by users sending transactions. To improve mining efficiency, miners work together to solve the PoW problems by registering at a special institution named *mining pool*, which aggregates all the registrants' computing power to solve mining problems and distributes the reward to registrants according to their proportion of contributed computing power. Up to March 2018, top 3 mining pools occupy more than 65% of the hash rates in Ethereum[4].

2) *Exchanges.* The exchanges are the platforms for trading ETH and other cryptocurrencies, and they play an important role in the Ethereum ecosystem. Most of the cryptocurrency trading is done through centralized exchanges such as Binance, Huobi, OKEX, etc[5]. The centralized exchanges allocate deposit addresses to users who want to make transactions via the exchanges. These addresses are called *exchange deposits* and belong to the exchanges since users do not have the private keys of these addresses. In the recharging process, users transfer coins to the given deposit addresses from their own wallets and these coins will be transferred to the *exchange root* address automatically. In turn, users send requests to the exchange to withdraw their coins from an address called *exchange withdrawal*. The exchange root and exchange withdrawal are usually the same address.

3) *ERC-20 & ICO.* As a technical standard on Ethereum, ERC-20 defines a common list of rules that an Ethereum token has to implement, giving developers the ability to program new tokens within the Ethereum ecosystem. Such ERC-20 token transfer happens in a specific contract called *ERC-20 token contract*. The ERC-20 token standard becomes popular with crowdfunding companies working on ICO cases due to the simplicity of deployment, together with its potential for interoperability with other Ethereum token standards [25]. Up to July 26, 2018, there were more than 103,621 ERC-20 token contracts[6]. Some successful ERC-20 token sales are EOS, Filecoin, Bancor, Qash, and Nebulas, raising over 60 million each[7]. Participants in the initial ICO round are *primary market investors* who buy the ERC-20 tokens from ERC-20 smart contracts of the crowdfunding companies. These addresses used for ETH token holding are called *ICO wallets*.

4) *Phish/Hack.* While virtual property transactions are now becoming increasingly popular, they face many security issues. The number of frauds associated with ETH and ERC-20 tokens has kept increasing in recent years. We call these addresses related to frauds as *Phishes/Hacks*. According to Ethersacan, there are more than 2500 addresses are labeled as Phishes/Hacks, which takes up the highest proportion, and most of them are disguised as ERC-20 token sales or DApps such as casino.

4.3 Experiment Setting

We compare I^{2}GL against the state-of-the-art DeepWalk [26], PARW [27] and rGCN [21]. DeepWalk is an embedding technique using random walks on graphs to obtain node representations. Similar to I^{2}GL, rGCN tackles node representation by defining a convolution operator on graph. PARW is a label propagation method based on partially absorbing random walks. Note that DeepWalk is an unsupervised method, a logistic regression model is added for classification. The GCN model has two hidden layers and each layer has 16 units. Models are trained with Adam optimizer for 100 epochs, and the dropout rate is set to 0.5 to avoid overfitting. All graph learning and classification programs run on the server with an Intel Xeon E5 CPU of 55 processors and 128GB memory. The GPU used for graph learning is Nvidia 1080. Experiment settings clearly explain how long it takes for the server to generate the model. We choose the parameters of other models by convention. In DeepWalk, the number of random walks to start at each node is 10, the walk length is 80, and the representation size is set to 128. In PARW, the absorbing parameter λ is set to 0.001.

4.4 Experiment Results

For clarity, we have two assumptions about identity inference: 1) each account is supposed to have a single identity

4. Investoon, https://investoon.com/charts/mining/eth.

5. "Top 100 Cryptocurrency Exchanges by Trade Volume", https://coinmarketcap.com/rankings/exchanges

6. Etherscan Token Tracker Page, https://etherscan.io/tokens

7. "Token Data, data and analytics for all ICO's and tokens", https://www.tokendata.io

TABLE 4
Identity Classification Results

	DeepWalk			PARW			rGCN			I^{2}GL		
	Precision	**Recall**	**F_1**	**Precision**	**Recall**	**F_1**	**Precision**	**Recall**	**F_1**	**Precision**	**Recall**	**F_1**
Phish/Hack	0.609	0.394	0.479	0.565	0.333	0.419	0.913	0.212	0.344	0.773	0.758	0.765
Token Contract	0.857	0.735	0.791	0.354	0.718	0.475	0.908	0.602	0.724	0.960	0.970	0.965
Exchange Deposit	0.586	0.531	0.557	0.692	0.281	0.400	0.688	0.440	0.537	0.567	0.680	0.618
Exchange Root	0.647	0.759	0.698	0.667	0.759	0.710	0.923	0.686	0.787	0.964	0.771	0.857
Pool	0.692	0.750	0.720	0.789	0.625	0.697	1.000	0.727	0.842	0.800	0.727	0.762
Miner	0.400	0.694	0.508	0.667	0.872	0.756	0.867	0.951	0.907	0.857	0.976	0.909
Investor	0.405	0.548	0.466	0.727	0.516	0.604	0.739	0.548	0.630	0.727	0.516	0.604
ICO Wallet	0.364	0.353	0.358	0.630	0.500	0.558	0.546	0.158	0.245	0.759	0.579	0.657
average	0.614	0.583	**0.585**	0.623	0.577	**0.570**	0.848	0.496	**0.593**	0.829	0.792	**0.806**

instead of multiple identities, and (2) identity of a certain account does not change. These assumptions are reasonable because even though a participant have multiple identities (e.g., it can be a miner and an investor simultaneously) in practice, it uses different accounts for these activities, respectively. Because we use transaction records during a period of merely three months, cases of identity conversion are negligible.

We use three metrics, precision, recall and F_1 score, to evaluate the performance of different approaches. Precision is the fraction of relevant instances among the retrieved instances, while recall is the fraction of relevant instances that have been retrieved over the total amount of relevant instances. High precision means the classifier is unlikely to consider accounts with other identities as this identity, while high recall means that most of the accounts in this identity will be recognized. The F_1 score is a measurement that combines precision and recall, which is computed as

$$\begin{aligned} F_1 &= \left(\frac{\text{precision}^{-1} + \text{recall}^{-1}}{2} \right)^{-1} \\ &= 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}. \end{aligned} \tag{6}$$

In statistical analysis of label classification, the F_1 score is an important indicator since it is the harmonic average of the precision and recall. An ideal classifier has both high precision and recall, thus a high F_1 score.

Results are summarized in TABLE 4. Our proposed I^{2}GL has the highest F_1 score of 0.806, outperforming other approaches by at least 35.9%. Especially, it performs well in recognizing Token Contracts and Miners, with over 0.9 $F1$ score. That is because I^{2}GL can exploit global structural information and statistic information, but other approaches use only the local structure of nodes.

Note that rGCN achieves the highest average precision but a low average recall. Especially, rGCN has the lowest recall of category of Phishes/Hacks, which means that many normal accounts are inferred as threatening in rGCN model. I^{2}GL improves the overall performance by adding richer information, i.e., the time of transactions, into the model.

4.5 Visualization

I^{2}GL is a graph deep learning approach, and each node in the graph can be represented as a low-dimensional vector. It allows us to visualize the nodes and gain a better understanding of classification. We illustrate the visualization of DeepWalk, rGCN and I^{2}GL in Fig. 6. Similar to [28], we use the 128-dimensional embedding for each method, and apply t-SNE [29] to reduce the dimension to 2, so that all nodes can be visualized in a 2-dimensional space.

We use different colors to denote accounts labeled as different identities. Therefore, in a good embedding result, points with the same color should be grouped together. It can be observed that all GCN-based approaches outperform DeepWalk since GCN model well preserves global structure information [23]. Furthermore, we find that many Phish/Hack accounts are usually disguised as ICO wallets, exchange deposits, and ERC-20 token contracts because their points overlap in Fig 6(c).

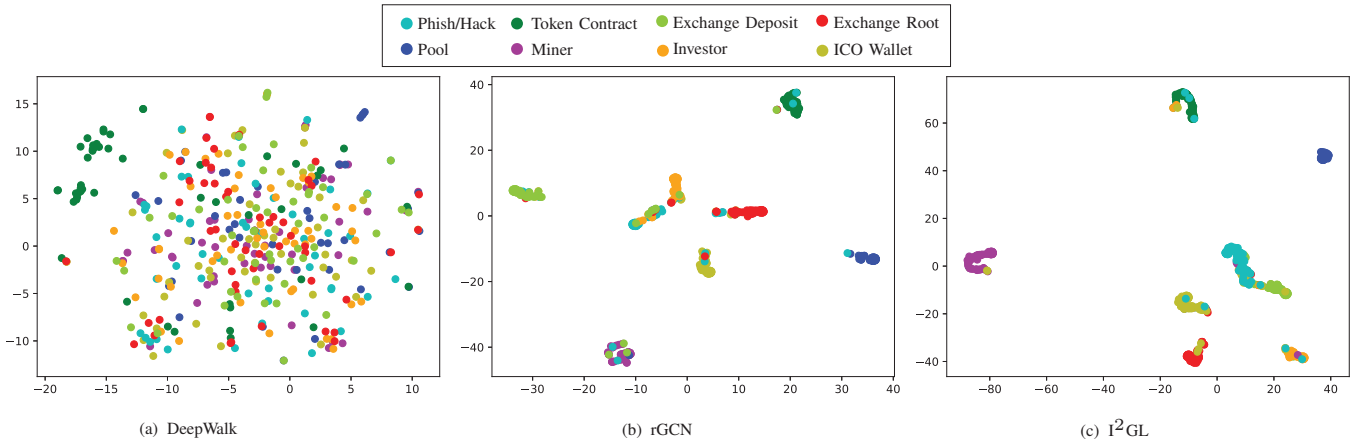
4.6 Comparison to Different Design Choices

We propose a series of techniques in I^{2}GL to exploit different features of blockchain transaction graphs. To clearly show the contribution of each technique, we compare the performance of I^{2}GL variants in Fig. 7.

First, we compare the performance of I^{2}GL with single adjacency matrix and single type (I^{2}GL$_{sR}$ for short) and I^{2}GL with multi-adjacency matrices (I^{2}GL$_{mR}$ for short). Note that both I^{2}GL$_{sR}$ and I^{2}GL$_{mR}$ do not use transaction density matrices. As shown in Fig. 7(a), the classification accuracy of I^{2}GL$_{mR}$ increases greatly by introducing multi-adjacency matrices, which indicates that such heterogeneous activities are important in preserving features of transaction graph.

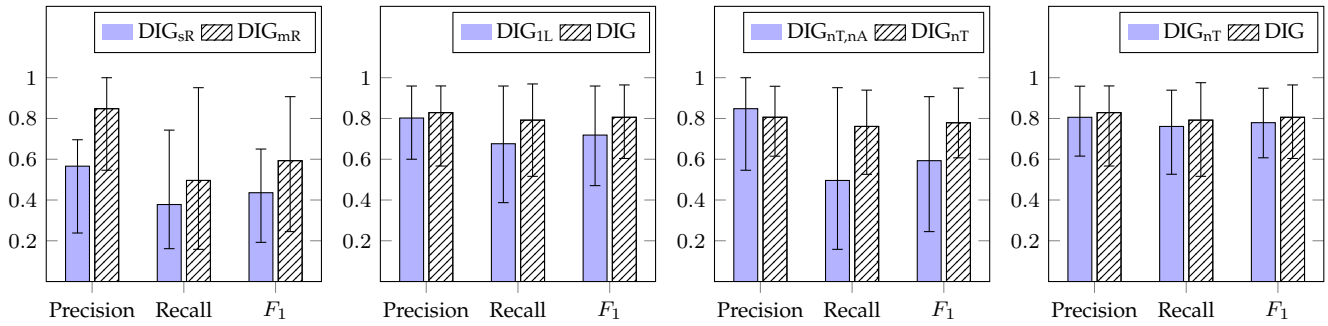
Second, we evaluate the I^{2}GL with 1-layer convolutional network (I^{2}GL$_{1L}$ for short) and the complete I^{2}GL with 2-layer convolutional network. As shown in Fig. 7(b), the F_1 increases 13% by preserving second-order proximity, which demonstrates the importance of second-order proximity in the structure of transaction graph.

Then, we investigate the influence of asymmetric proximity in Ethereum transaction graph, by comparing the I^{2}GL without asymmetric coefficient and transaction density matrices (I^{2}GL$_{nT,nA}$ for short) with one without transaction density matrices (I^{2}GL$_{nT}$ for short) only. As shown



(a) DeepWalk (b) rGCN (c) I^{2}GL

Fig. 6. Visualization of nodes with different labels in 2-dimensional space.



(a) Multi-Adjacencies (b) Second-order proximity (c) Asymmetric proximity (d) Transaction density

Fig. 7. Classification results of diminished I^{2}GLs.

in Fig. 7(c), I^{2}GL$_{nT,nA}$ has higher precision but I^{2}GL$_{nT}$ outperforms in metrics of recall and F_1 by 56% and 31%, respectively.

Finally, we evaluate the performance of I^{2}GL without transaction density matrices (I^{2}GL$_{nT}$ for short). As shown in Fig. 7(d), I^{2}GL outperforms I^{2}GL$_{nT}$ in all metrics. The reason is that different types of accounts have diverse active time distribution, and transaction density helps to distinguish them.

5 Related Works

5.1 Blockchain and Ethereum

Satoshi Nakamoto published Bitcoin whitepaper [30] in October of 2008. As the earliest implementation of blockchain, Bitcoin is the most striking example of the concept of a decentralized cryptocurrency system. The production of Bitcoin depends on massive computations solving cryptographic puzzles without any organization, which guarantees the consistency and security in the distributed ledger system.

With the development of blockchain technology, more successors have merged and tried to extend the functions related to different applications. The most significant one is Ethereum [4], providing Turing-complete smart contracts, which opens new possibilities of applications. People can develop distributed applications (DApps) with complex functions based on the Ethereum smart contract. These DApps provide the solutions for various fields other than basic transactions, such as decentralized exchange (DEX), initial coin offering (ICO), lending and so on.

However, as one of the most salient features, the anonymity of blockchain leads to that it is hard to find the real identities behind addresses as well as it protects the users' privacy. It makes trading analysis difficult and offers a breeding ground for frauds. According to Cointelegraph, the Ethereum network has experienced considerable phishing, Ponzi schemes, and other scams events, accounting for about 10% of ICOs [8].

5.2 Analysis of Blockchain

Many studies focus on characterizing blockchain system such as Bitcoin through graph analysis. It is hard to figure out the transaction trace in Bitcoin since it uses unique UTXO model instead of traditional account model. Earlier studies abstract Bitcoin into a transaction graph [1], where each node represents a transaction and a directed edge from node A to node B means that the output of A is the input of B. Meiklejohn et al. propose address clustering to associate users with their addresses [31], so that Bitcoin can be modeled as an account based graph. Based on the graph analysis, many applications have been developed, such as deanonymization [1], and money laundering detection [2], [3], [10].

On the other hand, the account/balance model makes the record-keeping for Ethereum just like that in a bank. Based on that, smart contracts can keep track of states to perform different tasks. Consequently, the transactions happen on Ethereum are even more complex than Bitcoin with various forms. So far as we know, few studies have investigated the transactions on Ethereum. Chen et al. [11] conduct a systematic study on Ethereum by leveraging graph analysis, however, they mainly proposed insights on the basis of statistics without any specific task. Bartoletti et al. [32] have presented a comprehensive survey of financial frauds on Ethereum, analyzing their behaviours and their impact from various viewpoints. Tan et al. [33] have applied graph convolutional graph for detecting abnormal financial transactions in e-payment networks. It is different from our work because we focus on node identity inference instead of abnormal transactions. Wu et al. [34] have studied the detection of addresses belonging to mixing services, and have proposed a feature-based network analysis framework, which cannot be applied in solving the identity inference problem studied in this paper.

5.3 Graph Learning

Graph analysis has been attracting increasing attention in the recent years which enables researchers to understand the various network system in a systematic manner. In the survey of graph learning [18], graph analytic tasks can be broadly abstracted into a lot of categories, such as node classification [35], link prediction [36], clustering [37] and visualization [29]. For example, node classification aims at determining the label of nodes based on other labeled nodes and the topology of the network.

Graph learning provides an effective way to solve the graph analytics problem which converts the graph into a low dimensional space in which the graph information is preserved [14]. In the past decade, there has been a lot of research in the field of graph learning, and the most significant methods are factorization based methods [38]–[40], random walk based methods [26], [41] and deep learning based methods [20], [28].

Embedding graphs into low dimensional spaces is not a trivial task and the challenges of graph learning depend on the problem setting. The input of graph embedding is a graph which constructed from raw data. In [23], the most studied graph embedding input is a heterogeneous graph in which both nodes and edges have multiple types respectively. Typical heterogeneous graphs mainly exist in the scenarios such as community-based question answering, multimedia networks, and knowledge graphs. To the best of our knowledge, this paper is the first work to analyze the transaction graph of blockchain based on graph embedding techniques.

6 Conclusion

In this paper, we have studied the identity inference in Ethereum and similar DApp platform blockchains. An approach called I^{2}GL has been proposed to infer user identities using graph deep learning. Since the blockchain transaction graph is too large to be handled by traditional graph analytics methods, we use graph learning technique to convert the original graph into a low-dimensional one. Furthermore, we exploit unique features of blockchain transaction graphs, such as transaction density, second-order proximity, and asymmetric proximity, and correspondingly propose a series of enhancement. We evaluate I^{2}GL by comparing it with three state-of-the-art and experimental results on Ethereum transaction records show the superiority of I^{2}GL.

References

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

[2] D. D. F. Maesa, A. Marino, and L. Ricci, "An analysis of the bitcoin users graph: inferring unusual behaviours," in *International Workshop on Complex Networks and their Applications*. Springer, 2016, pp. 749–760.

[3] S. Ranshous, C. A. Joslyn, S. Kreyling, K. Nowak, N. F. Samatova, C. L. West, and S. Winters, "Exchange pattern mining in the bitcoin transaction directed hypergraph," in *International Conference on Financial Cryptography and Data Security*. Springer, 2017, pp. 248–263.

[4] V. Buterin *et al.*, "Ethereum white paper," 2013.

[5] "Cryptokitties," https://www.cryptokitties.co.

[6] "Idex," https://idex.market/eth/aura.

[7] "Etherscan," https://etherscan.io/accounts?l=Phish/Hack.

[8] P. Cerchiello, A. M. Toma *et al.*, "Icos success drivers: a textual and statistical analysis," University of Pavia, Department of Economics and Management, Tech. Rep., 2018.

[9] B. Jiang, Y. Liu, and W. Chan, "Contractfuzzer: Fuzzing smart contracts for vulnerability detection," in *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*. ACM, 2018, pp. 259–269.

[10] C. Zhao and Y. Guan, "A graph-based investigation of bitcoin transactions," in *IFIP International Conference on Digital Forensics*. Springer, 2015, pp. 79–95.

[11] T. Chen, Y. Zhu, Z. Li, J. Chen, X. Li, X. Luo, X. Lin, and X. Zhange, "Understanding ethereum via graph analysis," in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*. IEEE, 2018, pp. 1484–1492.

[12] X. Geng, H. Zhang, J. Bian, and T.-S. Chua, "Learning image and user features for recommendation in social networks," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 4274–4282.

[13] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, and J. Taylor, "Freebase: a collaboratively created graph database for structuring human knowledge," in *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*. AcM, 2008, pp. 1247–1250.

[14] W. L. Hamilton, R. Ying, and J. Leskovec, "Representation learning on graphs: Methods and applications," *arXiv preprint arXiv:1709.05584*, 2017.

[15] Theethereum, "ERC-20 Token Standard - The Ethereum Wiki," https://theethereum.wiki/w/index.php/ERC20_Token_Standard, 2017, [Online; accessed 30-August-2017].

[16] "Erc draft," https://eips.ethereum.org/erc.

[17] TokenAnalyst, https://www.tokenanalyst.io/.

[18] H. Cai, V. W. Zheng, and K. Chang, "A comprehensive survey of graph embedding: problems, techniques and applications," *IEEE Transactions on Knowledge and Data Engineering*, 2018.

[19] P. Goyal, H. Hosseinmardi, E. Ferrara, and A. Galstyan, "Capturing edge attributes via network embedding," *arXiv preprint arXiv:1805.03280*, 2018.

[20] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[21] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. van den Berg, I. Titov, and M. Welling, "Modeling relational data with graph convolutional networks," in *European Semantic Web Conference*. Springer, 2018, pp. 593–607.

[22] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "Line: Large-scale information network embedding," in *Proceedings of the 24th International Conference on World Wide Web*. International World Wide Web Conferences Steering Committee, 2015, pp. 1067–1077.

[23] P. Goyal and E. Ferrara, "Graph embedding techniques, applications, and performance: A survey," *Knowledge-Based Systems*, vol. 151, pp. 78–94, 2018.
[24] C. Zhou, Y. Liu, X. Liu, Z. Liu, and J. Gao, "Scalable graph embedding for asymmetric proximity." in *AAAI*, 2017, pp. 2942–2948.
[25] BitcoinForBeginners, " What is an ERC20 Token," https://www.bitcoinforbeginners.io/cryptocurrency-guide/what-is-an-erc20-token/, 2018, [Online; accessed 14-June-2018].
[26] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2014, pp. 701–710.
[27] X.-M. Wu, Z. Li, A. M. So, J. Wright, and S.-F. Chang, "Learning with partially absorbing random walks," in *Advances in Neural Information Processing Systems*, 2012, pp. 3077–3085.
[28] D. Wang, P. Cui, and W. Zhu, "Structural deep network embedding," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2016, pp. 1225–1234.
[29] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of machine learning research*, vol. 9, no. Nov, pp. 2579–2605, 2008.
[30] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008.
[31] S. Meiklejohn, M. Pomarole, G. Jordan, K. Levchenko, D. McCoy, G. M. Voelker, and S. Savage, "A fistful of bitcoins: characterizing payments among men with no names," in *Proceedings of the 2013 conference on Internet measurement conference*. ACM, 2013, pp. 127–140.
[32] M. Bartoletti, S. Carta, T. Cimoli, and R. Saia, "Dissecting ponzi schemes on ethereum: Identification, analysis, and impact," *Future Generation Computer Systems*, vol. 102, pp. 259 – 277, 2020.
[33] D. S. H. Tam, W. C. Lau, B. Hu, Q. Ying, D. M. Chiu, and H. Liu, "Identifying illicit accounts in large scale e-payment networks - a graph representation learning approach," *ArXiv*, vol. abs/1906.05546, 2019.
[34] J. Wu, J. Liu, W. Chen, H. Huang, Z. Zheng, and Y. Zhang, "Detecting mixing services via mining bitcoin transaction network with hybrid motifs," *ArXiv*, vol. abs/2001.05233, 2020.
[35] S. Bhagat, G. Cormode, and S. Muthukrishnan, "Node classification in social networks," in *Social network data analytics*. Springer, 2011, pp. 115–148.
[36] D. Liben-Nowell and J. Kleinberg, "The link-prediction problem for social networks," *Journal of the American society for information science and technology*, vol. 58, no. 7, pp. 1019–1031, 2007.
[37] C. H. Ding, X. He, H. Zha, M. Gu, and H. D. Simon, "A min-max cut algorithm for graph partitioning and data clustering," in *Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on*. IEEE, 2001, pp. 107–114.
[38] A. Ahmed, N. Shervashidze, S. Narayanamurthy, V. Josifovski, and A. J. Smola, "Distributed large-scale natural graph factorization," in *Proceedings of the 22nd international conference on World Wide Web*. ACM, 2013, pp. 37–48.
[39] M. Belkin and P. Niyogi, "Laplacian eigenmaps and spectral techniques for embedding and clustering," in *Advances in neural information processing systems*, 2002, pp. 585–591.
[40] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *science*, vol. 290, no. 5500, pp. 2323–2326, 2000.
[41] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2016, pp. 855–864.