

Evolve Path Tracer: Early Detection of Malicious Addresses in Cryptocurrency

Ling Cheng
Singapore Management University
Singapore

Feida Zhu
Singapore Management University
Singapore

Yong Wang
Singapore Management University
Singapore

Ruicheng Liang
Hefei University of Technology
Heifei, China

Huiwen Liu
Singapore Management University
Singapore

ABSTRACT

With the ever-increasing boom of Cryptocurrency, detecting fraudulent behaviors and associated malicious addresses draws significant research effort. However, most existing studies still rely on the full history features or full-fledged address transaction networks, thus cannot meet the requirements of early malicious address detection, which is urgent but seldom discussed by existing studies. To detect fraud behaviors of malicious addresses in the early stage, we present *Evolve Path Tracer* which consists of *Evolve Path Encoder LSTM*, *Evolve Path Graph GCN*, and *Hierarchical Survival Predictor*. Specifically, in addition to the general address features, we propose asset transfer paths and corresponding path graphs to characterize early transaction patterns. Further, since the transaction patterns are changing rapidly during the early stage, we propose *Evolve Path Encoder LSTM* and *Evolve Path Graph GCN* to encode asset transfer path and path graph under an evolving structure setting. *Hierarchical Survival Predictor* then predicts addresses' labels with nice scalability and faster prediction speed. We investigate the effectiveness and versatility of *Evolve Path Tracer* on three real-world illicit bitcoin datasets. Our experimental results demonstrate that *Evolve Path Tracer* outperforms the state-of-the-art methods. Extensive scalability experiments demonstrate the model's adaptivity under a dynamic prediction setting.

KEYWORDS

Early malice detection, Asset transfer path, Evolve encoder, Cryptocurrency, Bitcoin

1 INTRODUCTION

Cryptocurrency has rapidly grown into a decentralized global financial system in the past decade. Unfortunately, it has long been criticized for accommodating various cybercrime due to its anonymity. According to the recent Crypto Crime Report by chainalysis¹, malicious addresses' illegal profits exceeded 2.5 billion dollars. Among all cryptocurrency platforms, Bitcoin (BTC) has the largest volume, while the on-chain record data for a certain address are much scarcer than other popular platforms (e.g., ETH, EOS with smart contracts). Thus, researchers and practitioners have made significant efforts to fight against these fraudulent activities and identify the associated *malicious addresses* on BTC. Moreover, these methods are compatible with those on other cryptocurrency platforms.

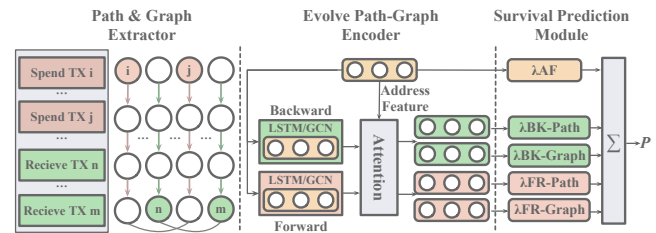


Figure 1: The framework of our Evolve Path Tracer. At each time step, the path extractor first extracts forward (FR: pink arrows) and backward (BK: green arrows) asset transfer paths. The graph extractor builds the path graph to link related asset transfer paths. Then, the Evolve Path-Graph Encoder encodes asset transfer paths with corresponding Path-LSTM models. The Path-GCN module refines the path representations with the Path-Graphs. The attention module will propose corresponding features with an attention summation. The weights of the Path-LSTM, Path-GCN, and attention modules are provided by address features. Finally, based on five types of features (Address Feature, BK/FR Path Feature, BK/FR Graph Feature), the survival probability is given by the corresponding survival rates λ . TX stands for a transaction, P stands for predictions.

Most existing detection methods focus on designing features to characterize specific types of malicious activity with detailed case studies. Besides, by combining statistic analysis and visualization technology, they successfully identified some representative malicious transaction patterns. Recent studies [3, 5, 30] further leverage deep-learning techniques to detect malicious addresses. By encoding account features and the transaction network structure with deep-learning models, they achieve great performance improvement in malicious address detection.

However, malicious activities are evolving faster than ever before, and it is impossible to build a unique feature set for every newly emerging malicious activity. Although some studies [13] can detect categories of malicious activities, they are still only available for post-hoc analysis. Most of them invariably require a full-history feature observation, which is consequentially scarce at the early stage of fraud activities, thus they can't be directly deployed to detect illicit activities at the early stage. Recently, Random walk and graph neural network [3, 7, 30, 31] are adopted to encode the topological feature of transaction network automatically, which improves the performance of general malice detection tasks. But

¹<https://go.chainalysis.com/2021-Crypto-Crime-Report.html>

most of them require a full-fledged address transaction network which is also unavailable under the early stage settings, as the newly created transaction graphs are usually small, unconnected and fast-evolving. Also, traditional address transaction networks may suffer from the issue of scalability, shadow addresses, and the dilution of the minority class.

Considering the essence of blockchain is a ledger of transactions (TXs), malicious addresses' objective is to transfer illicit money to a legal place. We can derive their intentions by monitoring their real-time transactions. Thus, in this work, we first set up the *Early Malicious Address Detection (EMAD)* task, which is urgent but seldom discussed by existing studies. Then, we develop a path extractor to focus on transaction paths, especially those *significant paths*, which can characterize the early-stage transactions of illicit addresses effectively. As shown in Figure 1, two kinds of asset transfer paths, i.e., forward and backward transition paths, are proposed to describe the flow-in and flow-out transition patterns. The asset transfer paths focus on the token (BTC) flow, thus can relieve the problem of shadow addresses.

Besides, illicit activities are usually organized together for specific purposes, and the behavior patterns evolve fast during the early stage. As a result, encoding each path individually may miss critical information. Furthermore, static encoding models can't capture the dynamism of evolving graphs. Thus, we build an asset path GCN module to encode the path's inter-relation. In this graph, asset transfer paths connect to each other if they share the same intersection addresses. Inspired by [23], we equip our path encoder and path GCN module with an evolving mechanism for more sophisticated path representations under the dynamic setting.

Considering the scalability issue, we implement a *Hierarchical Survival Prediction* module to alleviate the workload of feature preparation during the prediction. Previous prediction results can be directly used in the next time step, which empowers the model with a faster prediction speed and the ability to deal with a dynamic setting. In summary, the contributions of this paper can be summarized as follows:

- The Asset Transfer Paths are proposed for the *EMAD* task, which is urgent but seldom discussed. These paths exhibit high versatility in monitoring transaction patterns of various malicious behaviors in the early stage and relieve the problem of shadow addresses. This novel concept can be easily transplanted to all current blockchain-based cryptocurrencies and traditional financial systems.
- We propose the *Evolve Path Tracer* model that can fully utilize the asset transfer paths to encode various transaction patterns. Besides, it can also encode the paths' structural relationship under a dynamic setting with a novel evolve path graph encoding module. The versatility and dynamic flexibility are unachievable by other existing models.
- We conduct extensive evaluations to assess the model's effectiveness, and the results show that *Evolve Path Tracer* delivered a substantially better performance for three different illicit datasets than the state-of-the-art models. Also, owing to the *Hierarchical Survival Prediction* module, our *Evolve Path Tracer* can effectively predict addresses' labels scales well for the increasing data.

2 RELATED WORK

Early detection of malicious addresses on Bitcoin is an urgent yet seldom discussed task. Thus, we first review the related works about malice detection on Bitcoin and related cryptocurrencies. Then, we briefly review previous works about early rumor detection, which discussed similar tasks but under the social media circumstance.

2.1 Malice Detection on Cryptocurrency

Case Analysis mainly focuses on addresses' behaviors in a certain case. Reid et al. [24] identified entities by considering similar transaction times over an extended timeframe. Androulaki et al. [2] considered several features of transaction behavior, including the transaction time, the index of addresses, and the value of transactions. Jourdan et al. [10] explored five types of features, including address features, entity features, temporal features, centrality features, and motif features. Vasek et al. [29] gave a list of Bitcoin scams and conducted a statistical study. Case-Related features are often helpful in interpreting certain cases based on heuristic clustering and tainted fund flow. However, these methods require intensive case analysis, and most of the insights are only available in some specific cases, let alone apply to other platforms with complex heterogeneous relationships in general [12, 27].

Machine Learning can automatically learn general address features to address the poor generalization ability of case-related methods. Yin et al. [33] applied supervised learning to classify entities that might involve in cybercriminal activities. Akcora et al. [1] applied the topological data analysis (TDA) approach to generate the bitcoin address graph for ransomware payment address detection. Shao et al. [25] embedded the transaction history into a lower-dimensional feature for entity recognition. Nerurkar et al. [21] used 9 features to train the model for segregating 28 different licit-illicit categories of users. The general address features improve the model's generality significantly. However, they are challenging to characterize addresses' behaviors comprehensively. Particular transaction patterns of asset flow are difficult to be reflected in these characteristics.

Graph Based Methods focus on the interaction patterns between object addresses and related addresses. Harlev et al. [7] considered transaction features in a supervised machine learning framework to de-anonymize Bitcoin addresses and predict the type of yet-unidentified entities. Wu et al.[31] proposed two kinds of heterogeneous temporal motifs in the Bitcoin transaction network and applied them to detect mixing service addresses. Weber et al. [30] encodes address transaction graph with GCN, Skip-GCN, and Evolve-GCN. Chen et al.[3] proposed E-GCN for phishing node detection on the ETH platform. Tam et al.[28] proposed EdgeProp, a GCN-based model, to learn the representations of nodes and edges in large-scale transaction networks. Lin et al.[16] proposed two kinds of random walk-based embedding methods to encode specific network features. By changing the sampling strategy in Node2Vec, Wu et al.[32] proposed the Trans2Vec model, which can consider the temporal information. Li et al.[13] proposed TTAGN to model the temporal information of historical transactions for phishing detection. Chen et al.[4] proposed the AntiBenford subgraph framework for anomaly detection in the Ethereum transaction network under an unsupervised setting. Network-based methods perform

well for retrospect analysis, as they encode the structural information. However, in the early stages, the trading network is often too small to form a discriminative topological structure. The performance will degrade greatly if the networks are of sparse structures for the emerging networks with few connections[36]. Also, these methods may lead to Over-Smoothing issues and the dilution of the minority class [18] under the data-unbalanced setting [9, 35, 37].

2.2 Early Rumor Detection

Cho et al. [6] used GRU, a typical neural network for sequence modeling. At each time split, previous hidden state and current summation features are fed into the GRU unit to predict the labels for the given samples. Song et al. [26] proposed CED, which also uses GRU for sequence modeling. They proposes the concept of "Credible Detection Point," to increase the prediction speed. Yuan et al. [34] developed a multi-source long-short term memory network (M-LSTM) to model user behaviors by using a variety of user edit aspects as inputs. Zheng et al. [38] put forward SAFE. Instead of predicting the labels directly, it generates hazard rates for the survival models. The positive samples should die out fast, while the negative samples should stay alive.

3 PROBLEM DEFINITION

In the BTC system, a transaction tx is bound to an address. Each tx has a set of inputs $I = \{i_1, i_2, \dots, i_{|I|}\}$ and a set of outputs $J = \{j_1, j_2, \dots, j_{|J|}\}$. The input and output are still transactions. tx records token distribution between I and J . Narratively speaking, the incoming tokens flow into a pool and then flow to the outgoing transactions according to the prior agreement proportion. There is no record of how many tokens flow from an Input i to an Output j . Thus, we have to build a complete transaction bipartite graph for this tx and generate $|I| \times |J|$ transaction pairs in total. In other words, a BTC transaction has $|I| \times |J|$ transaction pairs inside. $D_{t_m} = \{d_{t_m}^i\}_{i=1}^N = \{(l^i, T_{in,t_m}^i, T_{out,t_m}^i)\}_{i=1}^N$ is the input data by the t_m -th time step, where $l^i \in \{0, 1\}$ is the label of Address i , and 0 or 1 stands for regular and malicious addresses respectively. $T_{in,t_m}^i = [tx_{in,1}^i, tx_{in,2}^i, \dots, tx_{N_{in,t_m}}^i]$ and T_{out,t_m}^i are the transaction sets where Address i acts as the input and output address by the t_m -th time step respectively. For ease of understanding, we denote these two transaction sets as *Receive* set and *Spend* set respectively.

Early Malicious Address Detection (EMAD). Given a set of addresses A , and D_{t_m} at timestep t_m , the problem is to build a binary classifier F such that

$$F(d_{t_m}^i) = \begin{cases} 1 & \text{if Address } i \text{ is malicious} \\ 0 & \text{Otherwise} \end{cases} \quad (1)$$

In the early detection task, to prevent the model predicts conflict labels at different timesteps, which will confuse users, thus we require the prediction to be consistent and predict the correct label as early as possible. We denote the confident time as t_c , where all classifier predictions F after t_c are consistent. The smallest t_c is denoted as $t_{f.c}$. Our purpose is to train a classifier that predicts the correct label of an address with a smaller $t_{f.c}$.

4 ASSET TRANSFER PATH

4.1 Overview and Motivation

Take Address i 's *Receive* transaction set as an example. For a *Receive* transaction j_0 in the set, suppose we find its significant asset source j_1 , which is also a transaction. Then we link j_1 and j_0 to form an asset transfer pair. After we trace the asset source iteratively, the asset transfer pairs form an asset transfer path naturally. As shown in Fig2 (b), for Receive Tx R-1 in the Flow of Receive Tx, after we trace its asset source, we can get three critical transfer pairs, namely $(10 \rightarrow R-1, 11 \rightarrow R-1, 12 \rightarrow R-1)$. As shown in *Backward Asset Transfer Path*, iteratively, we can get four paths (P-1, P-2, P-3, P-5) ended with R-1. Among them, three paths(P-1, P-2, P-3) have the same source (Tx-1 colored green). Also, another path (P-4) ended with R-2 is initiated by the same source, thus we group them into a path graph, which is colored green, as shown in the *Backward Path Graph*.

Since every transaction is bound to an address at a specific timestamp, thus, if all paths come from the same source, then we can say that, at a particular time point, an address initiates multiple transactions at the same time, and all transactions' destinations are Address i . If we encode each path as a node, these nodes can be connected through this source of funds, thus forming a graph. We build the path graph to reflect: 1) the characteristics of each path, 2) the interaction between paths, and 3) the characteristics of the asset source. We believe this information is crucial for the early detection of malicious behavior, and we will justify the statement in the experiments.

4.2 Influence and Trust Transaction Pair

Not all transaction pairs help identify illicit addresses. Those noteworthy pairs typically constitute a significant amount portion of the entire transaction. As illustrated in Fig. 2(a), the receive TX_j receive assets from three spend TXs (each contributing 35%, 30% and 35% to the total transaction amount). On the other hand, the spend TX_i transfers out BTCs to three receive TXs (with a distribution of 25%, 45%, and 30% as in this example). Given an Address i and a time step t_m , TXs in the shaded dotted-lined box represent all spend TXs, and receive TXs occurred up to timestep t_m associated with Address i .

As mentioned in Section 3, given a set $I = \{i_1, i_2, \dots, i_{|I|}\}$ of $|I|$ spend transactions to an receive transaction j and the set $\{I \rightarrow j\}$ of all transaction pairs, i.e., $\{I \rightarrow j\} = \{(i_1, j), (i_2, j), \dots, (i_{|I|}, j)\}$, we define *Influence Transaction Pair* as follows: Given an influence activation threshold θ , (i_k, j) is called an **Influence Transaction Pair** for transaction j , if there exists a k ($1 \leq k \leq |I|$) such that the amount of transaction pair (i_k, j) contributes at least a certain proportion of the received amount of transaction j , i.e. $\hat{A}(i_k, j) \geq \theta \times \hat{A}(\{I \rightarrow j\})$ where $\hat{A}(\cdot)$ denotes the amount of a transaction pair or the sum of all transaction pairs.

Similarly, given a set $J = \{j_1, j_2, \dots, j_{|J|}\}$ of $|J|$ transactions, and a transaction i , and the set $\{i \rightarrow J\}$ of all transaction pairs whose spend transaction is i , i.e., $\{i \rightarrow J\} = \{(i, j_1), (i, j_2), \dots, (i, j_{|J|})\}$. If there exists a receive transaction j_k , $1 \leq k \leq |J|$ such that transaction i transfers at least a certain proportion of its spend amount to it, this transaction pair is called a **Trust Transaction Pair** for

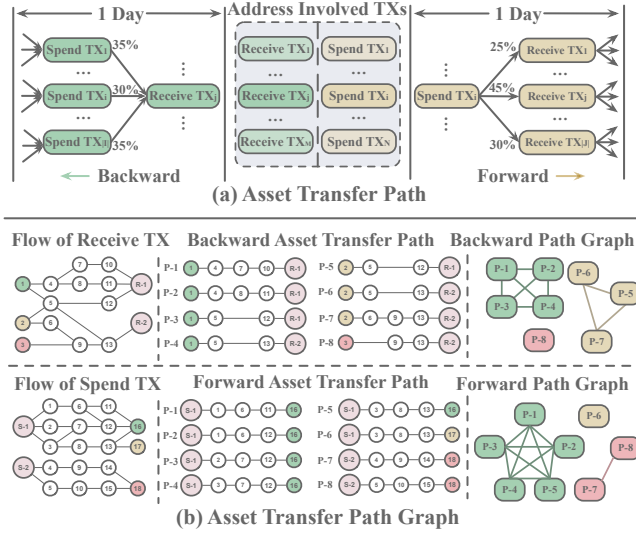


Figure 2: (a) Asset transfer pairs. Given an address, we collect its transaction history (All its Receive and Spend TXs in the dotted box). For each Receive TX, we trace its asset source from the inflow (Green Spend TXs) to build Influence TX pairs. For each Spend TX, we trace its asset destination from the outflow (Yellow Receive TXs) to build Trust TX pairs. $N = N_{in,t_m}$, $M = N_{out,t_m}$. (b) Asset transfer path and path graph. By tracing the asset source iteratively, we get a series of Influence TX pairs. We combine them end-to-end to form Backward Asset Transfer Path (Similar to Forward Asset Transfer Path). If paths have the same source or destination, we connect them through their intersection to form a path graph (Graph in the same color). Different colors stand for different starting or ending points.

transaction i , as it indicates a certain form of trust from i to j_k in terms of asset transfer.

Given an influence transaction pair (i_k, j) , we can conclude that transaction j obtains at least a significant amount (based on the threshold) of the asset in this transaction from transaction i_k . Accordingly, given a transaction j , if there exists a sequence of transaction pairs such that (I) each pair is an influence transaction pair; (II) the spend transaction of each pair is the receive transaction of the previous pair; and (III) the receive TX of the last pair is transaction j , we call such a sequence an **Backward Path** for j as indicated by the green arrow in Fig. 2(a). It reveals where j obtains the asset and can be used to trace back to the source of the asset. The detail to prepare the backward asset transfer path is shown in Appendix. Similarly, we can define a **Forward Path** to trace the destinations of transaction i 's asset flow. For brevity, we would refer to both the **Backward Path** and **Forward Path** as **Asset Transfer Paths**.

4.3 Asset Transfer Path Graph

The transaction graph between addresses has been widely used in the detection task of cryptocurrency. However, these methods may suffer from the perturbation of shadow addresses and the scalability issues caused by mixing services. If the malicious use

mixing services, although we will get more asset transfer paths, suspicious paths will still converge.

Therefore, unlike the previous address-based graph, we build graphs based on the asset transfer path. As shown in Fig. 2(b), in the path graph part, each node represents an asset-transfer path. If two paths share the same source (for backward paths) or destination (for forward paths), we then connect them with an edge, thus we can get a group of fully-connected graphs. Since every source or destination has a binding address at a specific timestep, we use the feature of this binding address at this time point to represent the edge feature in the corresponding graph. The address features and transaction features are illustrated in Appendix.

5 EVOLVE PATH TRACER

At the t -th timestep, We will generate five kinds of features to calculate corresponding hazard rates (λ) for prediction (1: Address Feature Hidden Vector (AF Hidden Vector), 2: Backward Path Feature (BK-Path), 3: Backward Graph Feature (BK-Graph), 4: Forward Path Feature (FR-Path), 5: Forward Graph Feature (FR-Graph)). Before generating the hazard rates, all these features will be encoded through the corresponding LSTM modules (T-1 to T-5 LSTM).

Besides, to capture the dynamics of path evolution, the parameters of forward and backward Evolve Path Encoder LSTM (E-1, E-2 LSTM) are provided by AF hidden vector. Also, forward and backward Evolve Path Graph GCN parameters are calculated with AF hidden vector. As shown in Fig 3, in the *Evolve Path Encoder LSTM* and *Evolve Path Graph GCN*, the parameters (the gray and white nodes) are consistent with the AF hidden vector to show their interactions.

5.1 Evolve Path Encoder LSTM

Address Feature is the basis for modeling and reasoning the address transaction pattern. We implement an address feature LSTM ($T-1$ LSTM), which will guide the processing in other modules.

$$h_t^T, c_t^T = \text{LSTM}^T(f_t^u, h_{t-1}^T, c_{t-1}^T), \quad (2)$$

where $h_t^T \in \mathbb{R}^d$ and $c_t^T \in \mathbb{R}^d$ are the hidden state and the cell state of $T-1$ LSTM at time t . $f_t^u \in \mathbb{R}^{d_n}$ is the address feature at time t . d is the dimension of hidden state, d_n is the dimension of address feature.

As mentioned in Section 4, asset transfer path is composed of a series of transaction nodes. The lengths of these paths are different. To encode them uniformly, we project an original path P_o to an uniform path P_u with length of L_u . Given an original path P_o with length of L_o , the zoom ratio is calculated by $R_z = L_o/L_u$, then the i -th (start from 0) node in P_u is calculated by the average feature of the $(\lfloor i \times R_z \rfloor)$ -th node to the $(\lceil (i+1) \times R_z \rceil - 1)$ -th node of P_o . Then, we denote the uniform path as:

$$P_{t:L_u} = [p_1, p_2, \dots, p_{L_u}], \quad (3)$$

Different addresses have different characteristics, their path structures also differ along the timeline. Thus we may lose dynamic information with a static encoder. Inspired by Evolve-GCN, the parameters of our Path Encoder LSTM are determined by the current address feature. In Evolve-GCN, the weights are updated by themselves or those representative nodes, thus may dismiss

the individual address property. Instead, we use the combination of address feature and the temporal path feature to generate the weights of Path Encoder LSTM module. Take backward asset transfer paths as example, for the j -th node in the input asset transfer path, backward *Evolve Path Encoder LSTM* ($E-1$ LSTM) computes the following function:

$$\begin{aligned}
H_t^p &= [h_t^{T_1} || h_{t-1}^{T_2}], \\
i_j &= \sigma((W_{ii}H_t^p)p_j + (W_{hi}H_t^p)h_{j-1}^{E_1} + b_iH_t^p), \\
f_j &= \sigma((W_{if}H_t^p)p_j + (W_{hf}H_t^p)h_{j-1}^{E_1} + b_fH_t^p), \\
g_j &= \tanh((W_{ig}H_t^p)p_j + (W_{hg}H_t^p)h_{j-1}^{E_1} + b_gH_t^p), \\
o_j &= \sigma((W_{io}H_t^p)p_j + (W_{ho}H_t^p)h_{j-1}^{E_1} + b_oH_t^p), \\
c_j &= f_j \odot c_{j-1} + i_j \odot g_j, \\
h_j^{E_1} &= o_j \odot \tanh(c_j),
\end{aligned} \quad (4)$$

where $||$ stands for concatenation, $h_{t-1}^{T_2} \in \mathbb{R}^d$ is the hidden state of $T-2$ LSTM at timestep $t-1$. $T-2$ LSTM encodes the temporal information of the backward asset transfer path set. $W_{**} \in \mathbb{R}^{d \times d \times 2d}$ and $b_* \in \mathbb{R}^{d \times 2d}$ are learnable weights that transfer H_t^p to the weights of projection layers and bias terms. σ stands for sigmoid function, \odot is the Hadamard product. Finally, each backward asset transfer path is denoted as final hidden state $h_{L_w}^{E_1}$ of $E-1$ LSTM. The representation of the i -th path at timestep t is $f_{i,t}^p$.

Not all paths are equally informative for the prediction. We expect to select more informative paths, thus we adopt multi-head attention for the selection.

$$a_{i,t}^j = W^{a,j} \tanh(W^{p,u} [f_{i,t}^p || h_t^{T_1}]), \quad (5)$$

$$\alpha_{i,t}^j = \text{Softmax}(a_{i,t}^j) = \exp(a_{i,t}^j) / \sum_{k=1}^{N_{E_1}} \exp(a_{k,t}^j), \quad (6)$$

$$\hat{f}_t^p = ||_{j=1}^{M^p} \hat{f}_t^{p,j}; \quad \hat{f}_t^{p,j} = \sum_{i=1}^{N_{E_1}} \alpha_{i,t}^j f_{i,t}^p, \quad (7)$$

where $W^{p,u} \in \mathbb{R}^{\frac{d}{M^p} \times 2d}$ and $W^{a,j} \in \mathbb{R}^{1 \times \frac{d}{M^p}}$ are learnable matrices, j stand for the index of the attention head, M^p is the total head number. N_{E_1} is the backward asset transfer path number. where $\hat{f}_t^{p,j}$ is the weighted summed path feature vector of j -th head, \hat{f}_t^p is the concatenation of all heads' output. The hidden state $h_t^{T_2}$ and cell state $c_t^{T_2}$ of $T-2$ LSTM are updated as:

$$h_t^{T_2}, c_t^{T_2} = \text{LSTM}^{T_2}(\hat{f}_t^p, h_{t-1}^{T_2}, c_{t-1}^{T_2}). \quad (8)$$

5.2 Evolve Path Graph GCN

If several paths are initiated by or converge at the same transaction, it may indicate certain suspicious patterns. By encoding the relationships between these path, model can capture certain significant patterns in detection. Similarly, due to the volatility of the path graph, we may lose the discriminative characteristics with a static model. To resolve this challenge, we propose *Evolve Path Graph GCN*. Take backward asset transfer paths as example, the nodes in

the path graph are updated as follow:

$$\begin{aligned}
H_t^g &= [h_t^{T_1} || h_{t-1}^{T_3}], \\
f_t^g &= \sigma(\tilde{D}^{-\frac{1}{2}} \tilde{\mathcal{A}} \tilde{D}^{-\frac{1}{2}} (f_t^p W^g H_t^g)), \\
\tilde{\mathcal{A}} &= \mathcal{A} + \mathcal{I}, \\
\mathcal{A}_{i,:j} &= (W^e H_t^g) S_{i,j}, \\
\tilde{D} &= \text{diag}(\sum_j (A_{i,j} + I_{i,j})),
\end{aligned} \quad (9)$$

where $h_{t-1}^{T_3} \in \mathbb{R}^d$ is the hidden state of $T-3$ LSTM at time $t-1$. $T-3$ LSTM encodes the temporal information of the backward path graph. $f_t^p \in \mathbb{R}^{N_{E_1} \times d}$ are the representations of path set at timestep t . $A \in \mathbb{R}^{N_{E_1} \times d \times N_{E_1}}$ and $I \in \mathbb{R}^{N_{E_1} \times d \times N_{E_1}}$ are the adjacent matrix and the identity matrix respectively. If the i -th path and j -th path have the same source, then $A_{i,:j} = \mathbf{1} \in \mathbb{R}^d$. Otherwise, $A_{i,:j} = \mathbf{0} \in \mathbb{R}^d$. If the i -th path and j -th path have the same source, $S_{i,j} \in \mathbb{R}^{d_n}$ is the intersection address feature of path i and path j . Otherwise, $S_{i,j} = \mathbf{0} \in \mathbb{R}^{d_n}$. $W^g \in \mathbb{R}^{d \times d \times 2d}$ and $W^e \in \mathbb{R}^{d \times d_n \times 2d}$ are learnable weights, and they project H_t^g to the weights of the corresponding projection layers. Thus $\mathcal{A} \in \mathbb{R}^{N_{E_1} \times d \times N_{E_1}}$.

We denote the output of *Evolve Path Graph GCN* encode as interaction-aware asset transfer paths. Resemble the previous calculation, we adopt multi-head attention to select significant signals for these interaction-aware paths. We denote the \hat{f}_t^g as the final result of multi-head attention of interaction-aware paths. The hidden state $h_t^{T_3}$ and cell state $c_t^{T_3}$ of $T-3$ LSTM are updated as:

$$h_t^{T_3}, c_t^{T_3} = \text{LSTM}^{T_3}(\hat{f}_t^g, h_{t-1}^{T_3}, c_{t-1}^{T_3}). \quad (10)$$

5.3 Hierarchical Survival Predictor

Due to the property of consistent prediction, survival analysis [38] is proved to be effective in the early detection task. The survival function $S(t)$ of an event represents the probability that this event has not occurred by time t . The hazard rate function λ_t is the event's instantaneous occurrence rate at time t given that the event does not occur before time t . In our case, the observation time is discrete in our case, we use t to denote a timestamp. The association between $S(t)$ and λ_t can be calculated as:

$$\begin{aligned}
S(t) &= P(T \geq t) = \prod_{k=t}^{\infty} f(x), \\
\lambda_t &= f(t)/S(t), \\
S(t) &= \exp(-\sum_{k=1}^t \lambda_k).
\end{aligned} \quad (11)$$

Considering the model's scalability during the real-time prediction, we define the event as "the address is benevolent" and we call hazard rate as benevolent rate. As the majority addresses are negative (benevolent) in the BTC platform. Once the address is classified as benevolent, we remove it from the monitoring list to reduce the computation cost.

To get more consistent predictions, previous work deployed a *Softplus* function $\lambda_t(x_t) = \ln(1 + \exp(x_t))$ to guarantee the hazard rate λ_t is always positive. Hence, the survival probability $S(t)$

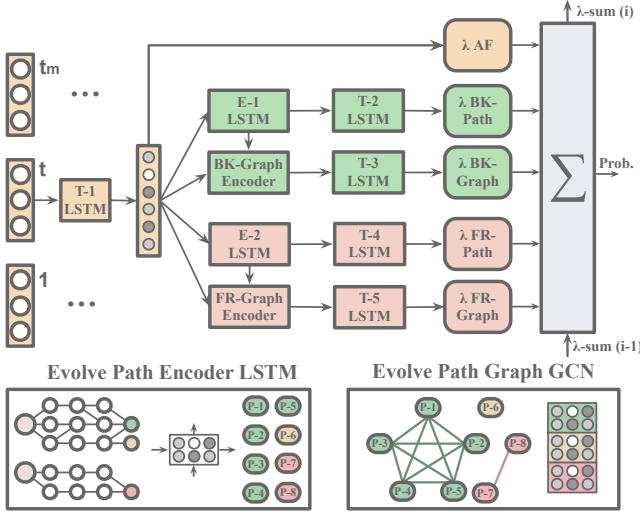


Figure 3: Detailed pipeline of *Evolve Path Tracer*. Five LSTM models (T-1 to T-5 LSTM) are implemented to encode temporal information of different features. AF hidden vectors will update the parameters of Forward and Backward Evolve Path Encoder LSTM (E-1/2 LSTM) and Evolve Path Graph GCN (BK/FR-Graph Encoder). Evolve Path Encoder LSTM and Evolve Path Graph GCN are proposed to encode asset transfer paths and path graphs dynamically. For detailed descriptions of each module, please refer to Appendix.

monotonically decreases. However, the model can hardly classify addresses correctly in the early hours. Those false-positive predictions will never be corrected with the monotonically decreasing survival probability. Thus, we release this restriction with a *tanh* activation function for benevolent rate calculation. The consistency is assured by *Consistency Loss Function* which will be elaborated later.

We designed five parallel benevolent rates corresponding to each kind of information (address feature, path feature (backward and forward), and graph feature (backward and forward)). At time step t , the calculation of these benevolent rates and the prediction as follows:

$$\begin{aligned} \lambda_{j,t} &= \tanh(W_{T_j}^{hz} h_t^{T_j}), \\ \hat{y}^t &= \exp(-\text{ReLU}(\sum_{i=1}^t \sum_{j=1}^5 \lambda_{j,i})), \end{aligned} \quad (12)$$

where $W_{T_j}^{hz} \in \mathbb{R}^{1 \times d}$ is the linear projection matrices for the output of T_j LSTM. At each time step, survival analysis first sums all previous benevolent rates, then it sums the current 5 benevolent rates hierarchically. Once addresses' current benevolent rates reach a certain threshold, we can remove them from monitoring list to speed up the prediction and relieve the computing cost in the following hours.

5.4 Training and Dynamical Prediction

Model Training Model should give higher $S(t)$ to malicious addresses and lower $S(t)$ to benevolent addresses in every time split.

For Address i , at timestep t_m , the early detection likelihood function and the negative logarithm prediction $loss^P$ are shown as below:

$$\begin{aligned} \text{likelihood} &= (1 - S(t_m))^{1-l_i} S(t_m)^{l_i} \\ &= (1 - \exp(-\sum_{t=1}^{t_m} \sum_{j=1}^5 \lambda_{j,t}))^{1-l_i} (\exp(-\sum_{t=1}^{t_m} \sum_{j=1}^5 \lambda_{j,t}))^{l_i}, \\ \text{loss}_{i,t_m}^P &= l_i \sum_{t=1}^{t_m} \sum_{j=1}^5 \lambda_{j,t} + (l_i - 1) \ln(1 - \exp(-\sum_{t=1}^{t_m} \sum_{j=1}^5 \lambda_{j,t})). \end{aligned} \quad (13)$$

Besides, $loss^P$ is weighted by $\sqrt{t_m}$ to avoid the perturbation in the early period due to the data insufficiency.

Consistency-boosted Loss Function Since the rate function is not guaranteed to be positive in our model, a consistency loss $loss^C$ is necessary for consistent predictions. In every time split, the benevolent rate should have the same sign as the previous time split.

$$\text{loss}_{i,t_m}^C = \begin{cases} 0 & \text{sign}(\lambda_{t_{m-1}} * \lambda_{t_m}) \geq 0 \\ 1 & \text{else} \end{cases}, \quad (14)$$

where $\lambda_{t_0} = 0$. Similarly, model should be able to rectify the poor prediction in the early period, thus the $loss^C$ is also weighted by $\sqrt{t_m}$.

Besides, since the numbers of positive and negative instances are imbalanced, different penalty coefficients are allocated to each class. Then, given a set of training samples with N_p malicious addresses and N_n legal addresses, the overall loss function is defined as:

$$\begin{aligned} \mathcal{L} &= \sum_{t=1}^{t_M} \sqrt{t} (C^+ \sum_{i=1}^{N_p} (\text{loss}_{i,t}^P + \gamma \text{loss}_{i,t}^C) + \\ & C^- \sum_{i=1}^{N_n} (\text{loss}_{i,t}^P + \gamma \text{loss}_{i,t}^C)), \end{aligned} \quad (15)$$

where C^+ and C^- are inversely proportional to the number of positive and negative instances in our settings. γ is a coefficient to control the contribution between $loss^P$ and $loss^C$.

Dynamical Prediction Besides the ‘‘Early Stop’’ mechanism provided by *Hierarchical Survival Predictor*, our dynamical construction scheme of asset transfer paths can also relieve the time cost of feature preparation. As shown in Fig. 4, the path data can be reused if no new transaction occurs in this interval. If an address has new *Receive* or *Spend* transactions, model will create new backward or forward asset transfer paths accordingly. Moreover, model also check the endpoint of the forward paths to determine whether they need to be extended or not.

6 EXPERIMENT AND ANALYSIS

In this section, we perform empirical evaluation to answer the following research questions:

- **RQ1:** What is the performance with respect to different uniform path lengths?
- **RQ2:** Does Evolve Path Tracer outperform the state-of-the-art methods for early malicious address detection?
- **RQ3:** How dose each components benefit the final detection performance?

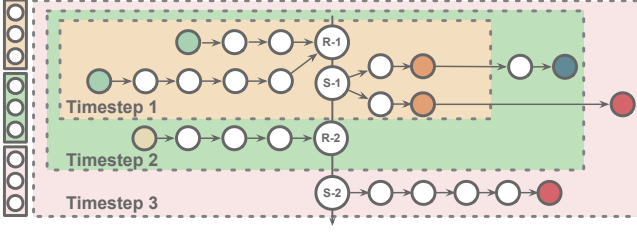


Figure 4: Dynamical Construction of asset transfer path. The three vectors on the left are Address Features corresponding to Timestamp 1 to Timestamp 3. Different dash boxes represent input information at different timestamps.

Table 1: Dataset Statistics

Type	Definition	Posi.	Nega.	P/N(%)
H	Hack and steal tokens	302	6582	4.03
R	Encrypt data for ransoms	3224	21100	15.28
D	Illegal BTC darknets	5838	109937	5.31

- **RQ4:** Dose the time overhead of the preparation procedure and scalability satisfy the real-time requirement?

6.1 Data Collection and Preparation

The transaction data are publicly accessible by running a Bitcoin client. We obtained all the data from the 1-st to the 700,000-th block for higher credibility, as we only collect addresses verified by enough participants. For a given address, we get the related transaction history based on the APIs exposed by BlockSci [11]. Based on this transaction history, we can calculate the related features and prepare the asset transfer paths and path graphs. To get the labels for three different illicit activities (Hack, Ransomware, and Darknet), we performed a manual search on public forums and datasets, such as Bitcointalk forum², Reddit, WalletExplorer³ and several prior studies [14, 22, 31] to fetch related labels. Negative (Regular) addresses are collected in the same method as [17, 31]. We set the activation threshold as 0.01 to prepare the asset transfer path. One can set a smaller threshold depending on the operating device. More detail about the statistical properties of the asset transfer path can be found in Appendix A. Table. 1 shows the summarized descriptions: The definition and numbers of positive (Posi), negative (Nega), and Positive/Negative ratio (P/N) for each malicious type (H: Hack, R: Ransomware, D: Darknet).

6.2 Settings and Metrics

As our purpose is to detect malicious addresses as early as possible, the model should detect them before the institution’s daily settlement when the institutions may find the malice by themselves. Therefore, our experiments focus on early illicit detection during the first day. Although the experiments investigate the performance during the first day, our *Evolve path Tracer* can work with an arbitrary timespan. To evaluate the performance of our model, we get

²<https://bitcointalk.org/>

³<https://www.walletexplorer.com>

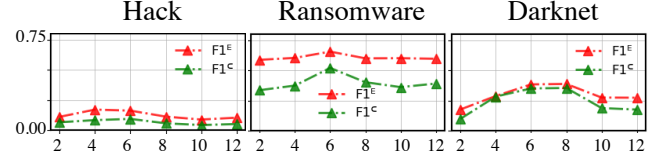


Figure 5: $F1^E$ and $F1^C$ of different uniform path lengths on three datasets.

24 hours data with 1 hour interval, and we average the evaluation metrics on all timesteps.

The selected metrics are accuracy (Acc.), precision (Prec.), and recall (Rec.). Besides, the model should predict correct labels fast to prevent economic loss earlier. Also, due to data insufficiency, the model may predict conflict labels at different timesteps, thus confusing users. Thus we require the predictions to be consistent. we introduce the early-weighted F1 score $F1^E$ and consistency-weighted score $F1^C$ as follows:

$$F1^E = \frac{\sum_{i=1}^N F1_i / \sqrt{i}}{\sum_{i=1}^N 1 / \sqrt{i}}, \quad (16)$$

$$F1^C = \frac{\sum_{i=1}^{N-1} \sqrt{i} \times F1_i \times \mathbb{1}_{y_c}(y_i)}{\sum_{i=1}^{N-1} \sqrt{i}},$$

where i is the timestep, y_c is the set of predictions where $\text{sign}((y_i - 0.5) \times (y_{i+1} - 0.5)) > 0$. The indicator function $\mathbb{1}_{y_c}(y_i) = 1$ when $y_i \in y_c$. $F1_i$ is the F1 score of the prediction at timestep i .

6.3 Effects of Uniform Path Length (RQ1)

As mentioned in Section 5.1, to encode the asset transfer paths, we need to project asset transfer paths to the same length L_u . We further analyze the effects of uniform path length with a simplified **AF/Path** model (using Address features and Asset Transfer Path features). We test 6 groups with different path lengths (From 2 to 12 by the interval of 2) on all three datasets. As we use a simplified model, we focus on the path length that maximizes the performance, as shown in Figure 5.

A Uniform path with a longer length can preserve more information that contributes to better performance. Thus the model performs better as L_u increases at the beginning. However, if the L_u is longer than most asset transfer paths, the uniform path may introduce more redundant noise. Therefore, the model does not perform better when the path length is too large.

Since hack addresses get funds directly from the victim’s account and need to transfer money as soon as possible, its asset transfer path is shorter. As shown in Fig 5, the model achieves the best $F1^E$ and $F1^C$ scores when setting L_u to 4 and 6, respectively. Considering both scores, the model performs best when L_u equals 6. Ransomware is malicious software that threatens the victims to pay a ransom fee. In many cases, the ransom demand comes with a deadline. Victims buy bitcoins from exchanges and transfer them to criminals, thus slightly increasing the lengths of the asset transfer paths. As shown in Fig 5, the model performs best When the L_u is 6. A darknet is an overlay network within the Internet that can only be accessed with specific authorization. Thereby, users could buy and sell illicit goods anonymously via the darknet. Since platforms need to wait for the activity of buyers and sellers, there

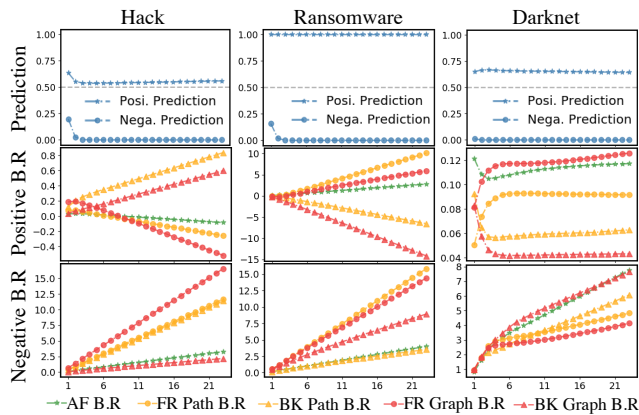


Figure 6: Prediction evolution of different address groups and corresponding average Benevolent Rates (B.R) of different features.

will be a longer asset transfer path. The model performs best When the L_u is 8. However, setting L_u to 6, the model gets similar scores. Considering the model’s scalability, in the actual experiment, we also set L_u to 6.

6.4 Performance Comparison (RQ2)

To verify the effectiveness and versatility of our *Evolve Path Tracer*, we first compare the most common machine learning models, then compare our encoder module with the encoder in other early detection models. At last, we also compare the address graph-based models. The models are detailed in the appendix. The main results for comparing all different methods are shown in Table 2, and the major findings are summarized as follows:

(1) In terms of the five evaluation metrics, our *Evolve Path Tracer* outperforms most compared methods by a significant margin. Especially for early detection performance metrics $F1^E$ and $F1^C$, our *Evolve Path Tracer* achieves the best performance under all three datasets. Compared to the second-best methods, *Evolve Path Tracer* has an average increase of 14.54% on $F1^E$ and an average increase of 15.63% on $F1^C$. Besides, none of these methods can perform well on all three datasets. These significant performance margins justify the effectiveness and versatility of our *Evolve Path Tracer*. Besides, the "Early Stop" mechanism accelerates the prediction speed and helps the model discard subsequent noise, improving the model’s performance.

(2) Traditional machine learning algorithms do not perform well on the three datasets because these algorithms are difficult to encode temporal information. It is difficult for decision-tree-based machine learning algorithms to consider shifts in the feature decision boundary along the times [19, 20]. Therefore, our model has an average improvement of 80.52% on $F1^E$ and 83.44% on $F1^C$ compared to the best decision-tree-based model. The sequential deep learning methods perform well on the three datasets. However, our *Evolve Path Tracer* still has an average 21.82% improvement on $F1^E$ and 24.11% on $F1^C$ compared to the best model in this group. The first reason is that the inter-relationships among asset transfer paths can reveal specific transaction patterns (e.g., two addresses transfer money through multiple paths to avoid monitoring). In

addition, since the transaction pattern evolves in the early stage, a static encoding module can hardly encode evolving information.

(3) Compared with Address Graph methods, our *Evolve Path Tracer* has an average increase of 22.74% on $F1^E$ and an average increase of 23.36% on $F1^C$. Among them, Evolve-GCN performs the best in most datasets, which verifies the fast-evolving of early transaction networks. However, as [18] implies, the address GCN may lead to Over-Smoothing issues and the dilution of the minority class. In our cases, most neighbors of malicious nodes are victims or shadow addresses. Thus the Address Graph models do not perform well. To avoid the dilution problem, in *Evolve Path Tracer*, we set vertices as transactions to utilize the relevant address information in a "safer" way.

6.5 Ablation Study (RQ3)

As shown in Table 3, AF performed poorly on the three datasets. Because many benevolent addresses (change address, ICO, and legal market addresses) behave similarly to these malicious addresses. As shown in Fig. 6, the AF benevolent rates for most negative samples do not exceed 2.5. The introduction of asset transfer path features significantly improves the performance. As shown in Fig. 6, for the Hack addresses, the forward transaction signal is more important than the backward one because the Hack address will transfer the funds faster and more centralized. Ransomware and Darknet addresses usually require victims or buyers to transfer funds according to certain conditions. Thus the backward information is more valuable.

Comparing **+Path** and **+Graph**, by encoding the paths’ interrelationships, the model gives predictions based on transaction patterns rather than the fluctuation of a single path. As shown in Fig. 6, the prominent signals of malicious nodes are enhanced by introducing path graphs. In the cryptocurrency transaction network, Therefore, the model should be able to handle the differences between various types of addresses at different times. By comparing the performance differences between **+Graph** and **+Evolve**, we found that this Evolve mechanism is necessary. **+Graph** only performs well if the address has a shorter life span, and these addresses will be discarded after the first few transactions. However, for other addresses with longer lifetimes, **+Evolve** can better reflect changes in the transaction patterns of these addresses, resulting in better performance.

6.6 Scalability and Dynamical Prediction (RQ4)

Feature Preparation Time Cost When a new block appears, real users will generally monitor the addresses that have transactions with them. Those new and large-volume addresses are likely to participate in dangerous activities. Thus, we randomly selected 1,000 blocks (from the first block of 2018 to the first block of 2022) and collected the daily BTC price during this period. We filter out transactions lower than \$10,000 and retrieve address with a lifespan less than one week. We prepare every address’s data of the first 24 hours, the time cost is illustrated as follows: During each interval (1 hour is about 6 blocks), we need to monitor about 1,166 new addresses, which will only cost 5 minutes. Moreover, as shown in Table 4, our time cost resembles address graph preparation, but we can collect information much further than 2 hops.

Table 2: Scores of different prediction model. Evo-PT and Evo-PT (E) are our *Evolve Path Tracer* with/wo “Early Stop” mechanism. Underline stands for best score in the group, Bold stands for best score in all groups.

Type	Model Name	Hack					Ransomware					Darknet				
		Acc.	Prec.	Rec.	F1 ^E	F1 ^C	Acc.	Prec.	Rec.	F1 ^E	F1 ^C	Acc.	Prec.	Rec.	F1 ^E	F1 ^C
Machine Learning	DT	0.995	0.347	0.137	0.197	0.197	0.955	0.736	0.432	0.545	0.545	0.982	0.448	0.152	0.227	0.227
	RF	0.996	<u>0.405</u>	<u>0.242</u>	<u>0.303</u>	<u>0.303</u>	0.955	0.735	<u>0.436</u>	0.547	0.547	0.983	0.519	0.109	0.181	0.181
	XGB	0.997	0.347	0.137	0.197	0.197	0.960	0.865	0.435	<u>0.579</u>	<u>0.579</u>	0.985	0.790	<u>0.191</u>	<u>0.308</u>	<u>0.308</u>
Sequen. Deep Learning	GRU	0.928	0.298	0.438	0.354	0.354	0.885	0.558	0.949	0.703	0.703	0.942	0.470	0.838	0.603	0.603
	M-LSTM	<u>0.949</u>	<u>0.418</u>	0.272	0.328	0.333	0.887	0.561	0.969	0.711	0.710	<u>0.951</u>	<u>0.520</u>	0.845	<u>0.642</u>	<u>0.645</u>
	CED	0.909	0.265	<u>0.563</u>	<u>0.360</u>	<u>0.360</u>	<u>0.909</u>	<u>0.617</u>	0.960	<u>0.752</u>	0.751	0.943	0.478	0.829	0.606	0.606
	SAFE	0.918	0.285	0.438	0.271	0.330	<u>0.909</u>	0.616	0.963	<u>0.752</u>	<u>0.752</u>	0.949	0.508	0.838	0.632	0.632
Addr. Graph	GCN	<u>0.920</u>	<u>0.433</u>	0.670	<u>0.501</u>	<u>0.507</u>	0.887	0.564	0.936	0.700	0.706	<u>0.942</u>	<u>0.459</u>	0.613	0.525	0.524
	Skip-GCN	0.917	0.410	0.690	0.443	0.432	0.903	0.603	0.935	0.729	0.735	0.941	<u>0.459</u>	0.629	<u>0.531</u>	0.530
	Evo-GCN	0.893	0.428	0.749	0.427	0.442	0.906	<u>0.613</u>	0.944	<u>0.736</u>	<u>0.746</u>	0.941	0.459	0.633	0.530	0.533
TX. Graph	Evo-PT	0.963	0.607	<u>0.739</u>	0.664	0.668	0.938	0.743	<u>0.869</u>	0.799	0.802	0.963	0.624	<u>0.764</u>	0.686	0.686
	Evo-PT (E)	0.969	0.650	0.731	0.689	0.683	0.940	<u>0.751</u>	<u>0.869</u>	0.802	0.807	<u>0.964</u>	0.625	0.754	0.686	0.687

Table 3: Scores of different ablation models on Hack (H), Ransomware (R), and Darknet (D). Ablation modules include Address Features (AF), Path features (+Path), Path Graph features (+Graph), and Evolve schemes (+Evolve).

	Model	Acc.	Prec.	Rec.	F1 ^E	F1 ^C
H	AF	0.920	0.309	0.590	0.389	0.412
	+Path	0.954	0.537	0.546	0.509	0.538
	+Graph	0.965	0.686	0.476	0.545	0.559
	+Evolve	0.961	0.606	0.553	0.551	0.576
R	AF	0.911	0.710	0.632	0.626	0.628
	+Path	0.929	0.727	0.805	0.760	0.765
	+Graph	0.927	0.696	0.875	0.773	0.776
	+Evolve	0.937	0.735	0.871	0.795	0.798
D	AF	0.961	0.619	0.571	0.611	0.604
	+Path	0.961	0.611	0.693	0.649	0.650
	+Graph	0.960	0.586	0.804	0.678	0.678
	+Evolve	0.963	0.626	0.758	0.685	0.685

Table 4: Time cost of different input data, includes Block Number, Transaction Number, Address Number, Address Feature, Asset Transfer Path, Path Graph, and Address Graph.

#Blk	#TX	#Addr	A-Feat	Path	P-G	A-G
1,000	306,258	194,310	175s	7.7h	6.9h	14.3h
Avg.	306	194	0.2s	27.6s	24.8s	51.4s

Scalability of Early Stop To justify the Scalability of our “Early Stop” mechanism, we plot the skip ratios with a different threshold. As shown in Fig. 7, all models can filter out most (80%) addresses by the fourth hour. The mechanism improves the model’s Scalability significantly. Moreover, choosing a reasonable threshold helps the

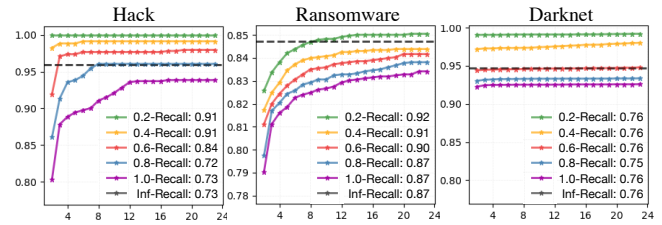


Figure 7: Skip Ratio evolution and Recall scores with different thresholds. The gray line is the Positive/Negative ratio of each dataset.

model to discard subsequent noise and improve the model’s performance, as mentioned in Section 6.4. A lower threshold means a faster prediction speed.

However, there is a concern about missing malicious addresses as we decrease the threshold for faster prediction speed. Which then decreases the model’s Recall scores. As shown in Fig, compared to the model without “Early Stop” (labeled as “Inf”), the model has better Recall scores as we decrease the threshold. This is because our model can predict the most benevolent addresses in the early hours. Removing them from the monitoring list can avoid subsequent noise, which improves the model’s Recall scores. Thus our *Evolve Path Tracer* has a faster prediction speed without missing malicious addresses.

7 CONCLUSION AND FUTURE WORK

In this paper, we present *Evolve Path Tracer*, a novel framework for early malicious address detection on BTC. We first propose asset transfer paths and encode them with *Evolve Path Encoder LSTM*. The asset transfer paths exhibit high versatility in monitoring transaction patterns of various malicious behaviors in the early stage. To take full advantage of these paths, the *Evolve Path Graph GCN* is built to encode corresponding path graphs. The graphs capture the interrelation among the paths. In particular, all modules are evolving along with the timeline to encode the dynamics of paths’ content and inter-relation. Finally, we implement *Hierarchical Survival Predictor* with *Consistency Loss Function* to achieve better

prediction performance, higher consistency, and excellent scalability. The model is quantitatively and qualitatively evaluated on three malicious address datasets. Extensive ablation studies elaborate on the mechanisms behind the effectiveness and excellent scalability. In future work, we would like to extend *Evolve Path Tracer* to malicious address detection in other crypto-currency platforms and traditional financial domains.

REFERENCES

- [1] Cuneyt G Akcora, Yitao Li, Yulia R Gel, and Murat Kantarcioglu. 2020. Bit-coinheist: Topological data analysis for ransomware prediction on the bitcoin blockchain. In *Proceedings of the twenty-ninth international joint conference on artificial intelligence*.
- [2] Elli Androulaki, Ghassan O Karame, Marc Roeschlin, Tobias Scherer, and Srdjan Capkun. 2013. Evaluating user privacy in bitcoin. In *International conference on financial cryptography and data security*. Springer, 34–51.
- [3] 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.
- [4] Tianyi Chen and Charalampos Tsourakakis. 2022. Antibenford subgraphs: Un-supervised anomaly detection in financial networks. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 2762–2770.
- [5] Weili Chen, Zibin Zheng, Jiahui Cui, Edith Ngai, Peilin Zheng, and Yuren Zhou. 2018. Detecting ponzi schemes on ethereum: Towards healthier blockchain technology. In *Proceedings of the 2018 world wide web conference*. 1409–1418.
- [6] Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the Properties of Neural Machine Translation: Encoder–Decoder Approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*. 103–111.
- [7] Mikkel Alexander Harlev, Haohua Sun Yin, Klaus Christian Langenheldt, Raghava Mukkamala, and Ravi Vatrappu. 2018. Breaking bad: De-anonymising entity types on the bitcoin blockchain using supervised machine learning. In *Proceedings of the 51st Hawaii International Conference on System Sciences*.
- [8] Mikkel Alexander Harlev, Haohua Sun Yin, Klaus Christian Langenheldt, Raghava Mukkamala, and Ravi Vatrappu. 2018. Breaking bad: De-anonymising entity types on the bitcoin blockchain using supervised machine learning. In *Proceedings of the 51st Hawaii international conference on system sciences*.
- [9] Shuli Jiang, Robson Leonardo Ferreira Cordeiro, and Leman Akoglu. 2022. D. MCA: Outlier Detection with Explicit Micro-Cluster Assignments. *arXiv preprint arXiv:2210.08212* (2022).
- [10] Marc Jourdan, Sebastien Blandin, Laura Wynter, and Pralhad Deshpande. 2018. Characterizing entities in the bitcoin blockchain. In *2018 IEEE international conference on data mining workshops (ICDMW)*. IEEE, 55–62.
- [11] Harry Kalodner, Malte Möser, Kevin Lee, Steven Goldfeder, Martin Plattner, Alishah Chator, and Arvind Narayanan. 2020. Blocksci: Design and applications of a blockchain analysis platform. In *29th {USENIX} Security Symposium ({USENIX} Security 20)*. 2721–2738.
- [12] Jonathan Kuck, Honglei Zhuang, Xifeng Yan, Hasan Cam, and Jiawei Han. 2015. Query-based outlier detection in heterogeneous information networks. In *Advances in database technology: proceedings. International Conference on Extending Database Technology*, Vol. 2015. NIH Public Access, 325.
- [13] 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.
- [14] Yang Li, Yue Cai, Hao Tian, Gengsheng Xue, and Zibin Zheng. 2020. Identifying illicit addresses in bitcoin network. In *International Conference on Blockchain and Trustworthy Systems*. Springer, 99–111.
- [15] Jiaqi Liang, Linjing Li, Daniel Zeng, Shu Luan, and Lu Gan. 2019. Bitcoin exchange addresses identification and its application in online drug trading regulation. (2019).
- [16] Dan Lin, Jiajing Wu, Qi Yuan, and Zibin Zheng. 2020. T-edge: Temporal weighted multidigraph embedding for ethereum transaction network analysis. *Frontiers in Physics* 8 (2020), 204.
- [17] Bing Liu, Yang Dai, Xiaoli Li, Wee Sun Lee, and Philip S Yu. 2003. Building text classifiers using positive and unlabeled examples. In *Third IEEE International Conference on Data Mining*. IEEE, 179–186.
- [18] Yang Liu, Xiang Ao, Zidi Qin, Jianfeng Chi, Jinghua Feng, Hao Yang, and Qing He. 2021. Pick and Choose: A GNN-Based Imbalanced Learning Approach for Fraud Detection. In *Proceedings of the Web Conference 2021*.
- [19] Mohammad Masud, Jing Gao, Latifur Khan, Jiawei Han, and Bhavani M. Thuraisingham. 2011. Classification and Novel Class Detection in Concept-Drifting Data Streams under Time Constraints. *IEEE Transactions on Knowledge and Data Engineering* 23, 6 (2011), 859–874. <https://doi.org/10.1109/TKDE.2010.61>
- [20] Mohammad M. Masud, Qing Chen, Latifur Khan, Charu C. Aggarwal, Jing Gao, Jiawei Han, Ashok Srivastava, and Nikunj C. Oza. 2013. Classification and Adaptive Novel Class Detection of Feature-Evolving Data Streams. *IEEE Transactions on Knowledge and Data Engineering* 25, 7 (2013), 1484–1497. <https://doi.org/10.1109/TKDE.2012.109>
- [21] Pranav Nerurkar, Yann Busnel, Romaric Ludinard, Kunjal Shah, Sunil Bhirud, and Dhiren Patel. 2020. Detecting illicit entities in bitcoin using supervised learning of ensemble decision trees. In *Proceedings of the 2020 10th international conference on information communication and management*. 25–30.
- [22] Masarah Paquet-Clouston, Bernhard Haslhofer, and Benoit Dupont. 2019. Ransomware payments in the bitcoin ecosystem. *Journal of Cybersecurity* 5, 1 (2019), tz003.
- [23] Aldo Pareja, Giacomo Domeniconi, Jie Chen, Tengfei Ma, Toyotaro Suzumura, Hiroki Kanezashi, Tim Kaler, Tao Scharidl, and Charles Leiserson. 2020. Evolvegcn: Evolving graph convolutional networks for dynamic graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 5363–5370.
- [24] Fergal Reid and Martin Harrigan. 2013. An analysis of anonymity in the bitcoin system. In *Security and privacy in social networks*. Springer, 197–223.
- [25] Wei Shao, Hang Li, Mengqi Chen, Chunfu Jia, Chunbo Liu, and Zhi Wang. 2018. Identifying bitcoin users using deep neural network. In *International Conference on Algorithms and Architectures for Parallel Processing*. Springer, 178–192.
- [26] Changhe Song, Cheng Yang, Huimin Chen, Cunchao Tu, Zhiyuan Liu, and Maosong Sun. 2019. CED: Credible early detection of social media rumors. *IEEE Transactions on Knowledge and Data Engineering* (2019).
- [27] Yizhou Sun, Jiawei Han, Charu C. Aggarwal, and Nitesh V. Chawla. 2012. When Will It Happen? Relationship Prediction in Heterogeneous Information Networks. In *Proceedings of the Fifth ACM International Conference on Web Search and Data Mining (WSDM '12)*. 663–672. <https://doi.org/10.1145/2124295.2124373>
- [28] Da Sun Handason Tam, Wing Cheong Lau, Bin Hu, Qiu Fang Ying, Dah Ming Chiu, and Hong Liu. 2019. Identifying Illicit Accounts in Large Scale E-payment Networks—A Graph Representation Learning Approach. *arXiv preprint arXiv:1906.05546* (2019).
- [29] Marie Vasek and Tyler Moore. 2015. There’s no free lunch, even using Bitcoin: Tracking the popularity and profits of virtual currency scams. In *International conference on financial cryptography and data security*. Springer, 44–61.
- [30] Mark Weber, Giacomo Domeniconi, Jie Chen, Daniel Karl I Weidele, Claudio Bellei, Tom Robinson, and Charles E Leiserson. 2019. Anti-money laundering in bitcoin: Experimenting with graph convolutional networks for financial forensics. *arXiv preprint arXiv:1908.02591* (2019).
- [31] Jiajing Wu, Jieli Liu, Weili Chen, Huawei Huang, Zibin Zheng, and Yan Zhang. 2021. Detecting mixing services via mining bitcoin transaction network with hybrid motifs. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* (2021).
- [32] 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).
- [33] Haohua Sun Yin and Ravi Vatrappu. 2017. A first estimation of the proportion of cybercriminal entities in the bitcoin ecosystem using supervised machine learning. In *2017 IEEE International Conference on Big Data (Big Data)*. IEEE, 3690–3699.
- [34] Shuhan Yuan, Panpan Zheng, Xintao Wu, and Yang Xiang. 2017. Wikipedia vandal early detection: from user behavior to user embedding. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 832–846.
- [35] Ge Zhang, Zhenyu Yang, Jia Wu, Jian Yang, Shan Xue, Hao Peng, Jianlin Su, Chuan Zhou, Quan Z Sheng, Leman Akoglu, et al. 2022. Dual-discriminative Graph Neural Network for Imbalanced Graph-level Anomaly Detection. In *Advances in Neural Information Processing Systems*.
- [36] Jiawei Zhang, Congying Xia, Chenwei Zhang, Limeng Cui, Yanjie Fu, and S Yu Philip. 2017. BL-MNE: emerging heterogeneous social network embedding through broad learning with aligned autoencoder. In *2017 IEEE International Conference on Data Mining (ICDM)*. IEEE, 605–614.
- [37] Lingxiao Zhao, Saurabh Sawlani, Arvind Srinivasan, and Leman Akoglu. 2022. Graph Anomaly Detection with Unsupervised GNNs. <https://doi.org/10.48550/ARXIV.2210.09535>
- [38] Panpan Zheng, Shuhan Yuan, and Xintao Wu. 2019. Safe: A neural survival analysis model for fraud early detection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 1278–1285.

A SUPPLEMENTARY MATERIAL

A.1 Reproducibility

We release Evolve Path Tracer on [GitHub](#)⁴. We first download full-node BTC raw data with Bitcoin Core. The whole data size is about 500GB. After downloading all blocks before the 700,000th block, we parse all data by Blocksci for querying block, transaction, and address index. The parsed data size is about 400GB. For each address in our dataset, We then prepare its asset transfer paths for the transactions during the first 24 hours. The process was executed on AMD Ryzen 9 3900X Processor with 64.0GB of memory. We implement Evolve Path Tracer in Pytorch and Geometric. All experiments are conducted on a single NVIDIA RTX 2080TI with 11G memory.

A.2 Address Features

We use the following features to characterize an address at a specific timestamp.

- the current balance of the address
- the number of receive (spend) transactions,
- the ratio of receive (spend) transactions number,
- the maximum receive (spend) transactions number,
- the life span of the address,
- address active rate.

A.3 Transaction Features

We use the following features to characterize a transaction, which is the component of every asset transfer path.

- the height interval to the path source,
- the influence (trust) score with the previous transaction,
- the input amount of the previous transaction,
- the transaction fee,
- the total amount (resp. max, min, avg, and var) of all receive (spend) transactions,
- the number of receive (spend) transactions.

A.4 Preparation of Asset Transfer Path

Algo. 1 gives the detail to prepare *Backward Asset Transfer Paths* that reveal where j obtains the asset. The pipeline to construct *Forward Asset Transfer Path* is similar to *Backward Asset Transfer Path*. The only difference is the tracing direction. The essence of each node is a transaction, thus we use a sequence of transaction features to represent an asset transfer path.

A.5 Baseline Models

We give details of our baseline methods from two related tasks: **Malicious Detection in Cryptocurrency**. We compare Evolve Path-Tracer with several models for malicious address detection in cryptocurrencies. For decision tree models, we use address features and path statistic features as the feature set. For GCN models, at each time step, we get the addresses' embedding after two graph convolutional layers as implemented in [30]. Then, we feed the embeddings into a sequential model for prediction.

Algorithm 1: Backward Path Preparation

```

input : Initial Output Tx  $j^o$ , Threshold  $\theta$ , Time Span  $T_{Span}$ .
output: Backward Path Set  $P$ .
1 Initialize Backward Path Set:  $P \leftarrow \{-, 1, j^o\}$ ;
2 Initialize Previous hop recorder:  $P_{pre} \leftarrow \{-, 1, j^o\}$ ;
3 Initialize Ending Flag:  $F_{end} \leftarrow False$ ;
4  $j^o$ 's Time:  $T_{j^o} \leftarrow$  Time of  $j^o$ ;
5 while  $F_{end} \neq True$  do
6   Current hop recorder  $P_{now} \leftarrow \{\}$ ;
7    $F_{end} \leftarrow True$ ;
8   for  $p$  in  $P_{pre}$  do
9      $j \leftarrow$  Output Tx  $p[2]$ ;
10     $I \leftarrow$  Input Tx Set of  $j$ ;
11    for  $i$  in  $I$  do
12       $Prop_i \leftarrow Amt_i / Amt_j$ ;
13       $Score_i \leftarrow Prop_i * p[1]$ ;
14       $T_i \leftarrow$  time of  $i$ ;
15      if ( $Score_i \geq \theta$  and  $T_{j^o} - T_i \leq T_{Span}$ ) then
16        Append  $[j, Score_i, i]$  to  $P_{now}$ ;
17         $F_{end} \leftarrow F_{end} \&\& False$ ;
18   $P_{pre} \leftarrow P_{now}$ ;
19   $P \leftarrow P \cup P_{pre}$ ;
20 return  $P$ 

```

- **Decision Tree** [15, 21] utilize Decision Tree for identifying these malicious addresses.
- **Random Forest** [21] utilize Decision Tree for identifying these malicious addresses.
- **XGB** [8, 21] predict the type of a yet-unidentified entity with Gradient Boosting based algorithms.
- **GCN** [30] encodes the objective address based on its transaction address graph.
- **Skip-GCN** [30] inserts a skip connection between the intermediate embedding and the input node features.
- **Evolve-GCN** [30] updates GCN weights with an RNN module.

Early Rumor Detection on Social Media. For these sequential models, we build an extra path LSTM encoder for a fair comparison. We concatenate address features with the path-encoder output and feed them into the sequential prediction model.

- **GRU** [6] is a typical neural network for sequence modeling. At each time split, previous hidden state and current summation features are fed into the GRU unit to predict the labels for the given addresses.
- **M-LSTM** [34] adopts LSTM for every kind of feature to generate its own temporal features at each timestamp. Here we build three LSTM models for Address Features, Forward Paths, and Backward Paths.
- **CED** [26] also uses GRU for sequence modeling, it proposes the concept of "Credible Detection Point," making it possible to make predictions as early as possible dynamically.

⁴<https://github.com/Cranooooooo/ADS-Demo>

Table 5: Key components and module description. TX stands for the transaction.

Name(Notation)	Description
Influence TX pair ($j \rightarrow i$)	A certain portions of TX i's BTCs come from TX j
Trust TX pair ($i \rightarrow j$)	A certain portions of TX i's BTCs flow to TX j
Backward Asset Transfer Path ($j_n \rightarrow \dots \rightarrow i$)	Build the Influence TX pairs iteratively and link them to form a path
Forward Asset Transfer Path ($i \rightarrow \dots \rightarrow j_n$)	Build the Trust TX pairs iteratively and link them to form a path
Backward Path Graph	Backward Asset Transfer paths share the same source TX are grouped to form a graph
Forward Path Graph	Forward Asset Transfer paths share the same destination TX are grouped to form a graph
T-1 LSTM	LSTM to encode temporal information of address features
E-1 LSTM	An Evolve Path Encoder LSTM for encoding Backward Asset Transfer Path to Backward Path feature
E-2 LSTM	An Evolve Path Encoder LSTM for encoding Forward Asset Transfer Path to Forward Path feature
T-2 LSTM	LSTM to encode temporal information of Backward Path feature
T-3 LSTM	LSTM to encode temporal information of Forward Path feature
BK-Graph Encoder	An Evolve Path Graph GCN for encoding Backward Path Graph to Backward Graph feature
BK-Graph Encoder	An Evolve Path Graph GCN for encoding Forward Path Graph to Forward Graph feature
T-4 LSTM	LSTM to encode temporal information of Backward Graph feature
T-5 LSTM	LSTM to encode temporal information of Forward Graph feature

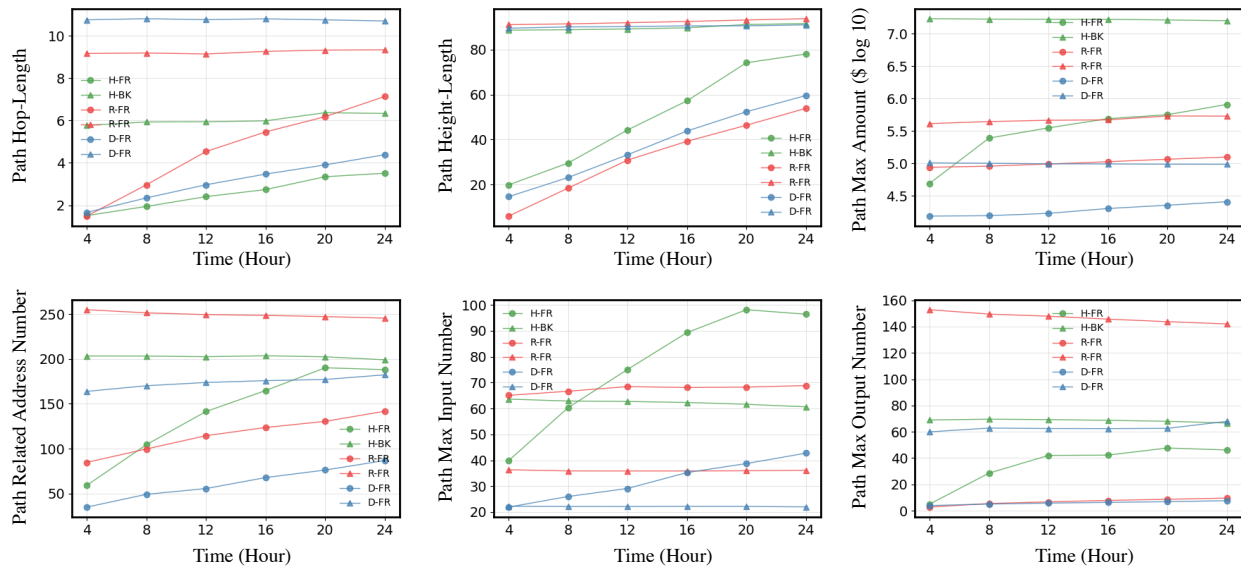


Figure 8: Asset transfer path's statistical properties of different malicious addresses under the backward and forward direction.

- **SAFE** [38] adopts survival probability as the prediction. Instead of predicting the labels directly, it generates hazard

rates for the survival models. The positive samples should die out fast, while the negative samples should stay alive.