TGC: Transaction Graph Contrast Network for Ethereum Phishing Scam Detection

Sijia Li[1,2], Gaopeng Gou[1,2], Chang Liu[1,2,*], Gang Xiong[1,2], Zhen Li[1,2], Junchao Xiao[1,2], Xinyu Xing[3]
[1] Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
[2] School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China
[3] Northwestern University, Evanston, Illinois, USA
{lisijia,gougaopeng,liuchang,xionggang,lizhen,xiaojunchao}@iie.ac.cn,xinyu.xing@northwestern.edu

ABSTRACT

Phishing scams have become the most serious type of crime involved in Ethereum. However, existing methods ignore the natural camouflage and sparse distribution of phishing scams in Ethereum leading to unsatisfactory performance, and they are also limited by the data scale which cannot be applied to real-world dynamic scenarios. In this paper, we propose a Transaction Graph Contrast network (TGC) to enhance phishing scam detection performance on Ethereum. TGC inputs subgraphs instead of the entire graph for training, which eases the model's requirements for machine configuration and data connectivity. Motivated by phishing nodes are surrounded by normal nodes, we design the comparison between node-level to help phishing nodes learn the unique properties of themselves different from their neighbors. Observing the small number and sparse distribution of phishing nodes, we narrow the distance between phishing nodes by comparing node context-level structures, so as to learn universal transaction patterns. We further combine the obtained features with common statistics to identify phishing addresses. Evaluated on real-world Ethereum phishing scams datasets, our TGC outperforms the state-of-the-art methods in detecting phishing addresses and has obvious advantages in large-scale and dynamic scenarios.

CCS CONCEPTS

• **Applied computing → Digital cash**; • **Security and privacy → Phishing**.

KEYWORDS

Blockchain, Ethereum, Phishing scam detection, graph representation learning

ACM Reference Format:
Sijia Li[1,2], Gaopeng Gou[1,2], Chang Liu[1,2,*], Gang Xiong[1,2], Zhen Li[1,2], Junchao Xiao[1,2], Xinyu Xing[3]. 2023. TGC: Transaction Graph Contrast Network for Ethereum Phishing Scam Detection. In *Annual Computer Security Applications Conference (ACSAC '23), December 04–08, 2023, Austin, TX, USA.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3627106.3627109

1 INTRODUCTION

Ethereum is widely recognized as one of the largest blockchain systems, heralding the advent of blockchain 2.0 [43]. However, despite its success, the rise of decentralized finance (DeFi) and the allure of blockchain's anonymity have unfortunately given rise to a plethora of cybercrimes, with phishing scams garnering significant attention [25]. Disturbingly, phishing scams constitute a substantial portion of malicious fraud within the blockchain ecosystem, accounting for approximately 50% of such incidents [3]. Furthermore, these scams tend to resurface periodically, exacerbating the need to address the identification of phishing scams on Ethereum. Consequently, combating this issue has become an urgent and paramount concern [44], necessitating a sustained, long-term effort.

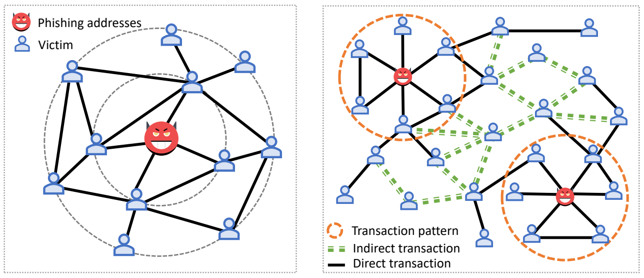


Figure 1: Illustration of the Ethereum phishing scam peculiarities. The phishing address is surrounded by normal addresses. The transaction pattern of the phishing addresses are very similar, but they are not directly connected.

Phishing represents a form of cybercrime that exploits user vulnerabilities with the intent of acquiring personal and confidential information [20]. Within the Ethereum ecosystem, phishing organizations employ enticing tactics to encourage remittances. For instance, they entice investors by offering additional Ether coins as incentives, prompting them to visit fraudulent platforms or websites and provide personal information. Alternatively, these entities promise high returns to induce investors into purchasing digital assets from them [3].

The current approaches for detecting phishing activities on Ethereum primarily rely on the transaction network, leveraging the openness and transparency of the blockchain. These methods can be categorized into two main groups. The first category involves combining traditional machine learning techniques with manually designed features, such as structural and statistical characteristics of nodes, to detect phishing attempts [7]. However, these methods heavily rely on expert knowledge to extract these features, which can be inefficient and non-automated.

The second category involves employing network representation learning techniques to extract deep features from the Ethereum transaction network. This approach utilizes methods like random walk [44, 49] and graph neural networks [4, 41] to automatically learn representations from the transaction network. These techniques offer a higher level of intelligence compared to relying solely on statistical features.

While previous research has made significant strides in developing advanced techniques, they often overlook the unique challenges

*Chang Liu is the corresponding author.

presented by Ethereum phishing scams. These challenges can be summarized as follows:

- **The natural camouflage:** As depicted in the left panel of Figure 1, we observe that phishing addresses tend to have predominantly normal addresses as their neighbors. This contradicts existing methods that rely on the homophily assumption [4, 44], where connected nodes in the graph are presumed to have similar representations. Traditional approaches assume that normal users primarily interact with other normal users, while abnormal users interact predominantly with other abnormal users. However, these methods fail to consider the natural camouflage employed by Ethereum phishing scams. Consequently, directly applying homophily-based techniques to detect Ethereum phishing fraud is not suitable for such scenarios.
- **Sparsity of distribution:** We have discovered that phishing addresses exhibit a low proportion and sparse distribution. Phishing addresses typically avoid transacting with each other to evade detection, as illustrated in the right panel of Figure 1. When similar nodes have no direct connections, the performance of existing graph neural network (GNN)-based convolution detection methods significantly deteriorates. The current methods struggle to learn the transaction patterns' similarity among phishing addresses.
- **Large scale and dynamic nature:** The Ethereum transaction network is both vast and dynamic. Most existing methods rely on transduction approaches that involve training on the entire graph. However, this becomes computationally expensive and impractical for predicting emerging addresses in dynamic networks. Consequently, these methods are ill-suited for Ethereum phishing detection scenarios.

To address these challenges comprehensively, we present a novel approach called **T**ransaction **G**raph **C**ontrast Network (TGC) in this paper. TGC is specifically designed to enhance the detection of phishing scams on the Ethereum platform. Unlike existing methods that require training on the entire graph, TGC operates on subgraphs, thereby reducing the machine configuration and data connectivity requirements, and can be well adapted to dynamic networks. Technically, TGC incorporates two key modules: Node-level contrast and Context-level contrast, inspired by our observation that phishing nodes often have normal nodes as neighbors. In the Node-level contrast module, we maximize the agreement between embeddings of the center nodes in two different views while minimizing agreement with other nodes. This encourages the nodes to learn distinctive characteristics unique to themselves, differentiating them from their neighbors. To tackle the sparsity issue associated with the distribution of phishing nodes, we introduce the Context-level contrast module. This module operates at multiple views derived from the node-level contrasts, enabling us to mine valuable node features. By aligning embeddings of correlated contexts and distinguishing them from negative pairs, we facilitate the learning of universal transaction patterns exhibited by phishing nodes. Finally, the node representations obtained from the contrastive learning process are inputted into a classifier to identify phishing addresses effectively. To validate the effectiveness of TGC, we conduct extensive experiments using real-world datasets.

In summary, the paper makes the following contributions:

- We propose a Transaction Graph Contrast Network (TGC) to mine deep features by designing subgraphs contrast, which both detects Ehtherum phishing scams better and applies to the large-scale dynamic network.
- Based on the observation of the phishing nodes surrounded by normal nodes, we design node-level contrast to learn their unique properties by setting neighbors as their negative samples.
- Considering the sparse distribution of phishing nodes, context-level contrast is proposed to learn the universal transaction patterns of phishing nodes, by narrowing the distance between their different views representations.
- Evaluated on real-world Ethereum phishing scam dataset, TGC outperforms state-of-the-art methods on multiple metrics, and has obvious advantages in large-scale and dynamic scenarios.

2 RELATED WORK

In this section, we provide an overview of prior work on Ethereum phishing scam detection. We then review graph representation learning and contrastive representation learning, which are essential for this task. Finally, we summarize and compare our proposed method with related works in the field.

2.1 Ethereum Phishing Scams Detection

Phishing is an online threat where attackers impersonate legitimate websites to obtain users' private information, like usernames and passwords [1]. Traditional phishing scams involve creating fake platforms. Therefore, the detection effort focuses on analyzing patterns in URLs, source code, CSS styles, and page layouts [15, 26, 30, 33].

Ethereum phishing scams offer a wider array of fraudulent methods compared to traditional scenarios. These scams not only exploit phishing websites but also deceive victims through phishing addresses shared via emails, chat groups and social media [44]. As a result, traditional phishing detection methods are ill-suited for the Ethereum context.

Given the open and transparent nature of blockchain technology, all transactions on Ethereum can be observed. This makes it possible to extract information from transaction records and analyze the transaction behavior between addresses to identify phishing addresses [2, 27]. The transaction history of Ethereum naturally forms a transaction network, where addresses serve as nodes and transactions between addresses as edges. Current methods for detecting phishing scams on Ethereum primarily rely on this transaction network. They learn the representation of phishing nodes from the transaction network and classify nodes accordingly.

Existing methods for detecting phishing scams on Ethereum can be categorized into two main groups.

The first category utilizes shallow models, such as traditional machine learning approaches, coupled with feature engineering techniques that primarily focus on statistical features. For instance, Chen et al. [7] extracted a set of 219 statistical features from the node's 1-order and 2-order neighbors, including in-degree, out-degree, maximum transaction value, and more. They employed a LightGBM-based ensemble machine learning algorithm, along with a Dual-sampling Ensemble technique, to identify phishing nodes.

The second category utilizes graph representation learning methods, including DeepWalk [31], Node2Vec [11], and graph convolutional networks (GCN)[21], to extract deep features (detailed in Section 2.2). Yuan et al.[49] directly employed the Node2Vec algorithm on the Ethereum transaction network to learn node representations. Wu et al.[44] introduced Trans2Vec, a variation of Node2Vec specifically designed for Ethereum phishing detection. Trans2Vec employs a biased sampling process based on the last transaction of two nodes, making it more suitable for detecting phishing on Ethereum. Chen et al.[4] developed E-GCN, the first application of GCN for detecting phishing nodes on Ethereum. They extracted 8-dimensional statistical features and used GCN to learn the structural characteristics of the transaction network. Additionally, Wang et al.[41] proposed TSGN, a variant of the SGN model[46] tailored for the Ethereum transaction ecosystem. TSGN treats transactions between phishing addresses as nodes and connects two nodes represented by transactions if they share a common address.

However, these researches are mostly based on homophily assumption which means connected nodes in the graph have similar representations. In other words, the homophily assumption assumes that normal users interact more with normal users, while abnormal users interact more with abnormal users. Based on the homophily assumption, they can't learn the distinguishing feature between phishing addresses and normal addresses in the Ethereum transaction network. Moreover, the large-scale transaction network where the scarcity of labels and the huge volume of transactions makes it difficult and intricate to take advantage of GNN methods.

2.2 Graph Representation Learning

Graph representation learning (GRL) refers to the process of learning a parametric mapping from the raw graph input data domain to a feature vector or tensor, in the hope of capturing and extracting more abstract and useful concepts that can improve performance on a range of downstream tasks [23].

Traditional methods on graph representation learning are mostly based on random walk. These random walk-based algorithms utilize walk to perceive the centrality and similarity of nodes. DeepWalk [31] obtains multiple node sequences based on random walk, and on this basis tries to maximize the co-occurrence probability of nodes in the window. As for Node2Vec [11], in the first stage of generating nodes' corpus, its design walking decision is more flexible than DeepWalk, and the field of view can be selected deeper or wider through parameters, but the time consumption increases greatly. Different from DeepWalk, Large-scale Information Network Embedding (LINE) [34] aims to generate neighbors rather than nodes on a path based on current nodes. LINE [34] learns a low-dimensional embedding via preserving the first-order and second-order closeness of nodes.

Unlike the above shallow models, recent work mainly uses deep neural networks to learn non-linear information in graphs. Structural Deep Network Embedding (SDNE)[40] apply deep autoencoders to keep network proximities within 2-order. It uses a semi-supervised autoencoder to reconstruct the neighbor relationships of the nodes and uses a supervised approach to trim the results. As for the work on Graph Neural Networks (GNN), they employ graph convolutional encoders which are more powerful than traditional methods for representation learning. GraphSAGE[14] is an inductive GNN model based on a fixed sample number of the neighbor nodes and incorporates DeepWalk-like objectives as well. Variational graph auto-encoders (VGAE) [22] reconstruct the adjacency matrix to learn the generative distribution of graphs. Graph Attention Networks (GAT)[38] employs attention mechanism for neighbor aggregation. DGI [39] marries the power of GNN and contrastive learning, which focuses on maximizing Mutual Information (MI) between global graph embeddings and local node embeddings.

2.3 Contrastive Representation Learning

Intuitively, contrastive representation learning can be considered as learning representation by comparing among the input samples. The comparison can be performed between positive pairs of "similar" inputs and negative pairs of "dissimilar" inputs. Through handcrafted contrastive pretext tasks, these approaches learn discriminative representations by contrasting positive instance pairs against negative instance pairs [6]. In the process of learning representation, labels are "pseudo-labels" generated by specific method design, so contrastive learning is a significant branch of self-supervised learning (SSL) [28].

Many traditional graph representation learning methods also embody the concept of contrastive learning, DeepWalk [31] and Node2Vec [11] model probabilities of node co-occurrence pairs using noise-contrastive estimation, nodes appearing in the same random walk are considered as positive samples. Following the immense success of contrastive learning on CV [12, 17, 35, 45] and NLP [8, 9, 47], some recent works also exploit contrastive methods to graph representation learning on deep graph neural networks. DGI [39] considers a node's representation and graph-level summary vector obtained by a readout function as a contrastive instance pair, and generates negative samples with graph corruption. On the basis of DGI, Hassani et al. [16] suggests a multi-view contrastive learning framework by viewing the original graph structure and graph diffusion [10] as two different views. GraphCL [48] adopts SimCLR [6] to form its contrastive pipeline which pulls the graph-level representations of two views closer.

Comparisons with related GRL methods. To adapt to Ethereum phishing scam detection, our proposed TGC framework has essential differences in motivation and implementation compared with the above methods. From motivation, these methods focus on learning representations for all nodes indiscriminately. Instead, our proposed TGC focuses on detecting a minority class of Ethereum phishing addresses based on phishing peculiarities. From implementation, since the existing contrastive learning instance pair definitions cannot effectively capture Ethereum phishing scams, we design a new type of comparison instance pair that combines node-level contrast and context-level contrast learning, which fits the natural camouflage and sparsity of distribution of Ethereum, and the training method based on the subgraph can be applied to large-scale dynamic transaction networks in the real world.

3 DESIGN OF TGC

Detecting phishing scams in Ethereum is divided into three steps: data collection, address representation and address detection. We describe the data collection process in detail in Section 4.2. In the following section, we first give a description of the problem formulation of phishing identification. Then, we introduce the overall transaction graph contrast framework to represent addresses. Finally, we briefly describe the classification process for Ethereum address detection.

3.1 Problem Definition

In this paper, the Ethereum phishing scam detection task is phrased as a graph node classification problem. Given a set of addresses and their transactions on Ethereum, we can construct the transaction network as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, we treat the transaction address as a node v_i, $\mathcal{V} = \{v_1, \ldots, v_N\}$ is a set of addresses. The transaction as an edge $\mathcal{E}_i$, $\mathcal{E} = \{\mathcal{E}_1, \ldots, \mathcal{E}_R\}$ is the transaction set. We denote the feature matrix and the adjacency matrix as $X \in \mathbb{R}^{N \times F}$ and $A \in \{0,1\}^{N \times N}$. Our goal is to learn a graph representation $f(X, A) \in \mathbb{R}^{N \times F'}$ based on known nodes and transaction information, which can effectively represent unseen nodes in Ethereum, improving phishing scam detection performance in a large-scale dynamic Ethereum transaction network. It is worth noting that we do not use labels Y during the representation learning process, labels are only used in the classifier training.

3.2 Model Architecture

TGC is a graph representation framework to detect phishing addresses. In this section, we describe the general framework of our proposed TGC. As shown in Figure 2, the architecture could be divided into four objectives: ego network construct, random walk sample, node-level contrast, and context-level contrast.

At first, to generate the basic learning samples for contrastive learning, we construct each sample node's local substructures, named "ego network", which is composed of the r-order neighbors of the central sample node and the connection relationship among them. The construction of the ego network lays a good foundation for the subsequent design of comparing pairs and subgraph training strategies. Next, we execute instance pair sampling with random walk with restart (RWR) sampling strategy. Based on the ego network of each sample obtained in the previous step, we sample each ego network twice with RWR, so two local subgraphs are generated based on the ego network of each sample. In this step, we can get a pair of well-designed "local subgraph vs. local subgraph" for each sample. In the end, we feed the instance pairs into the designed node-level contrast and context-level module to learn node representations. In the node-level contrast module, we input the pair of local subgraphs generated by the same sample. we define the positive pair as the same center nodes in different subgraphs, while the negative pairs as the center node and the other nods in subgraphs. our design is to force the phishing nodes to learn discriminative representations different from their neighbors. In the context-level contrast module, we input local subgraphs generated by multiple samples, local subgraphs generated by the same ego network are regarded as positive sample pairs, and those generated by different ego networks are regarded as negative sample pairs. Our designed instance pairs can learn the transaction patterns by paying close attention to the contextual information of each node. We will introduce the main modules of the TGC architecture in detail in the following.

3.2.1 *Ego Network Construct.*

As mentioned in Section 2.3, SSL models are learned by solving a series of pretext tasks. Pretext tasks refer to the predesigned tasks, which help models to learn more generalized representations from unlabeled data, and thus benefit downstream tasks by providing a better initialization or more effective regularization [29]. In contrastive learning, the pretext tasks aim to estimate and maximize the MI between positive pairs and minimize the Mutual Information (MI) between negative pairs.

Therefore, the definition of the data instance is very important for contrastive learning. It is straightforward for CV and NLP tasks to define an instance as an image or a sentence. However, such ideas cannot be directly extended to Ethereum transaction graph data. Some previous work on graphs has proposed some instance pairs like "full graph v.s. node", "large subgraph v.s. node" and so on [16, 39, 48], but most of them input the whole graph for training. In the Ethereum scenario, the training method of directly inputting the entire large-scale transaction graph leads to high computational consumption, and can not be applied in transaction networks with dynamic changes in the real world. The design of the data instance also did not pay attention to the local property of Ethereum phishing addresses, which is essential for learning the transaction pattern.

To address these issues, we propose to extend each training sample node to its local structure, and then design contrastive instances on this basis. Specifically, for a certain node v, we define its r-ego network:

Definition 1 (A r-ego network). *Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{v_1, \ldots, v_N\}$, $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$, For a node v, its r-neighbors are defined as $S_v = \{u : d(u, v) \le r\}$ where $d(u, v)$ is the shortest path distance between u and v in the graph $\mathcal{G}$. The r-ego network of node v, denoted by $\mathcal{G}_v$, is the sub-graph induced by S_v.*

We extract the neighbors and transaction information of each training sample, and construct the r-ego network of the sample as the input of the next step, the sample node is the center node of its r-ego network. The left panel of Figure 1 shows an example of a 2-ego network.

Because of the anonymity of the blockchain platform, the node itself does not carry any attribute characteristics. The ego networks will provide essential node features for subsequent TGC learning of the structural features of nodes. we extract the following 10-dimensional features as the attribute characteristics of the node: the node's total degree, out-degree, in degree, the sum of transactions amount, transfer out transaction amount, transfer in transaction amount, the total number of neighbors, the inverse of transaction frequency, the percentage of neighbors whose transactions are all zeros, and the number of transactions with the most frequent neighbors.

3.2.2 *Subgraph Sampling.*

Contrastive pretext tasks and data augmentation are both measures to enrich supervision signals. Unlike pretext tasks, which train models by setting positive and negative sample pairs, data augmentation helps the model explore richer underlying semantic information by making pretext tasks more difficult to solve [42]. In our TGC method, we build two different views for the subsequent contrastive learning, also known as the data

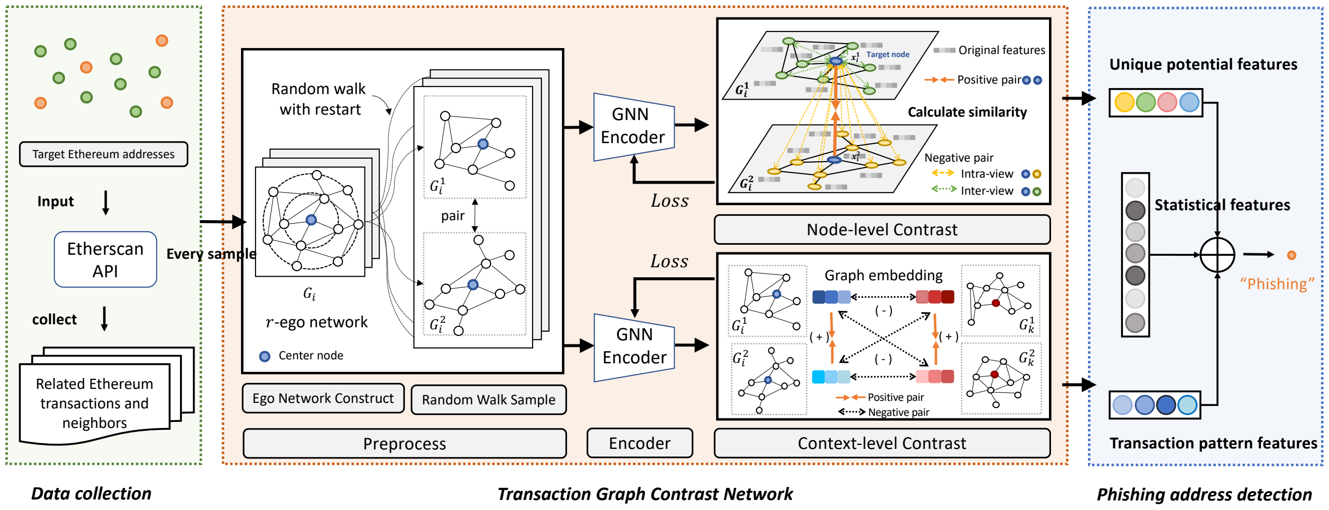


Figure 2: The overall framework of TGC. TGC inputs the related transactions and neighbors of the train samples to learn their representation. The two preprocessing modules construct the sample centered r-ego networks and build two different views based on them as the input of contrastive learning. The node-level contrast module forces nodes to learn distinguishing features different from neighbors to build unique potential features, while the context-level contrast module captures the universal transaction patterns of phishing nodes. Finally, the outputs of several modules are combined as the final representation of the node, which feeds into the classifier to get the result.

augmentation step, which helps the algorithms learn expressive representations [18].

However, unlike the regular grid-like image data where different views can be simply generated by standard augmentation techniques such as cropping or color distortion, the view augmentation on irregular graph data is not trivial, as graph nodes and edges do not contain visually semantic contents as in the image [39].

Although feature masking is a simple way to generate a related graph, it might damage the rare features of Ethereum addresses, thus degrading the representation results of graph convolutions [16]. An effective way to data augmentation for graph contrastive learning is to apply random walk with restart (RWR), by iteratively generating subgraph structure via random walk with a restart probability [50]. Instead of directly changing the features, we employ RWR to generate two different views revealing the important structural cues. In RWR sampling, the restart probability controls the radius of the ego-network (i.e., r) which TGC conducts data augmentation on. In this work, we follow [32] to use 0.8 as the restart probability. The proposed TGC framework is flexible to other graph sampling algorithms, such as neighborhood sampling [13] and forest fire [24].

By this mean, the representations from different views complement each other and thus enriching the final representation results. Specifically, by performing different levels of contrastive learning between the obtained representations from two views, the rich properties and transaction pattern information can be encoded simultaneously. The details will be described in Section 3.2.3 and 3.2.4.

3.2.3 Node-level Contrast. In this module, we force nodes to learn unique properties of themselves different from their neighbors by treating neighbors as their negative samples.

According to our observation on the neighbors of phishing nodes, 97.99% of the neighbors around phishing nodes are normal nodes (Detailed analysis in Appendix A). The GNN aggregation mechanism is based on homophily assumptions, so the distinctive representation of phishing nodes cannot be obtained by directly aggregating neighbor features through GNN.

In order for nodes to learn unique characteristics different from their neighbors, in this section, we define the contrastive instance pair as "target node v.s. node". We use node-level contrast loss to optimize the GNN encoder, which encodes the r-ego network, regarding the central nodes of different sampling subgraphs generated by the same ego network as positive sample pairs while all the other nodes form negative samples.

Specifically, the first element in the instance pair is our target node, and the target node refers to the central node in the local subgraph, which is obtained through the r-ego network RWR of the sample. The target node can be set as the center node of the local subgraph of any sample in the training set. For positive instance pairs, the second element in the instance pair is the central node of another local subgraph of the target node, for example, the blue node pair in Figure 3. For negative instance pairs, it is randomly selected from all neighbor nodes except the target node in both of the input subgraphs.

Formally, self-supervised contrastive learning is expected to achieve the effect as

$$\operatorname{score}\left(f\left(\mathbf{x}_i\right), f\left(\mathbf{x}_i^{+}\right)\right) \gg \operatorname{score}\left(f\left(\mathbf{x}_i\right), f\left(\mathbf{x}_i^{-}\right)\right) \quad (1)$$

where $\mathbf{x}_i^{+}$ is a node similar to $\mathbf{x}_i$, $\mathbf{x}_i^{-}$ is a node dissimilar to $\mathbf{x}_i$, f is the graph encoder, and the score function is used to measure the similarity of encoded features of two nodes. Here, $(\mathbf{x}_i, \mathbf{x}_i^{+})$ and $(\mathbf{x}_i, \mathbf{x}_i^{-})$ indicate the positive and negative instance pairs, respectively. Eq.(1) encourages the score function to assign large values

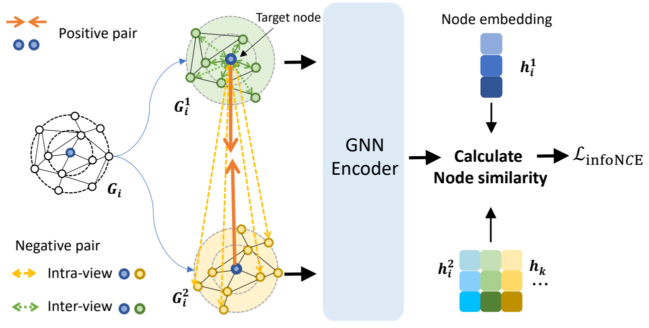


Figure 3: Illustration of the node-level contrast module.

to the positive pairs and small values to the negative pairs, which can be used as the supervision signals to guide the learning process of encoder f. By resorting to the above explanations and definition of instance pairs, our node-level self-supervised contrastive loss $\mathcal{L}_n$ can be presented as

$$\mathcal{L}_n = \frac{1}{2N} \sum_{i=1}^{N} \left[\ell_n \left(\mathbf{x}_i^1 \right) + \ell_n \left(\mathbf{x}_i^2 \right) \right] \tag{2}$$

where $\ell_n \left(\mathbf{x}_i^1 \right)$ and $\ell_n \left(\mathbf{x}_i^2 \right)$ respectively denote the self-supervised node-level pairwise contrastive losses of the central node $\mathbf{x}_i$ in two views $\mathcal{G}_i^1$ and $\mathcal{G}_i^2$, the two views generated by the r-ego network of the sample node $\mathbf{x}_i$.

We employ a contrastive objective that enforces the encoded embeddings of the center node in the two different views to agree with each other and can be discriminated from embeddings of other nodes. For each center node $\mathbf{x}_i$, it's embedding $\boldsymbol{h}_i^1$ generated in one view, is treated as the anchor, the embedding of it generated in the other view, $\boldsymbol{h}_i^2$, forms the positive sample, and we regard the other embeddings in the two views as negative samples. $\ell_n \left(\mathbf{x}_i^1 \right)$ can be obtained with the cosine similarity measured, namely

$$\ell_n \left(\mathbf{x}_i^1 \right) = -\log \frac{e^{\theta\left(\boldsymbol{h}_i^1, \boldsymbol{h}_i^2\right)/\tau}}{e^{\theta\left(\boldsymbol{h}_i^1, \boldsymbol{h}_i^2\right)/\tau} + \sum_{k \neq i} e^{\theta\left(\boldsymbol{h}_i^1, \boldsymbol{h}_k\right)/\tau}} \tag{3}$$

where τ is a temperature parameter, $\theta(u, v)$ is the cosine similarity, $\boldsymbol{h}_k$ is the embedding of negative instances of node $\mathbf{x}_i^1$.

The central node in the two views is a positive pair, and we define negative instances as all other neighbor nodes in the two views. Therefore, negative instances come from two sources, that are inter-view and intra-view nodes, corresponding to the second term in the denominator in Eq.(3), which can also be written as

$$\sum_{k \neq i} e^{\theta\left(\boldsymbol{h}_i^1, \boldsymbol{h}_k\right)/\tau} = \sum_{k \neq c} e^{\theta\left(\boldsymbol{h}_i^1, \boldsymbol{h}_k^1\right)/\tau} + \sum_{k \neq c} e^{\theta\left(\boldsymbol{h}_i^1, \boldsymbol{h}_k^2\right)/\tau} \tag{4}$$

where $\boldsymbol{h}_k^1$ is the embedding of intra-view negative instances, $\boldsymbol{h}_k^2$ is the embedding of inter-view negative instances.

3.2.4 *Context-level Contrast.*

In this module, we capture the transactional structural patterns behind phishing and normal addresses by distinguishing the context structure between them.

According to our statistical analysis, in the second-order connected graph constructed around all phishing addresses, the proportion of phishing nodes is only about 0.345%, and the distribution of phishing nodes is very sparse (Detailed analysis in Appendix A). When the scale of the transaction graph is large, inputting the entire connected graph for node representation learning cannot focus on the local transaction patterns of nodes, nor learning the same transaction patterns of different phishing nodes.

In order to capture the transactional structural patterns behind phishing and normal addresses, in this section, we propose to define contrastive instances pair as "target context v.s. context". As Figure 4 shows, the first element in the instance pair is a subgraph containing context information, which is generated by RWR on the r-ego network of the central node. The central node here refers to the training sample, which can be set to any node in the training set. The second element is also a subgraph containing context information, forming an instance pair with the target subgraph. When the context subgraph and the target context subgraph are generated by the same center node's r-ego network, they form a positive instance pair. In contrast, when the two context subgraphs are generated by different central nodes, they form a negative instance pair.

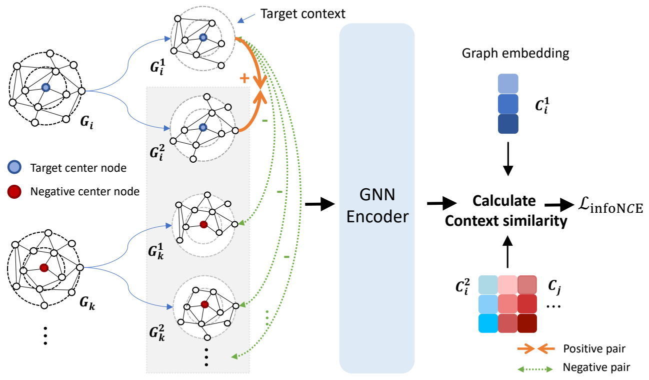


Figure 4: Illustration of the context-level contrast module.

So we consider two random augmentations of the same r-ego network as a positive instance pair and augmentations of different r-ego networks are all negative pairs. Because there are no supervision signals, the training task treats each context instance as a distinct class of its own and learns to discriminate between these instances. We propose to use InfoNCE [36] as our learning objective, and the overall loss function of the context-level contrast module can be formulated as

$$\mathcal{L}_c = \frac{1}{2N} \sum_{i=1}^{N} \left[\ell_c \left(\mathcal{G}_i^1 \right) + \ell_c \left(\mathcal{G}_i^2 \right) \right] \tag{5}$$

$$\ell_c \left(\mathcal{G}_i^1 \right) = -\log \frac{e^{\left(C_i^{1\top} C_i^2\right)/\tau}}{\sum_{j=1}^{N} e^{\left(C_i^{1\top} C_j\right)/\tau}} \tag{6}$$

where $\ell_c \left(\mathcal{G}_i^1 \right)$ and $\ell_c \left(\mathcal{G}_i^2 \right)$ respectively denote the self-supervised context-level pairwise contrastive losses of the central node $\mathbf{x}_i$ in two views $\mathcal{G}_i^1$ and $\mathcal{G}_i^2$, the two views generated by the r-ego network of the sample node $\mathbf{x}_i$. C_i^1 and C_i^2 are the context embedding of $\mathcal{G}_i^1$ and $\mathcal{G}_i^2$ respectively, and C_j represents the context embedding of data augmentation generated by the r-ego network of node $\mathbf{x}_j$.

In context-level contrast tasks, we compare the overall representation of the context rather than the central node, so we need an injective readout function to produce the context-level embedding.

The target of the readout function is to transfer the embeddings of nodes in subgraph $\mathcal{G}_i^*$ into a local subgraph embedding vector C_i^*. In this step, we use the average pooling functions as our readout module, which has been widely used in previous works [16]. Specifically, the readout function can be presented as

$$C_i^* = \text{Readout}\left(\mathbf{h}_i^*\right) = -\frac{1}{n}\sum_{i=1}^{n}\mathbf{h}_i^* \ (* \in \{1,2\}) \tag{7}$$

where n is the number of nodes in the subgraph $\mathcal{G}_i^*$, $\boldsymbol{h}_i^*$ is the representation of node $\mathbf{x}_i^*$ in subgraph $\mathcal{G}_i^*$.

Define graph encoders. For comparison tasks at different levels, we encode them via two graph neural network encoders f_n and f_c, respectively. Technically, any graph neural network can be used here as the encoder, and the TGC model is not sensitive to different choices. In practice, we adopt the GAT [38], a classical graph neural network model, as our graph encoder.

Given N_u denotes the adjacent nodes of node u and node $v \in N_u$, the importance node-node pair $\langle u, v\rangle$ can be formulated as follows:

$$\begin{aligned} e_{uv}^{\Phi} &= \sigma\left(a_{\Phi}^{T} \cdot [h_u \| h_v]\right) \\ \alpha_{uv}^{\Phi} &= \text{softmax}_{\text{v}}\left(e_{uv}^{\Phi}\right) = \frac{\exp\left(e_{uv}^{\Phi}\right)}{\sum_{k \in N_u^{\Phi}} \exp\left(e_{uk}^{\Phi}\right)} \end{aligned} \tag{8}$$

where h_u and h_v are the features of node u and node v, a_Φ is the attention parameterize matrix for transaction graph Φ, σ denotes the activation function, and || denotes the concatenate operation.

Then, the node u representation can be obtained by aggregating all neighbor attributes with the corresponding coefficients as follows:

$$z_u^{\Phi} = \|_{k=1}^{K} \sigma\left(\sum_{v \in N_u^{\Phi}} \alpha_{uv}^{k} \cdot h_v\right) \tag{9}$$

where z_u^{Φ} is the learned embedding of node u for the transaction graph Φ, K is the head number using the multi-head attention mechanism[37].

3.3 Phishing Addresses Detection

The task of this section is to classify nodes to distinguish between phishing nodes and normal nodes. After the above operations, we have obtained three types of features: unique potential features learned from *Node-level Contrast* module, transaction pattern features learned from *Context-level Contrast* module, and statistical features obtained from nodes. We splice them together as the complete representation of the node. On the basis of obtaining complete node representations, we need to learn the difference between phishing and normal node representations. So we input them into the classifier for Ethereum phishing addresses classification.

There are many choices of classifiers, and in this paper, we choose XGBoost [5], which is a new GBDT (Gradient Boosting Decision Tree) algorithm supporting efficient parallel training. The key concept behind GBDT is to iteratively train the weak classifier (decision tree) to get the optimal model. The fitting process of XGBoost uses the second-order Taylor expansion of the loss function, which is different from the traditional GBDT. Parallel and distributed computing make learning faster which enables quicker model exploration.

Dataset	#Total Nodes	#Labeled	#Edges	#Average Degree
D_1	30,000	106	24,965,770	832.2201
D_2	40,000	140	27,642,111	691.0701
D_3	50,000	166	31,597,197	631.9566
D_4	60,000	207	33,072,308	551.2143
D_5	70,000	238	34,450,265	492.1537
D_6	80,000	269	35,872,229	450.3111
D_p	9,237,535	5,639	219,927,673	23.8080

Table 1: Statistics of evaluation datasets. Labeled represents the number of labeled phishing nodes in the dataset, and each number is the average calculated by five subgraphs.

4 EXPERIMENTS

In this section, we perform empirical evaluations to demonstrate the effectiveness of the proposed TGC framework. We first construct the network datasets by exploiting transaction records from Ethereum, then we introduce the experimental setup, including baselines, evaluation metrics and parameter settings. Finally, we proceed to detail the experimental results and their analysis.

4.1 Datasets

According to the authorized website Etherscan[1], as of March 2023, 5,639 addresses have been verified to be phishing addresses. We crawl these phishing addresses from the Ethereum label cloud of Etherscan. In order to simulate the scene with unbalanced categories in the real environment, we randomly select 25000 active normal addresses at the same time. Taking these labeled phishing addresses and normal addresses as the central nodes, we extract their first-order, second-order neighbors and the transactions between all of them through the API provided by Etherscan. Finally, we obtain 9,237,535 Ethereum addresses and 219,927,673 transaction records. Based on the obtained transactions, we construct 30,639 transaction ego networks named D_p, which support the comparative experiments of Section 4.5 and 4.6.

In order to compare with the transductive methods of inputting whole graph training, in the comparative experiments, we refer to the data construction steps of [7]. We generate a large graph based on the transaction information crawled around all labeled phishing nodes, and select the largest connected component. Then we sample with random walks to obtain subgraphs of different sizes. These connected subgraphs serve as the entire large graph for our comparison model and TGC input. Dataset denoted as D_1, D_2, D_3, D_4, D_5, and D_6 with sizes of 30,000, 40,000, 50,000, 60,000, 70,000, and 80,000 respectively. For each subgraph of different sizes, we sample five times to ensure the effectiveness of the performance. D_1, D_2 and D_3 support the comparative experiments of Section 4.3 whlie D_4, D_5 and D_6 support Section 4.4, 4.7 and 4.8.

[1]https://etherscan.io

Method	Training Data	D_1			D_2			D_3		
		F1	Recall	Pre	F1	Recall	Pre	F1	Recall	Pre
Only Features	X	0.7713	0.7572	0.7859	0.7459	0.7759	0.7181	0.7825	0.7484	0.8198
DeepWalk	A	0.7108	0.7572	0.6697	0.7597	0.7747	0.7452	0.7639	0.7097	0.8271
Node2vec	A	0.7478	0.7624	0.7337	0.7931	0.8182	0.7695	0.7496	0.8065	0.7003
LINE	A	0.7990	0.8721	0.7373	0.7636	0.7470	0.7810	0.7666	0.8000	0.7359
SDNE	A	0.7447	0.6492	0.8732	0.7377	0.6838	0.8009	0.7472	0.7581	0.7367
GraphSAGE	X, A	0.8027	0.7709	0.8372	0.8273	0.8617	0.7956	0.8257	0.8097	0.8423
GAT	X, A,Y	0.8110	0.7749	0.8506	0.8577	0.8458	0.8699	0.8453	0.8194	0.8729
E-GCN	X, A, Y	0.8136	0.8796	0.7568	0.8650	0.8735	0.8566	0.8670	0.8516	0.8829
TSGN	X, A, Y	0.8174	0.7382	0.9156	0.8444	0.8261	0.8636	0.8892	0.9065	0.8727
TGC	X, A	**0.9261**	**0.9164**	**0.9360**	**0.9528**	**0.9565**	**0.9490**	**0.9550**	**0.9581**	**0.9519**

Table 2: Performance comparison results of F1-score, Recall and Precision on three datasets. Available data for each method during the learning representation phase is shown in the second column, where X, A, Y correspond to node features, the adjacency matrix, and labels respectively.

4.2 Experimental Setup

Comparison Methods. We compare our proposed TGC framework with three categories of Ethereum phishing scams detection methods, including (1) Feature-based methods where only the node attributes are considered [7], (2) Random walk-based GRL methods (i.e., DeepWalk [31], Node2Vec [11], and LINE [34]) that take topological information into consideration. In addition, we also use some of the popular (3) Deep learning-based GRL methods (SDNE [40], GraphSage [14], GAT [38], E-GCN [4] and TSGN [41]) to learn node representations to compare with the representations learned by our method TGC.

Evaluation Metrics. In this paper, we use the following three metrics to have a comprehensive evaluation of the performance of different methods in terms of Ethereum phishing scam detection: **(1) Recall.** The recall rate means the percentage of known phishing node samples detected. **(2) Precision.** The precision rate means the percentage of real phishing nodes are in the accounts that are judged to be suspicious. **(3) F1-score.** F1-score is a comprehensive evaluation of the Precision and Recall score.

Implementation Details. We use the mini-batch size of 128. For node-level contrast and context-level contrast, the temperature τ is set as 0.4 and 0.2 respectively, and we adopt GAT with 2 layers and 32 hidden units each layer as our encoders, we set the attention hidden size to 4 and the learning rate to 0.005. All models are implemented using Deep Graph Library 0.4.3, PyTorch 1.8.0, and NetworkX 2.5.1. All experiments are conducted on a computer server with three NVIDIA Tesla V100S GPUs (32GB memory each), Intel(R) Xeon(R) Silver 4110 CPU (2.10GHz) and 125 GB of RAM. For more experimental details, please refer to Appendix B.

4.3 Conventional Comparison Results

First, we conduct comparative experiments on a conventional data scale adopted by many studies. We evaluate the performance of all the compared methods in the task of Ethereum phishing scam detection on D_1,D_2 and D_3. The corresponding results are reported in Table 2.

4.3.1 Analysis of Conventional Comparison Results. From the Table 2, we can draw the following conclusions:

(1) Our approach TGC outperforms all the other compared methods by a significant margin. Our method TGC achieves the best performance at about 95.50% F1-score, 95.81% Recall and 95.19% precision under the D_3 dataset. The following method is deep learning methods which reach the F1-score around 85%. The performances of the random walk-based method and the feature-based method are similar, and their indicators are both around 75%. TGC consistently performs better than unsupervised baselines and is competitive with models trained with label supervision on all three datasets, the strong performance verifies the superiority of the proposed contrastive learning framework.

(2) TGC has better node representation capability than existing Ethereum phishing detection methods because we pay attention to the camouflage and sparsity of Ethereum phishing. Compared with the TSGN, E-GCN and features-only methods, the F1-score difference on D_3 is 6.58%, 8.80% and 17.25%, respectively. TSGN only extracts the transaction of the node's first-order neighbors and heavily relies on the preset graph proximity matrix, while E-GCN based on the entire graph cannot learn the transaction patterns of sparsely distributed phishing nodes, which we can obtain from our node-level contrast and context-level contrast modules. These results demonstrated TGC can fully mine addresses' unique properties and transaction patterns which enriches the features of the nodes and strengthens the nodes' representation ability.

(3) Compared with the feature-based methods, our four evaluation metrics are nearly 17% higher than them due to rich structure and contextual information awareness. Among existing Ethereum phishing detection methods, the performance of feature-only methods is the worst. When the dataset is small, its effect is better than the factorization-based method, but as the number of nodes increases, the information that the statistical features can learn is very limited. Apart from the lack of feature mining, it may be because of these methods' unawareness of the network structure and environment information that we obtain from the designed ego network and the context-level module.

(4) As for the random walk-based methods, LINE performed the best, which is lower than our method 18.84% F1-score on the D_3 dataset. LINE uses the deep excavation of proximity within the second-order, through it LINE can perceive the nearby information than Deep Walk and Node2Vec. However, this type of method focuses too much on domain similarity, resulting in nodes failing to learn discriminative features different from their neighbors. In our method TGC, we design the node-level contrast module to force nodes to learn unique features by setting neighbors as negative samples.

(5) Graph representation methods based on deep learning are our strong opponents, however, they are also not performing well. On dataset D_3, our three evaluation metrics are nearly 12% higher than it, which we believe can be attributed to their naive method of selecting negative samples that simply chooses contrastive pairs based on edges. This fact further demonstrates the important role of selecting negative samples in contrastive representation learning. The superior performance of TGC compared to GAT also once again verifies the effectiveness of our proposed TGC framework that contrasts nodes across graph views.

4.3.2 Case study. We think analyzing false positives and false negatives is very important, which can help us understand the nature of the problem and the shortcomings of the model. In this section, we analyzed 320 misclassified addresses in D_3 to find out the reasons for misclassification.

For false negatives, the pattern of phishing addresses usually does not match the two patterns we listed. For example, phishing address 0x017F86B90a46D8Fd999EAeFda1339355b98dA12F has only 2 transactions (one in and one out) and two neighbors, so we cannot correctly extract its transaction characteristics for recognition. For false positives, the benign address may have the characteristics of what we consider a phishing address, like some star addresses, they receive ether transfers from multiple users and only send out to a few addresses. Moreover, our subgraph is sampled, and the neighbors of the central node in the subgraph are not complete. The benign nodes misidentified as phishing nodes may have similar transaction patterns with phishing nodes after sampling, but the two may not be similar before sampling, the same as false negatives, which is the trade-off between the detection effect and computing performance.

4.4 Large-scale Data Comparison Results

In fact, the real Ethereum transaction data scale is much larger than the conventional data scale. In this part, we conduct a data scale experiment on D_4, D_5 and D_6, which compares the processing effects and capabilities of different methods on large-scale Ethereum transaction data. As Table 3 shows:

(1) The TGC subgraph sampling training method can remain lightweight in large-scale network scenarios. With the further increase of the data scale, the transductive methods of inputting the entire large graph for training have been unable to detect phishing scams due to limited machine capabilities, such as GAT and E-GCN. However, TGC can still maintain a strong performance by reducing the computational complexity through the subgraph training mode.

(2) TGC has better node representation capability and stable performance than other methods on large graphs. As the number of dataset nodes increases from 60,000 to 80,000, the gap between TGC and other comparison methods remains and even further widens. These results again demonstrate that TGC can better detect phishing addresses on large-scale transaction networks by fully mining interaction relationships, transaction patterns and unique characteristics of Ethereum addresses.

Method	D_4	D_5	D_6
Only Features	0.7850	0.8010	0.7806
DeepWalk	0.7104	0.7111	0.7075
Node2vec	0.7577	0.7477	0.7534
LINE	0.7637	0.7842	0.7794
SDNE	0.7239	0.7273	0.7056
GraphSAGE	0.8105	0.7938	OOM
GAT	OOM	OOM	OOM
E-GCN	OOM	OOM	OOM
TSGN	0.8286	0.8595	0.8878
TGC	**0.9538**	**0.9600**	**0.9580**

Table 3: Performance comparison results of F1-score on three larger datasets. OOM indicates Out-Of-Memory on a 32GB GPU.

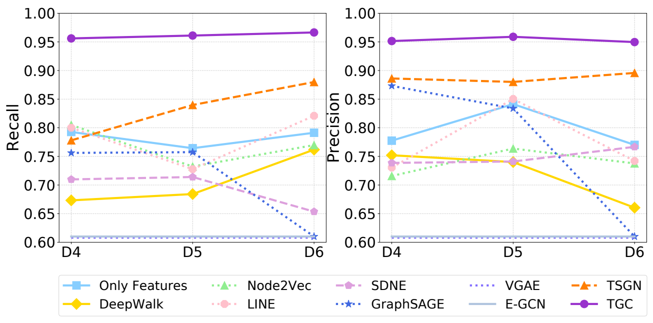


Figure 5: Performance comparison results of Recall and Precision on three larger datasets.

4.5 Dynamic Data Comparison Results

As time goes on, the Ethereum transaction network changes dynamically. The phishing detection method should be able to effectively identify the newly added nodes, not just detect the seen nodes. Therefore, based on the data described in Section 4.2, we conduct an experiment on the detection of Ethereum phishing addresses on a dynamic network.

In order to build a dynamic Ethereum transaction network, we use the transaction data set before a specific time as the training set, and the subsequent ones as the test set. As for the choice of data division time point, in order to ensure the balance of the data, we make the distribution of phishing addresses in the training set consistent with the test set. We take the time of the first transaction of the phishing node as its creation date, count the creation time distribution of the phishing node, and finally choose January 2019 as the time point for dividing the data, detailed data time distribution can be found in Appendix B.

We train the model using transactions before January 2019, and then use the trained model to detect addresses occurring between

January 2019 and March 2023. The details of the training set and test set are shown in Table 4.

Dataset	#Total Nodes	#Labeled	#Edges	#Average Degree
Train	4,824,725	2,273	24,965,770	5.1745
Test	4,412,810	2,355	27,642,111	6.2640

Table 4: Statistics of dynamic network evaluation datasets. Labeled represents the number of labeled phishing nodes in the dataset.

Most of the existing Ethereum phishing detection methods input the whole graph for training, which requires the connectivity of the graph, and cannot predict new nodes that are not in the whole graph. Therefore, in the dynamic network experiment setting of the comparison method with connectivity requirements, for the test set nodes connected to the training set, we use its model to predict normally, and for the test set nodes not connected to the training set, we consider that it is unpredictable and sets its result to 0. From Table 6 we can see that:

Method	Retrain	Conn	F1	Recall	Pre
Only Features	✗	✗	0.7260	0.6333	0.8506
E-GCN	✓	✓	0.5009	0.4856	0.5172
TSGN	✗	✗	0.7663	0.7887	0.7451
TGC	✗	✗	**0.9237**	**0.9291**	**0.9183**

Table 5: Performance comparison results on dynamic network datasets. Conn indicates whether the model requires the connectivity of the entire graph.

(1) TGC can detect emerging addresses in real-world scenarios without model retraining, and has no requirement on the overall connectivity of the transaction network. While the transductive method E-GCN cannot predict unseen nodes that are not connected to the training set.

(2) Compared with existing TSGN methods trained on subgraphs, TGC obtains competitive prediction results with a 16% higher F1-score. The superior performance of TGC verifies that our proposed subgraph training combine contrastive learning scheme is able to help improve embedding quality by learning important transaction patterns and unique properties in the case of real-world large-scale graphs, and can be applied to the dynamic network.

In summary, the superior performance of TGC verifies that our proposed subgraph training combine contrastive learning scheme is able to help improve embedding quality by learning important transaction patterns and unique properties in the case of real-world large-scale graphs, and can be applied to dynamic networks.

4.6 Few-shot Analysis

In the real world, labeled data is usually scarce, and the acquisition of Ethereum phishing labels usually requires professional analysis, which is very expensive and time-consuming. Therefore, it is very meaningful and necessary to conduct a few-shot analysis in Ethereum phishing detection scenarios.

Method	100%	80%	50%	20%	10%
Only Features	0.7260	0.7154	0.7092	0.6870	0.6610
E-GCN	0.5009	0.5283	0.4091	0.2292	0.2087
TSGN	0.7663	0.7280	0.5800	0.4554	0.1473
TGC	**0.9237**	**0.8983**	**0.8635**	**0.8750**	**0.8409**

Table 6: Comparison results of F1-score on few-shot setting. The percentage means the proportion of labeled samples used in the training process.

To validate the effectiveness and robustness of TGC in few-shot settings, we design comparison experiments with different data proportions on the dynamic datasets used in Section 4.5. We select 80%, 50%, 20% and 10% of the samples from the training set to train the classifier. In the process of learning representation, TGC still uses the overall unlabeled data for contrastive learning. In Table 4.6, the comparison results illustrate that our proposed TGC method is least affected by the reduction of data size. The F1 scores of TGC with 80%,50%, 20%, 10% data size are respectively 89.83%,86.35%, 87.50% and 84.09%. Our model achieves the best results among all methods. In contrast, traditional supervised methods (e.g. E-GCN, TSGN) show substantial F1 performance degradation when the sample size is reduced, e.g. TSGN's performance drops to 14.73% when the sample size is reduced from the full size to 10%. This indicates that the contrastive learning approach with pre-training solves the few-shot Ethereum phishing scam detection problem more effectively.

4.7 Ablation Study

To validate the effectiveness of each module of our TGC, we eliminated the Node-level Contrast module (i.e. TGC/n) and the Context-level Contrast module (i.e. TGC/c) respectively. Also, we examines the impact of eliminating statistical features (i.e. TGC/s).

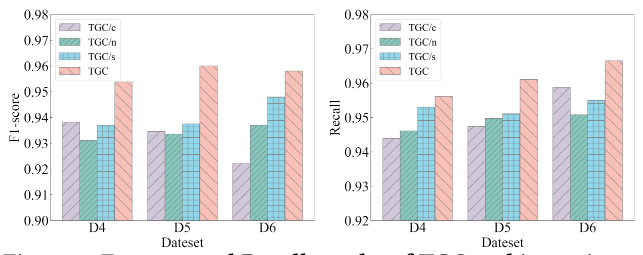


Figure 6: F1-score and Recall results of TGC and its variants.

From Figure 6, we can find that (1) our model performance gets worse by removing any key components, which reflects that all modules are important; (2) The performance of TGC/c drastically degrades, which is 1.56%, 2.55%, and 3.57% lower than TGC's F1-score on D_4, D_5, and D_6 datasets. This result indicates that learning transaction patterns in the transaction graph is essential for the phishing scams detection task, and also proves that the context structure of transaction nodes contains rich transaction pattern information; (3) The sharply decreased results in TGC/n is 2.64% lower than TGC on the D_5 dataset, which indicates the node-level contrast can capture the unique properties of the nodes and strengthen the nodes' representation ability; (4) After removing the statistical features, the F1-score of TGC on D4, D5, and D6 decreased by 1.68%, 2.25%, and 1.22% respectively, but the results still outperform the state-of-the-art methods.

4.8 Sensitivity Analysis

In this section, we perform sensitivity analysis on critical hyperparameters in TGC, i.e. the batch size that determine the negative sample selection and discrimination of contrastive learning, to show the stability of the model under perturbation of these hyperparameters. We only change the parameters in the sensitivity analysis, and other parameters remain the same as previously described.

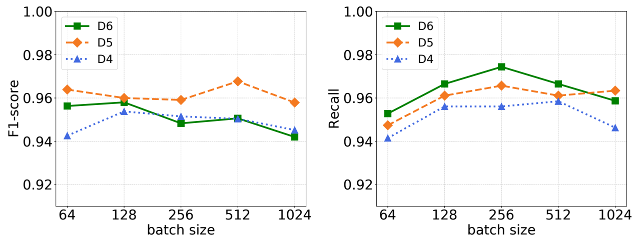


Figure 7: Sensitivity analysis of TGC with different batch size

The results on the D_4,D_5 and D_6 are shown in Figure 7. From Figure, under each batch size setting (1) TGC always achieves similar performance on all datasets, which is still the best performance compared with other methods in Table 2; (2) TGC can still achieve good performance when training with a relatively small batch size (e.g., batch size = 64), which demonstrates the strong capability of its infrastructure. But too small batch size is not conducive to the model learning discriminative features from too few negative samples. We thus conclude that, overall, our model is insensitive to the batch size, demonstrating the robustness to hyperparameter perturbation.

4.9 Discussion on Robustness and Extendibility

The robustness against evasion and extendibility is one of the important evaluation indicators of the model, which can reflect the sustainability and versatility of the method. In this section, we discuss TGC's robustness against evasion and extendibility to other blockchains.

Unlike traditional account registration, the process of creating an Ethereum account is private and does not require permission, thus having a low cost, and the same goes for other cryptocurrencies. In order to evade detection, some crimes prepare separate addresses for each victim to confuse the transaction analysis, such as Bitcoin ransomware attacks [19]. However, there is a big difference between Ethereum phishing scams and cryptocurrency attack scenarios where addresses are frequently switched. Phishing scams take high-reward propaganda to induce remittances, which usually sends phishing emails containing phishing addresses to multiple potential targets [25]. Therefore, phishing scams focus on masquerading as official or luring victims with high rewards. Usually, one phishing address corresponds to many victims, and the address will be changed only after a fixed start-up fund is collected. So our method TGC is robust against evasion.

As for the extendibility to other blockchains, The design of TGC is based on the analysis of the largest account-based blockchain, the Ethereum transaction network, and two Ethereum phishing address patterns proposed: the natural camouflage and sparsity of distribution. If other account-based blockchain phishing addresses involve these two patterns, TGC can be well adapted. Unlike account-based blockchains, transactions in a UTXO-based blockchain usually involve three or more addresses, so TGC cannot adapt to it such as Bitcoin, Litecoin, etc.

5 CONCLUSION

In this work, we propose a Transaction Graph Contrast Network (TGC) to enhance the performance of phishing scam detection on Ethereum. TGC inputs subgraphs instead of the entire graph for training, which eases the model's requirements for machine configuration and data connectivity, and can be well adapted to dynamic networks. Motivated by the natural camouflage and sparsity distribution of phishing addresses, we design node-level contrast and context-level contrast to learn the unique properties and universal transaction patterns of phishing addresses. We have conducted comprehensive experiments using various real-world datasets. Extensive experiments indicate that TGC's performance and practicality outperform state-of-the-art algorithms by a significant margin, and it can be better applied to large-scale dynamic networks.

ACKNOWLEDGMENTS

We sincerely appreciate Jiahao Yu of Northwestern University for writing assistance. We would also like to thank the anonymous reviewers for their constructive comments and input to improve our paper. This work is supported by The National Key Research and Development Program of China No.2020YFB1006100 and the Strategic Priority Research Program of Chinese Academy of Sciences, Grant No. XDC02030400.

REFERENCES

[1] Neda Abdelhamid, Aladdin Ayesh, and Fadi Thabtah. 2014. Phishing detection based associative classification data mining. *Expert Systems with Applications* 41, 13 (2014), 5948–5959.

[2] Israa Alqassem, Iyad Rahwan, and Davor Svetinovic. 2018. The anti-social system properties: Bitcoin network data analysis. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 50, 1 (2018), 21–31.

[3] Chainalysis. 2023. 2023-Crypto-Crime-Report. https://go.chainalysis.com/2023-crypto-crime-report.html. Accessed February, 2022.

[4] Liang Chen, Jiaying Peng, Yang Liu, Jintang Li, Fenfang Xie, and Zibin Zheng. 2020. Phishing scams detection in ethereum transaction network. *ACM Transactions on Internet Technology (TOIT)* 21, 1 (2020), 1–16.

[5] Tianqi Chen and Carlos Guestrin. 2016. XGBoost: A Scalable Tree Boosting System. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2016).

[6] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*. PMLR, 1597–1607.

[7] Weili Chen, Xiongfeng Guo, Zhiguang Chen, Zibin Zheng, and Yutong Lu. 2020. Phishing Scam Detection on Ethereum: Towards Financial Security for Blockchain Ecosystem.. In *IJCAI*. 4506–4512.

[8] Ching-Yao Chuang, Joshua Robinson, Yen-Chen Lin, Antonio Torralba, and Stefanie Jegelka. 2020. Debiased contrastive learning. *Advances in neural information processing systems* 33 (2020), 8765–8775.

[9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[10] Johannes Gasteiger, Stefan Weißenberger, and Stephan Günnemann. 2019. Diffusion improves graph learning. *Advances in neural information processing systems* 32 (2019).

[11] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. 855–864.

[12] Raia Hadsell, Sumit Chopra, and Yann LeCun. 2006. Dimensionality reduction by learning an invariant mapping. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, Vol. 2. IEEE, 1735–1742.

[13] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. *Advances in neural information processing systems* 30 (2017).

[14] William L Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*. 1025–1035.

[15] Shuichiro Haruta, Hiromu Asahina, and Iwao Sasase. 2017. Visual similarity-based phishing detection scheme using image and CSS with target website finder. In *GLOBECOM 2017-2017 IEEE Global Communications Conference*. IEEE, 1–6.

[16] Kaveh Hassani and Amir Hosein Khas Ahmadi. 2020. Contrastive Multi-View Representation Learning on Graphs. *ArXiv* abs/2006.05582 (2020).

[17] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. 2020. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 9729–9738.

[18] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross B. Girshick. 2019. Momentum Contrast for Unsupervised Visual Representation Learning. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2019), 9726–9735.

[19] Danny Yuxing Huang, Maxwell Matthaios Aliapoulios, Vector Guo Li, Luca Invernizzi, Elie Bursztein, Kylie McRoberts, Jonathan Levin, Kirill Levchenko, Alex C Snoeren, and Damon McCoy. 2018. Tracking ransomware end-to-end. In *2018 IEEE Symposium on Security and Privacy (SP)*. IEEE, 618–631.

[20] Mahmoud Khonji, Youssef Iraqi, and Andrew Jones. 2013. Phishing detection: a literature survey. *IEEE Communications Surveys & Tutorials* 15, 4 (2013), 2091–2121.

[21] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

[22] Thomas N Kipf and Max Welling. 2016. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308* (2016).

[23] Phuc H. Le-Khac, Graham Healy, and Alan F. Smeaton. 2020. Contrastive Representation Learning: A Framework and Review. *IEEE Access* 8 (2020), 193907–193934. https://doi.org/10.1109/ACCESS.2020.3031549

[24] Jure Leskovec and Christos Faloutsos. 2006. Sampling from large graphs. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. 631–636.

[25] Sijia Li, Gaopeng Gou, Chang Liu, Chengshang Hou, Zhenzhen Li, and Gang Xiong. 2022. TTAGN: Temporal Transaction Aggregation Graph Network for Ethereum Phishing Scams Detection. In *Proceedings of the ACM Web Conference 2022*. 661–669.

[26] Yukun Li, Zhenguo Yang, Xu Chen, Huaping Yuan, and Wenyin Liu. 2019. A stacking model using URL and HTML features for phishing webpage detection. *Future Generation Computer Systems* 94 (2019), 27–39.

[27] Dan Lin, Jiajing Wu, Qi Yuan, and Zibin Zheng. 2020. Modeling and understanding ethereum transaction records via a complex network approach. *IEEE Transactions on Circuits and Systems II: Express Briefs* 67, 11 (2020), 2737–2741.

[28] Xiao Liu, Fanjin Zhang, Zhenyu Hou, Li Mian, Zhaoyu Wang, Jing Zhang, and Jie Tang. 2021. Self-supervised learning: Generative or contrastive. *IEEE Transactions on Knowledge and Data Engineering* 35, 1 (2021), 857–876.

[29] Yixin Liu, Ming Jin, Shirui Pan, Chuan Zhou, Yu Zheng, Feng Xia, and Philip Yu. 2022. Graph self-supervised learning: A survey. *IEEE Transactions on Knowledge and Data Engineering* (2022).

[30] Jian Mao, Jingdong Bian, Wenqian Tian, Shishi Zhu, Tao Wei, Aili Li, and Zhenkai Liang. 2019. Phishing page detection via learning classifiers from page layout feature. *EURASIP Journal on Wireless Communications and Networking* 2019, 1 (2019), 1–14.

[31] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. 701–710.

[32] Jiezhong Qiu, Jian Tang, Hao Ma, Yuxiao Dong, Kuansan Wang, and Jie Tang. 2018. DeepInf: Social Influence Prediction with Deep Learning. *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (2018).

[33] Ozgur Koray Sahingoz, Ebubekir Buber, Onder Demir, and Banu Diri. 2019. Machine learning based phishing detection from URLs. *Expert Systems with Applications* 117 (2019), 345–357.

[34] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. Line: Large-scale information network embedding. In *Proceedings of the 24th international conference on world wide web*. 1067–1077.

[35] Yonglong Tian, Dilip Krishnan, and Phillip Isola. 2020. Contrastive multiview coding. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XI 16*. Springer, 776–794.

[36] Aäron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation Learning with Contrastive Predictive Coding. *ArXiv* abs/1807.03748 (2018).

[37] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*. 5998–6008.

[38] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903* (2017).

[39] Petar Velikovi, W. Fedus, William L Hamilton, P Liò, Y. Bengio, and R. D. Hjelm. 2018. Deep Graph Infomax.

[40] Daixin Wang, Peng Cui, and Wenwu Zhu. 2016. Structural deep network embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. 1225–1234.

[41] Jinhuan Wang, Pengtao Chen, Xinyao Xu, Jiajing Wu, Meng Shen, Qi Xuan, and Xiaoniu Yang. 2022. TSGN: Transaction Subgraph Networks Assisting Phishing Detection in Ethereum. *arXiv preprint arXiv:2208.12938* (2022).

[42] Xiao Wang and Guo-Jun Qi. 2022. Contrastive learning with stronger augmentations. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2022).

[43] Gavin Wood et al. 2014. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum project yellow paper* 151, 2014 (2014), 1–32.

[44] Jiajing Wu, Qi Yuan, Dan Lin, Wei You, Weili Chen, Chuan Chen, and Zibin Zheng. 2020. Who are the phishers? phishing scam detection on ethereum via network embedding. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* (2020).

[45] Zhirong Wu, Yuanjun Xiong, Stella X Yu, and Dahua Lin. 2018. Unsupervised feature learning via non-parametric instance discrimination. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 3733–3742.

[46] Qi Xuan, Jinhuan Wang, Minghao Zhao, Junkun Yuan, Chenbo Fu, Zhongyuan Ruan, and Guanrong Chen. 2019. Subgraph networks with application to structural feature space expansion. *IEEE Transactions on Knowledge and Data Engineering* 33, 6 (2019), 2776–2789.

[47] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems* 32 (2019).

[48] Yuning You, Tianlong Chen, Yongduo Sui, Ting Chen, Zhangyang Wang, and Yang Shen. 2020. Graph contrastive learning with augmentations. *Advances in neural information processing systems* 33 (2020), 5812–5823.

[49] Qi Yuan, Baoying Huang, Jie Zhang, Jiajing Wu, Haonan Zhang, and Xi Zhang. 2020. Detecting phishing scams on ethereum based on transaction records. In *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*. IEEE, 1–5.

[50] Chuxu Zhang, Dongjin Song, Chao Huang, Ananthram Swami, and N. Chawla. 2019. Heterogeneous Graph Neural Network. *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (2019).

A PHISHING ADDRESSES DATA ANALYSIS

A.1 Phishing Addresses Neighbor Statistics

First, we count the address categories and proportions around phishing addresses. Given a set of phishing addresses $V = \{v_1, v_2, \dots, v_n\}$, for address v_i, its r-neighbors are defined as $S_{v_i} = \{u : d(u, v_i) \leq r\}$ where $d(u, v_i)$ is the shortest path distance between u and v_i in its r-ego network. N denotes the length of the phishing address V set, while C_{v_i} is the length of S_{v_i}, which is the number of neighbors of phishing address v_i. The proportion of phishing addresses in the neighborhood can be calculated as

$$P_1 = \frac{1}{N} \sum_{i=1}^{N} p_i \tag{10}$$

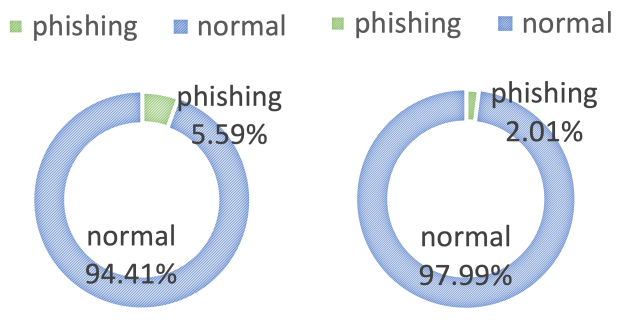
$$p_i = \frac{1}{C_{v_i}} \sum_{u \in S_{v_i}} \mathbf{1}_{[u \in V]} \tag{11}$$

When P_1 calculates the value at the local level, P_2 pays more attention to the whole

$$P_2 = \frac{\sum_{i=1}^{N} \sum_{u \in S_{v_i}} \mathbf{1}_{[u \in V]}}{\sum_{i=1}^{N} C_{v_i}} \tag{12}$$

We use the D_p's phishing addresses r-ego network as the data set for analysis which is described in detail in Section 5.1, and set $r = 2$. There are 5847 phishing r-ego networks in D_p.

According to the above formula, we get P_1, P_2 are 5.59% and 2.01% respectively, which means that the normal address occupies 94.41% or 97.99% of the neighbors of the phishing address.



(a) Calculated by Eq. 10 (b) Calculated by Eq. 12

Figure 8: Neighbor distribution of phishing addresses.

In conclusion, no matter which calculation method is used above, we can see that the normal addresses account for the vast majority of the neighbors of the phishing address, and in Section 4.1 we used the value calculated by Eq. 12 as the standard in our paper. Ethereum phishing address has natural camouflage, so existing methods which based on domain similarity assumptions cannot mine the unique properties of phishing addresses in this scenario.

A.2 Phishing Addresses Quantity Statistics

Next, we count the proportion of phishing addresses in all addresses. Given the number of phishing addresses as N_p, the number of total addresses as N_t, the overall proportion of phishing addresses P_3 can be presented as

$$P_3 = \frac{N_p}{N_t} \tag{13}$$

Dataset	Proportion of phishing addresses
D_1	0.353%
D_2	0.350%
D_3	0.332%
D_4	0.345%
D_5	0.340%
D_6	0.336%
D_p	0.061%

Table 7: The proportion of phishing addresses in seven datasets

We calculate the proportion of phishing addresses P_3 on the D_1, D_2, D_3, D_4, D_5, D_6 and D_p datasets described in Section 4.2, and the results are shown in the Table 7.

From Table 7 we can see that in either of the above seven datasets, the proportion of phishing addresses is very low. Moreover, we capture transactions centered on phishing addresses, which have increased the proportion of phishing addresses to a certain extent. This means that in real Ethereum transaction scenarios, the proportion of phishing addresses is even lower than the values we calculated.

In summary, the proportion of phishing addresses in the real Ethereum transaction network is very low. We use the median value 0.345% of D_1 to D_6 in Section 4.1 as the standard in our paper. Combined with the observation in Appendix A.1 that most of the neighbors of phishing addresses are normal addresses, we can conclude that the distribution of phishing addresses is very sparse. Existing methods cannot learn the similarity of the transaction patterns of phishing addresses.

B IMPLEMENTATION DETAILS

B.1 Computing Infrastructures

All models are implemented using Deep Graph Library 0.4.3, PyTorch 1.8.0, and NetworkX 2.5.1.

B.2 Hyperparameter Setting

For DeepWalk and Node2Vec, the walk length, window size, the latter's p and q are set to 20 and 4, 0.25, 0.4, respectively. For TSGN, we use Diffpool, which performs best in their papers, as their encoder. For the XGBoost model, the max depth and the learning rate are empirically fixed at 4 and 0.1, respectively. Due to the imbalance of the data, we upsample the minority class with a ratio of 8. For all the comparison methods, we set parameters based on their official implementations. We employ a two-layer GCN as the encoder for deep learning baselines like GraphSAGE due to its simplicity.

B.3 Baselines Encoder

We employ a two-layer GCN as the encoder for deep learning baselines like GraphSAGE and VGAE due to its simplicity. The

Creation time	Number of phishing addresses
2016	5
2017	340
2018	1928
2019	523
2020	1050
2021	218
2022	533
2023	31

Table 8: The creation time distribution of the phishing addresses

encoder architecture is formally given by

$$\mathrm{GC}_i(X, A) = \sigma\left(\hat{D}^{-\frac{1}{2}}\hat{A}\hat{D}^{-\frac{1}{2}}XW_i\right) \tag{14}$$

$$f(X, A) = \mathrm{GC}_2\left(\mathrm{GC}_1(X, A), A\right) \tag{15}$$

Where $\hat{A} = A + I$ is the adjacency matrix with self-loops, $\hat{D} = \sum_i \hat{A}_i$ is the degree matrix, $\sigma(\cdot)$ is a nonlinear activation function, e.g., $\mathrm{ReLU}(\cdot) = \max(0, \cdot)$, and W_i is a trainable weight matrix.

B.4 Phishing Addresses Creation Time Distribution

As shown in Table 8, we can see that the number of phishing nodes before January 2019 is roughly equal to the number of phishing nodes from January 2019 to March 2023,

we use the transactions before January 2019 to train the model to detect the emerging addresses from January 2019 to March 2023.

Gang Xiong1,2, Zhen Li1,2, Junchao Xiao1,2, Xinyu Xing3,,