Compare Results

Old File:		New File:
ChainletOrbitsVOld.pdf	versus	**ChainletOrbits.pdf**
13 pages (1.50 MB)		**16 pages (1.47 MB)**
2/8/24, 6:43:57 PM		8/9/24, 3:35:29 AM

Total Changes

659

Content

194 Replacements
215 Insertions
121 Deletions

Styling and Annotations

53 Styling
76 Annotations

Go to First Change (page 1)

Chainlet Orbits: Topological Address Embedding for Blockchain

Anonymous Author(s)

Abstract

The rise of cryptocurrencies like Bitcoin has not only increased trade volumes but also broadened the use of graph machine learning techniques, such as address embeddings, to analyze transactions and decipher user patterns. Traditional detection methods rely on simple heuristics and extensive data gathering, while more advanced Graph Neural Networks encounter challenges such as scalability, poor interpretability, and label scarcity in massive blockchain transaction networks.

To overcome the computational and interpretability limitations of existing techniques, we introduce a topological approach, *Chainlet Orbits*, which embeds blockchain addresses by leveraging their topological characteristics in temporal transactions. We employ our innovative address embeddings to investigate financial behavior and e-crime in the Bitcoin, Litecoin, and Ethereum networks, focusing on distinctive substructures that arise from user behavior.

In node classification experiments, our model demonstrates exceptional performance when compared to GNN-based approaches. Furthermore, our approach embeds all daily nodes of the largest blockchain transaction network, Bitcoin, and creates explainable machine learning models in less than 17 minutes which takes days for GNN-based approaches.

CCS Concepts

• **Mathematics of computing** → **Graph theory**; • **Information systems** → *Web searching and information discovery*.

Keywords

node embeddings; e-crime detection; blockchain networks; Bitcoin transaction graphs; graph neural networks

ACM Reference Format:

Anonymous Author(s). 2024. Chainlet Orbits: Topological Address Embedding for Blockchain. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '25), August 3–7, 2025, Toronto, CA.* ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/3447548.3467322

1 Introduction

The success of Bitcoin [25] has led to widespread blockchain adoption by a large number of users worldwide, facilitating a market cap of USD 1.3 trillion for Bitcoin alone. Although recent works have highlighted the use of cryptocurrencies for illegal activities, ranging from ransomware payments [5] to darknet market transactions [15], research in blockchain (e.g., price prediction [7]) has significantly expanded, offering new insights into market dynamics and improving the efficiency of blockchain data analytics. With its growing user base, blockchain is poised to become a profoundly influential research area.

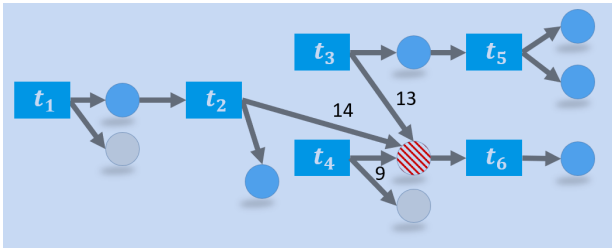


Figure 1: By analyzing transaction patterns within the directed Bitcoin network, we identify specific address (i.e., user account, shown as circular) behaviors in transactions (rectangular nodes, indexed with t) and assign addresses structural roles which we refer to as *orbits*. Here, an e-crime address (red, striped) has been identified and assigned orbits 9, 13, and 14, which are commonly used by addresses associated with ransomware.

Since a blockchain transaction network can be represented as a directed, weighted graph, graph machine learning models can be tailored for supervised tasks such as node classification [7]. Among the techniques used, graph neural network (GNN)-based models (e.g., BAClassifier [1]) learn powerful node embeddings by summarizing the subgraph information of node neighborhoods for each node. However, GNNs are computationally inefficient for the massive blockchain transaction graphs that contain ~600K daily nodes over more than 10 years [39]. Furthermore, the final results often need validation by human analysts, but the complex aggregations and embeddings performed by GNNs make it difficult for analysts, such as police analyzing ransomware patterns, to understand and summarize the transaction behavior.

Interpretability in blockchain analysis can be achieved by capturing the position of a node within the graph and the topology of its neighboring nodes. Graph orbits are ideally suited for this purpose, as they provide a systematic way to capture such information [24]. In graph theory, orbits denote unique configurations of nodes and edges within a homogeneous graph that recurs throughout its structure. In applying orbits to blockchain transaction networks, we reach a key observation: due to their directed and heterogeneous nature of transaction and address nodes, blockchain transaction networks are particularly amenable to orbit analysis as transactions are temporally ordered with well-defined input and output address pools.

We define *chainlet orbits* to determine node positions in a blockchain transaction graph and use the orbits of a node to create its address embeddings. For example, consider the ransomware address in Figure 1 that receives multiple payments, and then consolidates all the received coins into a single address. We show that three orbits (i.e., o9, o12, and o13) are sufficient to summarize the entire subgraph around the ramsomware address. We also apply orbits to the homogeneous transaction networks, such as Ethereum, by utilizing the temporal edge order.

KDD '25, August 3–7, 2025, Toronto, CA.

ACM ISBN 978-1-4503-8332-5/21/08
https://doi.org/10.1145/3447548.3467322

With chainlet orbits, we analyze the entire blockchain networks of Bitcoin, Litecoin, and Ethereum by focusing on special substructures induced by decentralized finance (DeFi) and e-crime, and show that orbits can efficiently match the existing GNNs in classification tasks while creating interpretable results. Furthermore, we find that orbit discovery is a computationally cheap process due to one-time-use-only transaction nodes and low average degree (< 3 on Bitcoin) of address nodes. As a result, we can compute the orbits of all nodes in the massive daily Bitcoin graph in less than 17 minutes.

Previously, orbits have been applied to specific transaction patterns of a limited number of addresses [19, 36], and blockchain address embeddings that use graph compression have been proposed [14]. However, to the best of our knowledge, this is the first work to provide ML-ready address embeddings along with **search and query capabilities over entire blockchains without graph compression/condensation, edge/node filtering or feature engineering**. Our orbits are off-the-shelf applicable to every blockchain transaction network, and raises the potential impact of scalable, production-ready address embedding techniques, which will open new avenues for research and application in blockchains. Our framework is well-suited for human-in-the-loop scenarios, particularly when there is a need for manual analysis of numerous blockchain addresses due to compliance requirements.
Our key contributions can be summarized as follows:

- *Efficient Orbit Discovery*: We present a topological-equivalence-based node embedding called *orbit* that can effectively handle massive blockchain graphs, and give the mathematical foundations of orbits by applying group actions on graphs.
- *Address Behavior Discovery*: We distinguish orbits that capture the transaction behaviors of e-crime and DeFi participants, enabling an analyst to focus on the actions of specific groups of addresses such as those controlled by ransomware operators.
- *Extensive Empirical Evaluation*: We conduct node classification experiments on the entire transaction networks of Bitcoin and Ethereum, and demonstrate that orbit-based models outperform existing topological and graph neural network-based methods.
- *Interpretable Models in Graph ML*: We use orbit patterns to profile typical behavior and, to the best of our knowledge, enable the first search/similarity queries on blockchains.

2 Related Work

The success of Bitcoin [25] has encouraged significant usage of cryptocurrencies. Early efforts in Blockchain graph analysis used heuristics based on transaction behavior to de-anonymize and associate Bitcoin addresses [23]. Two heuristics by Meiklejohn et al. [23] are widely used in Bitcoin address clustering: Co-spending and Transition. Co-spending posits that "if two addresses are inputs to the same transaction, they are controlled by the same user." Transition posits that "if we observe one transaction with addresses A and B as inputs, and another with addresses B and C as inputs, then we conclude that A, B, and C all belong to the same user."

As the number of e-crime transactions increased on cryptocurrencies, algorithmic models beyond simple heuristics have been developed to link Bitcoin addresses by using node features on the network [13]. Mark et al. [35] introduced the Elliptic dataset, which contains over 344K payments among 200K Bitcoin nodes in 49 transaction graphs. The authors employed machine learning techniques including Logistic Regression, Random Forest (RF), Graph Convolutional Networks (GCNs) [17], and EvolveGCN [28] for e-crime payment detection. The RF model achieved a recall score of 0.67 in the e-crime category, while GCNs achieved a score of 0.51. The Elliptic dataset only contains anonymized transactions and omits address nodes, which we need to extract orbits. As a result, we could not apply our approach to the dataset. Furthermore, our experiments show that using a neighborhood approach yields better results as measured by AUC. Vassallo et al. [34] detected illicit cryptocurrency activities such as Ponzi schemes, and used the Elliptic dataset to evaluate its effectiveness at both the address and transaction levels.

Feature extraction on the Bitcoin transaction network has been widely studied [7]. For example, Chaehyeon et al. [21] employed supervised machine learning algorithms to classify e-crime addresses in the Bitcoin network. The authors extracted classification features based on the characteristics of Bitcoin transactions, such as transaction fees and size. Our method avoids domain-specific feature engineering and creates topological embeddings that can be applied to any blockchain network.

Graph neural networks achieve better results and do not require feature engineering. AlArab et al. [6] proposed a model that combines GCNs and linear layers to classify illicit nodes in the Elliptic dataset [35]. The results showed that the model had an overall classification accuracy of 97.40% and a recall of 0.67 for detecting illicit transactions. Nan et al. used an autoencoder with graph embedding to detect mixing and demixing services for Bitcoin, and the proposed model effectively performed anomaly detection [26]. Pham et al. used unsupervised learning to identify suspicious nodes in the Bitcoin transaction graph [30]. However, these prior studies often focus on identifying addresses associated with specific transaction behaviors, such as exchange identification [32], money laundering [36], and semantic representation [37], without analyzing the entire network or focusing on time efficiency. For example, related work by Huang et al.[14] extracts an ego network for each blockchain address, compresses the graph, and uses GNNs for graph classification (we test a similar approach and report its results in Table 2 for a subset of addresses). However, such an approach cannot be scaled to the massive Bitcoin network for every address. Unlike them, we operate on entire blockchain networks and provide interpretable results.

3 Background and Preliminaries

In this section, we introduce preliminaries on blockchains. Afterward, we introduce graph machine learning on blockchain networks and discuss node representation learning.

A blockchain is an immutable public ledger that records transactions in discrete data structures called *blocks* [25]. There are two types of blockchains in terms of transaction structures: Unspent Transaction Output (UTXO)-based and account-based. The earliest blockchains, such as Bitcoin and Litecoin, are UTXO-based, where the heterogeneous network contains both address and transaction node types, and a transaction represents the transfer of coins between N-to-N addresses. In account-based blockchains, such as

Ethereum, each transaction is 1-to-1, hence the network contains only address nodes and omits the transaction node. In both cases, the graph is directed and weighted.

On blockchain, an address may appear in multiple blocks. To mine address behavior in time, we divide the network into 24-hour-long windows by using the UTC-6 timezone as reference. This window approach serves two purposes. First, the induced 24-hour network allows us to capture how many days a coin moves in the network. Second, temporal information has proved useful for clustering criminal transactions (see Figure 7 in [13]).

Machine Learning on Blockchain. Blockchain networks provide an ideal platform for developing and refining modern graph machine learning techniques for several reasons. Firstly, the networks often exhibit adversarial patterns such as those stemming from e-crime transactions (see Section 7) that compel a model to consider adversarial behavior. Secondly, the massive network data requires the creation of efficient ML models. Thirdly, the extensive data volume, coupled with labeled data scarcity, leads to a high rate of false positives, rendering standard ML techniques less effective (see Table 2 of [31]). Consequently, the realm of graph ML on blockchain has emerged as a vibrant area of ongoing research [7, 16].

4 Chainlets and Orbits

This section outlines chainlets and provides mathematical background before defining our topological address embeddings, called orbits. We then distinguish between active and passive orbits based on address behavior. We begin by sharing our intuition and motivation.

Intuition behind Orbits. The temporal position of an address node within a directed transaction graph (as shown in Figure 2) determines its structural role, referred to as an *orbit*, which can be leveraged in supervised learning tasks. For example, Figure 1 displays the most common orbits of ransomware addresses on a UTXO network which e-crime operators cannot change without incurring transaction costs. Our primary objective is to establish these roles and utilize them in graph machine learning models.

Orbits apply to UTXO blockchains naturally, because a one-time-use-only transaction appears in the heterogeneous graph as a node, which allows ordering address nodes (see Figure 2). For this reason, we use UTXO networks to define orbits. In Appendix A.2, we extend our definition to account-based blockchains by using the temporal order.

4.1 Chainlets

We model the UTXO network with a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{B})$ from a set of transactions TX and input and output addresses in TX. On $\mathcal{G}$, $\mathcal{V}$ is a set of nodes, and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is a set of edges. $\mathcal{B} = \{\textbf{Address}, \textbf{Transaction}\}$ represents the set of node types. A node $u \in \mathcal{V}$ has a node type $\phi(u) \in B$.

For each edge $e_{uv} \in \mathcal{E}$ between adjacent nodes u and v, we have $\phi(u) \neq \phi(v)$. In other words, there is no edge between the same node types (Transaction→Transaction or Address→Address), and an edge $e \in \mathcal{E}$ represents a coin transfer between an address node and a transaction node.

Chainlets. Chainlets have been introduced to define position invariant versions of transaction patterns (see overviews by [2, 3]).

The key idea of the chainlet approach is to offer a lossless summary of all transactions by assigning two indices i, o in the $[0, n]$ range to each transaction with respect to its input and output addresses. A Bitcoin subgraph $\mathcal{G}' = (\mathcal{V}', \mathcal{E}', \mathcal{B})$ is a *subgraph* of $\mathcal{G}$, if $\mathcal{V}' \subseteq \mathcal{V}$ and $\mathcal{E}' \subseteq \mathcal{E}$. If $\mathcal{G}' = (\mathcal{V}', \mathcal{E}', \mathcal{B})$ is a subgraph of $\mathcal{G}$ and $\mathcal{E}'$ contains all edges $e_{uv} \in \mathcal{E}$ such that $(u, v) \in V'$, then $\mathcal{G}'$ is called an *induced subgraph* of $\mathcal{G}$. Two graphs $\mathcal{G}' = (\mathcal{V}', \mathcal{E}', \mathcal{B})$ and $\mathcal{G}'' = (\mathcal{V}'', \mathcal{E}'', \mathcal{B})$ are called *isomorphic* if there exists a bijection $h : \mathcal{V}' \to \mathcal{V}''$ such that all node pairs $u, v \in \mathcal{V}'$ are adjacent in $\mathcal{G}'$ if and only if $h(u)$ and $h(v)$ are adjacent in $\mathcal{G}''$.

Definition 1 (k-chainlet). *Let k-chainlet C_k in $\mathcal{G}$ be the subgraph of $\mathcal{G}$ containing k subsequent transaction nodes, all output nodes of these k transaction nodes, and all input nodes of the first transaction (See Figure 2).*

A 1-chainlet lists a transaction node with its input and output addresses. Figure 2 shows a 2-chainlet; transactions t_1 and t_2 share the common address a_1. A k-chainlet where $k > 1$ has a natural ordering of transactions because transactions are ordered in blocks which, in turn, are ordered in a blockchain. This natural order allows us to assign roles to an address depending on where the address appears in a 2-chainlet. In the next section, we will count address roles and list them as address orbits.

4.2 Orbits

Before we define chainlet orbits, we need to recall a few basic notions from graph theory. A mapping $\varphi : \mathcal{G} \to \mathcal{H}$ is called *a graph isomorphism* if $\varphi : \mathcal{V}_\mathcal{G} \to \mathcal{V}_\mathcal{H}$ is a bijection such that $u, v \in \mathcal{V}_\mathcal{G}$ are adjacent in $\mathcal{G}$ if and only if $\varphi(u)$ and $\varphi(v)$ are adjacent in $\mathcal{H}$. If there exists an isomorphism from C_k to another chainlet C'_k of $\mathcal{G}$, we call C'_k an *occurrence* of C_k in $\mathcal{G}$. We define the chainlet class of C_k in $\mathcal{G}$ as the set of all occurrences of C_k as follows:

Definition 2 (Chainlet Classes). *Let $\mathcal{A}_k(\mathcal{G})$ be the set of all k-chainlets in $\mathcal{G}$. For a given k-chainlet $C_k \in \mathcal{A}_k(\mathcal{G})$, the equivalence class of C_k, $\mathcal{E}(C_k)$, is the set of all isomorphic copies (occurrences) of C_k in $\mathcal{A}_k$.*

Having defined equivalence classes of chainlets, we next define an equivalence class of address nodes for a fixed chainlet class.

Definition 3 (Orbit of an Address Node). *Let C_k be a k-chainlet in $\mathcal{G}$, and u be an address node in C_k. We define* the orbit of u with respect to C_k *as the collection of the images $\varphi(u)$ where φ is an isomorphism from C_k to another k-chainlet C'_k in $\mathcal{G}$, i.e. $O_u = \{\varphi(u)\}$. We call $u \in C_k$ and $v = \varphi(u) \in C'_k$* similar nodes *wrt C_k, i.e., $u \sim v \ (mod\ C_k)$.*

We use the notation o to refer to orbits, e.g., o15 for the orbit 15.

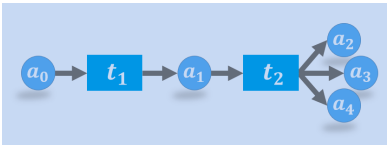
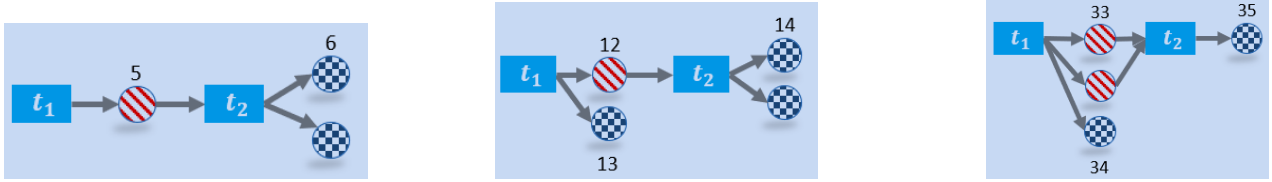


Figure 2: A 2-chainlet of five addresses $(a_0, \ldots, a_4)$. Each of the two transactions (t_1, t_2) has one input address, but t_2 has three output addresses.



(i) Orbits 5 and 6. **(ii) Orbits 12 , 13 and 14.** **(iii) Orbits 33, 34 and 35.**

Figure 3: Orbit samples. Active (striped) and passive (checkered) orbits denote the role of an address. Formal descriptions of all 48 orbits are given in the Appendix C.

Notice that the similarity of two nodes depends on the fixed chainlet class $\mathcal{E}(C_k)$. In other words, a node can belong to different chainlets C_k and C_k' because of their relative locations to the transaction nodes. Hence, the orbit of node u with respect to C_k versus with respect to C_k' can be completely different. In the remaining text, if the fixed chainlet type C_k is understood, we will refer to it as simply *orbit of* u, O_u. Note also that if $u \sim v$, then the corresponding chainlets containing u and v must be in the same equivalence class of chainlets. In other words, any orbit O_u belongs to only one chainlet class $\mathcal{E}(C_k)$. By definition, if two nodes are similar, then they have the same orbit and vice versa, i.e. $u \sim v \iff O_u = O_v$. However, the relationship between the size of the orbit O_u of a node $u \in C_k$, and the size of the equivalence class of chainlet $\mathcal{E}(C_k)$ remains unspecified, which we establish next.

Let $\mathcal{I}(C_k)$ be the set of all isomorphisms between any two k-chainlets in $\mathcal{E}(C_k)$. $\Lambda(C_k)$ be the group of all automorphisms (self-isomorphisms) of C_k. Then, for any node u in C_k, we define the stabilizer subgroup of u in $\Lambda(C_k)$ as $\mathcal{S}(u) = \{\varphi \in \Lambda(C_k) | \varphi(u) = u\}$. Let $|K|$ represent the cardinality of the set K.

Theorem 4.1. *Let C_k be a k-chainlet in $\mathcal{G}$, and u be a node in C_k, and $I(C_k), \Lambda(C_k), S(u)$ be as defined above. Then, we have*

$$|\mathcal{I}(C_k)| = |\mathcal{E}(C_k)|^2 \cdot |\Lambda(C_k)| \quad \text{and} \quad |O_u| = |\mathcal{E}(C_k)| \cdot \frac{|\Lambda(C_k)|}{|\mathcal{S}(u)|}$$

The proof of the theorem is given in Appendix B.

Choice on k-chainlets. We focus on 2-chainlet classes and define address orbits for 2-chainlets, preferring them over the alternatives $k = 1$ and $k \geq 3$ for the following reasons. $k = 1$ only models the receiving behavior of coins, whereas we want to model both the sending and receiving behavior, as captured by $k = 2$. Note that k=2 subsumes k=1. $k \geq 3$ is not informative because once coins are received (modeled with $k = 1$) and spent (modeled with $k = 2$), the old owner of the coins has no control over the transactions created by the new buyer of the coins. Hence, $k = 2$ is necessary and sufficient in understanding address behavior. Our analysis shows that a blockchain address has few unique orbits on average (e.g., 3 on Bitcoin); preferring $k = 2$ over $k \geq 3$, which reduces the embedding size, mitigates the sparsity issues in the computed orbit data.

When counting the total number of distinct orbits, we exclude the input addresses of transaction t_1 (i.e., the first transaction involved in a 2-chainlet) as their inclusion leads to a count of more than 3×48 orbits. By omitting the input information from the initial transaction, the orbit count is reduced to 48, **resulting in the embedding of each address into a 48-dimensional vector** (See Appendix Figures 8, 9, 10, 11 for all 48 orbits). Our experiments in Section 6 demonstrate that our orbit definition provides strong classification power.

4.3 Active and Passive Orbits

We define an orbit as active or passive based on the role of the associated address within a 2-chainlet.

Definition 4 (Active Orbit). *Let C be a 2-chainlet. An active orbit with respect to C is an orbit where its addresses appear as the output of the first and the input of the second transaction in the corresponding occurrence of C. When C is understood, we denote the set of active orbits as O_{act}.*

Active orbits indicate behavior created by the address owner, as the address in this orbit initiates a new transaction and sends coins to other addresses. Next, we define passive orbits as follows:

Definition 5 (Passive Orbit). *Let C be a 2-chainlet. A passive orbit with respect to C is an orbit where it appears as the output of the first or the second transaction but not as an input of the second transaction in the corresponding occurrence of C. When C is understood, we denote the set of passive orbits as O_{pass}.*

Unlike active orbits, passive orbits for a chainlet class $\mathcal{E}(C)$ receive coins without shaping any occurrences of C, as the address in this orbit only receives coins from another address, without initiating a new transaction. It is worth noting that an address can belong to an active orbit for one chainlet class while belonging to a passive orbit for a different chainlet class. This means that an address may exhibit both initiating and receiving behavior in different 2-chainlet types. Figure 3 provides sample active and passive orbit addresses in 2-chainlets.

The distinction between active and passive orbits is significant for analyzing specific behaviors within the transaction graph. Active orbits help train classifiers to identify spending patterns, such as paying for ransom, as they reflect behavior initiated by the address owner. Conversely, passive orbits are useful for examining receiver behavior, as they involve receiving coins without influencing chainlets. By studying both orbit types, we gain a deeper understanding of the transaction graph and its participants.

5 Experimental Setup

Datasets. We extract data from the Bitcoin, Litecoin and Ethereum transaction networks.

Bitcoin Transaction Network. We have downloaded and analyzed the **complete Bitcoin transaction graph** from its inception in 2009, by utilizing the Bitcoin Core Wallet and the Bitcoin-ETL library https://github.com/blockchain-etl/bitcoin-etl.

BitcoinHeist Addresses. Our ransomware dataset is a union of datasets from three widely adopted studies: Montreal [27], Princeton [13] and Padua [10]. The combined dataset contains addresses from 27 ransomware families. 19930 unique ransomware addresses appear a total of 62641 times since 2009.

Bitcoin Darknet Addresses. We have downloaded the Grams dataset from Darknet Market Archives [8]. The dataset spans from June 9, 2014, to July 12, 2015. We identified 7557 unique addresses associated with transactions from darknet marketplaces a total of 1288100 times.

Ethereum Transaction Network. We have downloaded and analyzed the **complete Ethereum transaction graph** from its inception in 2015, by utilizing the Geth Wallet and the Bitcoin-ETL library (https://github.com/blockchain-etl/ethereum-etl).

Ethereum DeFi Addresses. Our DeFi dataset is a union of lending, asset, derivative and decentralized exchange addresses from [18]. The combined dataset contains 1407 addresses from the four protocol families, which appear a total of $108M$ times since 2015.

Litecoin Transaction Network. We have downloaded and analyzed the **complete Litecoin transaction graph** from its inception in 2011, by utilizing the Litecoin Core Wallet and RPC calls to it.

Hardware: We ran experiments on a single Dell PowerEdge R630, featuring an Intel Xeon E5-2650 v3 Processor (10-cores, 2.30 GHz, 20MB Cache), and 192GB of RAM (DDR4-2133MHz).

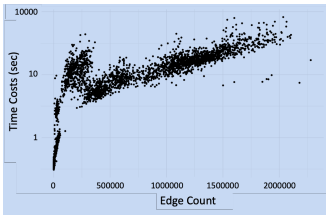


Figure 4: Orbit discovery time costs from the daily Bitcoin transaction graph. Each data point denotes a specific day.

Preprocessing: We do not apply any normalization or preprocessing. Extracting orbits is not demanding in terms of computational resources, so we use the entire transaction network without removing any edges or nodes. The full address orbits, which are 27 GB in size, are publicly available at ***.org/orbits. Classifier data is given at https://huggingface.co/datasets/Chainlet/Blockchain_Orbits/.

Computational Complexity of Orbit Discovery: Extracting a 2-chainlet requires three searches on the adjacency matrix. First, we must find all transaction nodes which has a cost of $O(\mathcal{V})$. Next, for every transaction pair t_1 and t_2, we must find if $\exists a \in \{O(t_1) \cap I(t_2)\}$, where $O(t_1)$ represents the set of outputs of transaction t_1 and $I(t_2)$ represents the set of inputs of transaction t_2. The second step thus has a complexity of $O(\mathcal{V}^3)$. However, 57.40% of Bitcoin transactions have less than two input and two output addresses. Hence the complexity is reduced to $k \times O(\mathcal{V}^2)$ where $k \approx 2$. As Figure 4 shows, on the majority of days, orbit extraction is completed within 1000 seconds (less than 17 minutes).

Code and Parameters: Our hyper-param selections and Bitcoin and Ethereum prediction models are given at Appendix E. Our orbit code is shared at https://github.com/chainletRepo/chainlet.

6 Experimental Results

We apply orbit discovery to the Bitcoin, Litecoin, and Ethereum blockchains. As ground truth address labels are predominantly available for the Bitcoin network, we primarily focus on the Bitcoin transaction network, detail our findings in this section. In Appendix E.1 we discuss the DeFi address classification experiments for Ethereum, and report AUC values as high as 0.86. The orbit extraction results for Litecoin are given in Appendix E.

This section begins by presenting the outcomes of widely employed heuristics for clustering Bitcoin addresses as a baseline. Then, we report orbit embeddings-based node classification results i) on the BitcoinHeist benchmark and ii) the large Bitcoin transaction network dataset.

6.1 Heuristics

We employ the co-spending and transition heuristics (Section 3) with all the history of ransomware addresses and discover only 40 unique addresses from CryptoLocker (Padua), CryptoWall (Padua), CryptoTorLocker2015 (Montreal), and CryptoTorLocker2015 (Padua) families. The number is low because researchers of the datasets had already used the heuristics to expand the datasets in an earlier time span.

6.2 Orbits on the BitcoinHeist Benchmark

We use orbits of Bitcoin addresses in a binary (white vs. ransomware) node classification task on the BitcoinHeist dataset and compare our results to those reported in the article by Akcora et al. [5]. We obtained 2916697 ransomware and white addresses from the authors of the dataset. The BitcoinHeist dataset comprises 1048576 transactions from January 2009 to December 2018. Transactions with amounts less than ₿0.3 were removed as ransomware payments are usually significantly larger. Since the dataset only includes ransomware and white addresses, we present the performance of our methods in terms of binary classification by using a Random Forest [9] with 300 trees and a 80/20 train/test split.

Orbits are more informative than BitcoinHeist features. Table 1 shows the AUC scores of the ROC curve for the binary address classification task. In BitcoinHeist+Orbit, we combine BitcoinHeist features and orbits of addresses. The AUC of the BitcoinHeist and BitcoinHeist+Orbit models improves as the white sample rate increases from $50K$ to $100K$, but larger sample rates decrease performance. However, the performance of the Orbit model remains relatively stable with larger white sampling rates, decreasing only by 0.018 from $50K$ to $1M$. This suggests that orbits are relatively robust against the class imbalance issues that often mar blockchain data analytics, as labeled data is typically scarce in these tasks.

The superiority of the BitcoinHeist+Orbit model over the Orbit model is attributed to the integration of the income feature from the BitcoinHeist dataset, which calculates the sum of incoming coin amounts for each address. Orbits do not consider the coin

Table 1: AUC scores for address classification using features from the BitcoinHeist dataset and orbits. White addresses were under-sampled without replacement, while ransomware addresses were not under-sampled. The results indicate that using orbits improves address classification significantly.

AUC@Sample	BitcoinHeist	Orbit	BitcoinHeist + Orbit
AUC@1M	0.791±0.005	0.839±0.003	**0.895±0.006**
AUC@500K	0.798±0.006	0.849±0.002	**0.903±0.003**
AUC@100K	0.808±0.003	0.856±0.005	**0.908±0.003**
AUC@50K	0.807 ±0.006	0.857±0.004	**0.908±0.003**

amounts that are transferred. As the ransom amounts are typically similar (e.g., ₿0.5), the income feature of BitcoinHeist provides a good feature to predict the ransomware label. The Mean Decrease in Impurity [9] identifies income, o9, and o12 as the three most important features of the classifier. Incorporating the income feature into the orbits data is computationally cheap as it only requires summing the incoming amounts for each address.

BitcoinHeist features are costlier than orbits. It's worth mentioning that the BitcoinHeist study employs graph flow-based address features, which are computationally expensive. To minimize the expenses, the authors applied edge filtering with a threshold of 0.3 Bitcoins. On the other hand, orbit discovery is computationally cheap, and unconstrained by the time complexity issues. This permits us to extract orbits for all Bitcoin addresses.

GNNs on the BitcoinHeist Benchmark. We could not apply existing node (i.e., address) classification approaches [38] to the Bitcoin transaction network because the network is too large and contains as few as four e-crime addresses per day (over ten thousand days). Instead, we re-formulated the node classification task as the classification of the subgraph of the two-hop neighborhood of an address. Formally, we define the neighborhood graph of an address as the combination of the induced subgraphs of i) 2-chainlets where the second transaction outputs coins to the address and ii) 2-chainlets where the first transaction outputs coins to the address. An example of a neighborhood graph is shown in Figure 1.

There are several advantages of using the neighborhood approach. Firstly, the neighborhood approach enables an inductive learner where an address from any day can be classified by using its neighborhood graph from that day, while transductive node classification is restricted to the specific graph on which it was trained and cannot be applied to new, unseen graphs. Secondly, the neighborhood approach provides us with the ability to control the number of addresses in classification, as GNNs can have enormous computational costs when applied to very large datasets.

GNN Models. We evaluate the orbit-based classifier against four graph neural networks on the BitcoinHeist data: DGCNN [40], GIN [39], a GCN [17], and GraphSage [12]. We report the average accuracy of five runs, along with their standard deviation. In these models, we adopted the grid search parameters from the benchmark article by Errica et al. [11]. While there are numerous methods, we are not conducting a comprehensive comparison of all of them. Instead, we aim to provide a good understanding of what GNNs could achieve if we could run them for each blockchain address.

Table 2: AUC values for address classification using BitcoinHeist addresses. Both white and ransomware addresses are under-sampled without replacement.

Sample	GCN	GIN	DGCNN
20K	0.76±0.002	0.90±0.02	0.79±0.09
10K	0.70±0.004	0.88±0.01	0.75±0.09
5K	0.70±0.002	0.86±0.02	0.78±0.09

Data Imbalance. Despite utilizing the neighborhood graph, neural networks still face the challenge of data imbalance. Blockchain graphs are massive, but contain only a few e-crime addresses. To address this problem, we take two steps. Firstly, we create a balanced dataset to evaluate the graph neural networks where close to half of the addresses are white (i.e., 57%). Secondly, we restrict the number of addresses fed into the model. The GNNs train on the neighborhood graph, complete with its edge weights. Additionally, these GNNs utilize the incoming bitcoin sum at each node as a node feature.

The resulting AUC scores are shown in Table 2. We also attempted to train a robust GraphSage [39] model, but it did not perform well as the model AUC did not improve throughout the training. We believe that this could be due to the sparsity of the Bitcoin transaction network, where many addresses have only one or two neighbors, and the usage of sampling-based approaches leads to disconnected graphs.

Comparison of Orbits and GIN. As shown in Table 2, the GIN model exhibits remarkable performance and consistently improves its AUC score as more training data is added. However, the model's complexity imposes limitations on the amount of training data that can be effectively employed. Furthermore, these results are achieved with balanced data, which may not hold when applied to the entire dataset. As demonstrated in the following section, our orbit-based method excels in achieving enhanced AUC scores, particularly in the challenging task of three-way classification when trained on extensive e-crime data.

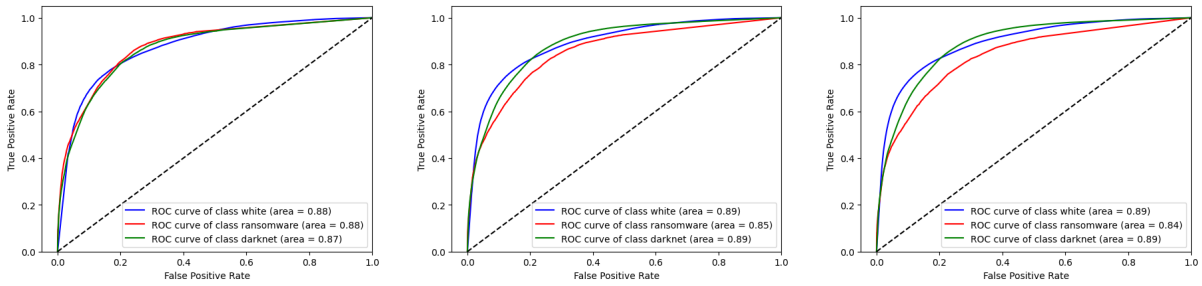
The AUC results of GIN in Table 2 (0.9 AUC) closely match those of the orbit-based results in Table 1 (0.857 AUC). However, GIN is constrained to smaller datasets containing fewer than 20K addresses, whereas orbit-based models can train on a complete day's worth of addresses for predictions in the subsequent day. This enables us to incorporate temporal patterns into our models and assign greater significance to recent orbit patterns.

6.3 Orbits on the Large Bitcoin Network

The goal of this experiment is to generalize e-crime address classification performance with a larger dataset. We take a sample of 1% of

Table 3: AUC scores for active (O_{act}) and passive (O_{pass}) orbits on 1% of Bitcoin addresses (4.3M addresses).

Class	O_{act} AUC	O_{pass} AUC
White	0.731±0.006	**0.870±0.002**
RS	0.725±0.004	**0.736±0.001**
DM	0.727±0.004	**0.876±0.001**



(a) White sample=100K. **(b) White sample=500K.** **(c) White sample=1M.**

Figure 5: AUC scores for orbit-based e-crime address classification. White and darknet classes are undersampled. Orbit performance is quite robust against the increasing sample sizes.

the daily white addresses in the Bitcoin history, resulting in 3.16 million addresses. Our labeled node data contains appearances of 1.28 million darknet market addresses and 64K ransomware addresses.

Figure 5 illustrates the three-way address classification performance for white, ransomware, and darknet addresses. The figure tracks the AUC score for each class with a one-versus-rest Random Forest classifier, as more white data is added in three steps to the classifier input. Training/test split is 80/20. The consistently high AUC score, above 0.8, demonstrates the strength of the classifier.

Active and passive orbit-based classifiers. We also investigate the role of orbit types in detecting e-crime payments. We consider two types of orbits as outlined in Section 4: active orbits, which refer to the behavior of e-crime operators such as darknet market sellers and ransomware hackers, and passive orbits, which refer to the behavior of darknet market buyers or ransomware victims. In Table 3, we evaluate the performance of different feature sets by limiting the data features to either active or passive orbits and calculating the macro-averaged AUC performance.

The results show that active orbit-based AUC values per class are similar but lower than those yielded by passive orbit-based classifiers. This interesting result implies that analyzing payer (i.e., victims in RS and buyers in DM) patterns is quite efficient in detecting e-crime addresses. Consequently, successful e-crime operations in the future may require dictating payer behavior to avoid detection, with directives such as "create t_1 with many outputs, and pay the coins from t_1 to my address in a t_2 that contains more than 2 inputs and 2 outputs".

7 Adversarial Patterns of E-Crime

Our analysis has revealed a total of 22682 unique orbit patterns associated with addresses linked to ransomware and 546838 unique orbit patterns associated with addresses associated with darknet marketplaces. Typically, an address is only connected to a single occurrence within an orbit for any given day, resulting in an orbit value of 1. Table 4 lists the most frequent orbit patterns for classes.

Additionally, 2.63% of all darknet market addresses appear in two 2-chainlets daily, within orbits o13 and o14. However, it should be noted that these orbits are prevalent across all types of addresses, including those associated with ransomware. As Table 4 illustrates, the top three patterns associated with ransomware addresses make up 16.53% of all ransomware addresses, indicating a lack of diversity

Table 5: Top-5 orbit patterns where addresses have the highest likelihood of being associated with ransomware. A checkmark indicates non-zero orbit count. The RS% column in the table represents the percentage of addresses that are identified as ransomware addresses. W (white), DM (darknet market) and RS (ransomware) are counts of addresses with the related pattern. For example, for the first orbit pattern, 95.9% (i.e., 94) of the addresses with that pattern are identified as ransomware addresses, while the remaining addresses are either white or associated with darknet.

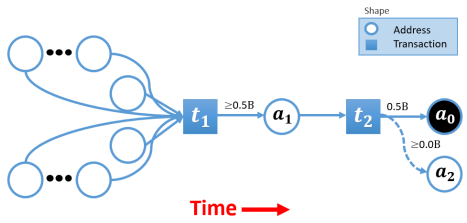
o1	o6	o8	o9	o13	o14	o17	o25	o28	o31	o32	o34	o37	o40	o42	o43	o45	o47	W	DM	RS	RS%
		✓					✓	✓	✓			✓		✓		✓		1	3	94	95.9
		✓				✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	64	3	216	76.3
✓	✓		✓	✓	✓													21	91	81	41.9
✓			✓	✓	✓													230	186	131	23.9
✓			✓	✓	✓													106	150	108	29.7

in ransomware address behavior. This implies that ransomware operators tend to use a limited number of patterns, which may facilitate tracking ransomware payments.

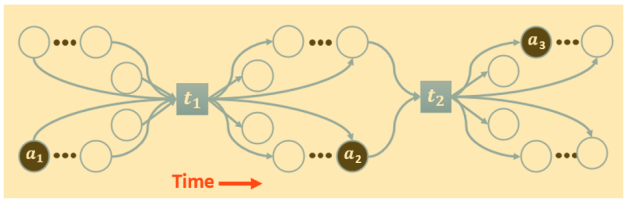
Table 5 further studies the RS patterns. We binarize orbit counts and use a checkmark to indicate that an address has a non-zero count for the associated orbit. Certain patterns exhibit a strong association with addresses linked to ransomware. A significant proportion of addresses (95.9%) with the pattern in the first row of the table are identified as ransomware addresses. Probably, the remaining lone white address with this pattern is an undisclosed e-crime address. Table 6 shows the most and the least likely patterns associated with white addresses. Based on these patterns, Figure 6 shows typical e-crime behavior for ransomware and coin mixing transactions (see Appendix Section C.1) where darknet market sellers launder their coins.

Table 4: Top-3 most frequent orbit counts of darknet market, ransomware (RS) and white addresses. Orbits that do not appear in the top-3 have been excluded. Percentages are computed within each address class.

	Rank	o1	o9	o12	o13	o14	Perc.
Darknet	1	0	0	0	1	1	2.63
	2	1	0	0	0	0	1.13
	3	1	0	0	0	1	0.97
RS	1	0	1	0	1	1	6.69
	2	0	0	0	1	1	6.61
	3	0	0	1	1	1	3.23
White	1	0	0	1	0	1	5.93
	2	0	0	1	1	1	4.89
	3	1	0	0	0	0	4.40



(a) An example of a ransomware 2-chainlet. The victim a_1 (orbit 05) gathers necessary coins from a blockchain exchange (e.g., Coinbase.com) in a transaction (t_1) with one output. The victim then creates a payment transaction t_2, sending coins to the dark address a_o (orbit 06). Any remaining change coins are sent to the victim's newly created change address a_2.



(b) An example of a coin mixing 2-chainlet. A dark address a_1 inputs coins into a many-to-many transaction, which outputs the dark coins to a_2 (orbit 039). Subsequently, a_2 inputs money into t_2, pouring dark coins into a_3 (orbit 041). This constitutes two rounds of coin-mixing, which can be repeated many times, albeit with increasing transaction and commission fees.

Figure 6: Illustration of ransomware and coin-mixing 2-chainlets. It's important to note that the coin-mixing pattern remains inevitable for e-crime operators. Likewise, the ransomware operator lacks control over t_1 unless they directly dictate to the victim. Consequently, the orbits associated with these e-crime instances remain clearly defined and resistant to alteration, which greatly aids in monitoring and tracing e-crime.

Table 6: The top-3 and bottom-3 orbit patterns with the lowest and highest likelihood of being white addresses, respectively. A checkmark indicates non-zero orbit count. The Non-White% column in the table represents the percentage of addresses that are NOT identified as white addresses. W (white), DM (darknet market) and RS (ransomware) are counts of addresses with the related pattern. For example, for the first orbit pattern, 99.9% (i.e., 9800) of the addresses with that pattern are identified as ransomware, while the remaining four addresses are white.

o10	o12	o13	o14	o25	o28	o30	o31	o32	o34	o37	o40	o41	o43	o44	o45	o46	o47	W	DM	RS	Non-White%
							✓	✓			✓	✓				✓	✓	4	0	9800	99.9
					✓	✓		✓										61	14	7316	99.0
✓	✓																	116	10	10666	98.8
				✓	✓		✓	✓	✓	✓				✓	✓	✓	✓	12915	0	1249	8.8
✓	✓	✓	✓															6029	1	442	6.8
				✓	✓		✓	✓	✓	✓								9309	0	533	5.4

Our analysis has provided us with the following insights into e-crime patterns:

- On average, white addresses have three distinct non-zero orbits, while darknet and ransomware addresses have six. These higher-than-expected values happen because, despite the Bitcoin community's discouragement of address reuse, many e-crime transactions involve multiple payments to the same address, indicating a disregard for common privacy practices.
- Orbit 1 in darknet addresses shows that transferred coins are not re-transferred on the same day, suggesting that payments remain in the receiving address. This reflects the historical escrow practices of darknet markets, where payments were held until goods were confirmed received. Notably, Orbit 1 does not feature among the top 10 orbits for ransomware or white addresses.
- Orbits 9 and 12 in ransomware addresses indicate immediate forwarding of payments to new addresses.
- Orbit 9 indicates merged coins from multiple payments. However, none of these orbits connect to coin-mixing transactions (see Figure 6b for coin-mixing transactions).
- Orbits 30, 31, 32, 39, 40, 41, 46, and 47 are observed in chainlets that may be involved in coin-mixing transactions. Interestingly, these orbits are not present in the top 10 most common patterns for any address type. E-crime coins are first combined in an address and then coin-mixed (see Appendix C.1).

Extracting such typical address behavior is immensely important; the learned orbit patterns can be queried (e.g., find all addresses appearing in ransomware followed by coin-mixing transactions) or searched (e.g., find all addresses that have the same transaction behavior as a_x) by a human analyst, and the results can be used to create interpretable ML models.

Limitations of orbits. By definition, orbits only consider topological information but ignore edge weights. In blockchain networks, edge weights represent transaction volumes and are crucial features in many classification tasks, such as ransomware address prediction, where ransom amounts tend to be similar. The exclusion of edge weights can potentially affect the accuracy of the results. Furthermore, orbits' reliance on structural features limits its ability to differentiate between transactions, which might share identical topological patterns but differ dramatically in their financial implications.

8 Conclusion

We have proposed a topological address embedding model, called *chainlet orbits*, that captures a node's structural role in a blockchain graph. This approach results in compact and effective embeddings that can be easily integrated into node classification models, outperforming GNNs that can be trained for a small subset of the addresses only. Additionally, the orbit patterns can be visualized and used with interpretable models.

Our approach is applicable to graphs of both UTXO and account-based blockchains. The orbit discovery process is computationally cheap, and scales to massive graphs easily; our approach embeds all daily nodes of the Bitcoin transaction network in less than 17 minutes.

Future work will focus on multi-modal graphs where meta attributes of transactions can be incorporated as node features. Another promising area of research is developing self-supervised ways to learn node and edge functions from data to define new orbits.

References

[1] Nazmiye Ceren Abay, Cuneyt Gurcan Akcora, Yulia R Gel, Murat Kantarcioglu, Umar D Islambekov, Yahui Tian, and Bhavani Thuraisingham. 2019. ChainNet: Learning on Blockchain Graphs with Topological Features. In *2019 IEEE International Conference on Data Mining (ICDM)*. IEEE Computer Society, 946–951.

[2] C. G. Akcora, A. K. Dey, Y. R. Gel, and M. Kantarcioglu. 2018. Forecasting Bitcoin Price with Graph Chainlets. In *The PAKDD, Melbourne, Australia*. 1–12.

[3] Cuneyt Gurcan Akcora, Yulia R Gel, and Murat Kantarcioglu. 2021. Blockchain networks: Data structures of Bitcoin, Monero, Zcash, Ethereum, Ripple, and Iota. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* (2021), e1436.

[4] Cuneyt Gurcan Akcora, Yulia R Gel, and Murat Kantarcioglu. 2022. Blockchain networks: Data structures of Bitcoin, Monero, Zcash, Ethereum, Ripple, and Iota. *Wiley WIRES* 12, 1 (2022), e1436.

[5] Cuneyt G Akcora, Yitao Li, Yulia R Gel, and Murat Kantarcioglu. 2021. BitcoinHeist: topological data analysis for ransomware prediction on the bitcoin blockchain. In *Proceedings of the Twenty-Ninth International Conference on International Joint Conferences on Artificial Intelligence*. 4439–4445.

[6] Ismail Alarab, Simant Prakoonwit, and Mohamed Ikbal Nacer. 2020. Competence of graph convolutional networks for anti-money laundering in bitcoin blockchain. In *Proceedings of the 2020 5th International Conference on Machine Learning Technologies*. 23–27.

[7] Poupak Azad, Cuneyt Gurcan Akcora, and Arijit Khan. 2024. Machine Learning for Blockchain Data Analysis: Progress and Opportunities. *arXiv preprint arXiv:2404.18251* (2024).

[8] Gwern Branwen. 2023. Darknet Market Archives. Online. https://gwern.net/DNM-archives#grams

[9] Leo Breiman. 2001. Random forests. *Machine learning* 45 (2001), 5–32.

[10] Mauro Conti, Ankit Gangwal, and Sushmita Ruj. 2018. On the economic significance of ransomware campaigns: A Bitcoin transactions perspective. *Computers & Security* (2018).

[11] Federico Errica, Marco Podda, Davide Bacciu, and Alessio Micheli. 2020. A Fair Comparison of Graph Neural Networks for Graph Classification. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net. https://openreview.net/forum?id=HygDF6NFPB

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

[13] D. Y. Huang and D. McCoy. 2018. Tracking Ransomware End-to-end. In *Tracking Ransomware End-to-end*. IEEE, 1–12.

[14] Zhengjie Huang, Yunyang Huang, Peng Qian, Jianhai Chen, and Qinming He. 2023. Demystifying Bitcoin address behavior via graph neural networks. In *2023 IEEE 39th International Conference on Data Engineering (ICDE)*. IEEE, 1747–1760.

[15] Sesha Kethineni, Ying Cao, and Cassandra Dodge. 2018. Use of bitcoin in darknet markets: Examining facilitative factors on bitcoin-related crimes. *American Journal of Criminal Justice* 43 (2018), 141–157.

[16] Arijit Khan. 2022. Graph analysis of the ethereum blockchain data: A survey of datasets, methods, and future work. In *2022 IEEE International Conference on Blockchain (Blockchain)*. IEEE, 250–257.

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

[18] Stefan Kitzler, Friedhelm Victor, Pietro Saggese, and Bernhard Haslhofer. 2023. Disentangling decentralized finance (DeFi) compositions. *ACM Transactions on the Web* 17, 2 (2023), 1–26.

[19] Banwari Lal, Rachit Agarwal, and Sandeep Kumar Shukla. 2021. Understanding Money Trails of Suspicious Activities in a cryptocurrency-based Blockchain. *arXiv preprint arXiv:2108.11818* (2021).

[20] Serge Lang. 2012. *Algebra*. Vol. 211. Springer Science & Business Media.

[21] Chaehyeon Lee, Sajan Maharjan, Kyungchan Ko, and James Won-Ki Hong. 2019. Toward detecting illegal transactions on bitcoin using machine-learning methods. In *International Conference on Blockchain and Trustworthy Systems*. Springer, 520–533.

[22] Greg Maxwell. 2013. CoinJoin: Bitcoin privacy for the real world. In *Post on Bitcoin Forum*.

[23] S. Meiklejohn, M. Pomarole, G. Jordan, D. Levchenko, K.and McCoy, G. M Voelker, and S. Savage. 2013. A fistful of bitcoins: characterizing payments among men with no names. In *IMC*. ACM, 127–140.

[24] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon. 2002. Network motifs: Simple building blocks of complex networks. *Science* 298, 5594 (2002), 824–827.

[25] Satoshi Nakamoto. 2008. Bitcoin: A peer-to-peer electronic cash system.

[26] Lihao Nan and Dacheng Tao. 2018. Bitcoin mixing detection using deep autoencoder. In *2018 IEEE Third international conference on data science in cyberspace (DSC)*. IEEE, 280–287.

[27] Masarah Paquet-Clouston, Bernhard Haslhofer, and Benoit Dupont. 2018. Ransomware payments in the bitcoin ecosystem. *arXiv preprint arXiv:1804.04080* (2018).

[28] Aldo Pareja, Giacomo Domeniconi, Jie Chen, Tengfei Ma, Toyotaro Suzumura, Hiroki Kanezashi, Tim Kaler, Tao Schardl, and Charles Leiserson. 2020. EvolveGCN: Evolving Graph Convolutional Networks for Dynamic Graphs. In *Proceedings of the AAAI Conference on Artificial Intelligence*. 5363–5370.

[29] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in Python. *the Journal of machine Learning research* 12 (2011), 2825–2830.

[30] Thai Pham and Steven Lee. 2016. Anomaly detection in bitcoin network using unsupervised learning methods. In *arXiv preprint arXiv:1611.03941*.

[31] Farimah Poursafaei, Reihaneh Rabbany, and Zeljko Zilic. 2021. Sigtran: signature vectors for detecting illicit activities in blockchain transaction networks. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 27–39.

[32] Stephen Ranshous, Cliff A Joslyn, Sean Kreyling, Kathleen Nowak, Nagiza F Samatova, Curtis L West, and Samuel Winters. 2017. Exchange pattern mining in the bitcoin transaction directed hypergraph. In *Financial Cryptography and Data Security: FC 2017 International Workshops, WAHC, BITCOIN, VOTING, WTSC, and TA, Sliema, Malta, April 7, 2017, Revised Selected Papers 21*. Springer, 248–263.

[33] Abdullah Mueen Sheng Zhong. 2024. BitLINK: Temporal Linkage of Address Clusters in Bitcoin Blockchain. (2024).

[34] Dylan Vassallo, Vincent Vella, and Joshua Ellul. 2021. Application of gradient boosting algorithms for anti-money laundering in cryptocurrencies. In *SN Computer Science*, Vol. 2. Springer, 1–15.

[35] 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. In *ACM SIGKDD International Workshop on Knowledge discovery and data mining*.

[36] 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).

[37] Zhiying Wu, Jieli Liu, Jiajing Wu, Zibin Zheng, Xiapu Luo, and Ting Chen. 2023. Know your transactions: Real-time and generic transaction semantic representation on blockchain & Web3 ecosystem. In *Proceedings of the ACM Web Conference 2023*. 1918–1927.

[38] Shunxin Xiao, Shiping Wang, Yuanfei Dai, and Wenzhong Guo. 2022. Graph neural networks in node classification: survey and evaluation. *Machine Vision and Applications* 33 (2022), 1–19.

[39] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2018. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826* (2018).

[40] Muhan Zhang, Zhicheng Cui, Marion Neumann, and Yixin Chen. 2018. An end-to-end deep learning architecture for graph classification. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 32.

Appendix

In this part, we give further details of our method and experiments. In Section A.1, chainlet orbits are visualized and described. InAppendix B, we give the proof of our theorem stated in Section 4.2. In Section C, we explain the coin-mixing behavior and give our related orbits. In Section D, we describe how orbits can be applied to other UTXO blockchains.

A Chainlets

A.1 UTXO network and Chainlets

In the context of the Bitcoin blockchain, a transaction is formally defined in terms of its input and output as follows: Input: A set of one or more digital signatures, or "unspent transaction outputs" (UTXOs), that authorize the transfer of value from the sender's Bitcoin address to the recipient's address. UTXOs are previously-created outputs of transactions that were not spent yet and they are used as an input in a new transaction. Output: A set of one or more new UTXOs that represent the transfer of value from the sender's address to the recipient's address. In other words, a Bitcoin transaction takes one or more UTXOs as input and creates one or more new UTXOs as output. The total value of the inputs must be greater than or equal to the total value of the outputs, otherwise the transaction will be considered invalid.

Figure 8, 9, 10, 11 list all orbits and define address roles in them by using 2-chainlets. We use the X-M-N notations to describe chainlets where the first chainlet has X inputs, M outputs whereas the second chainlet has N outputs. In orbits, we are particularly interested in M and N.

A.2 Account Network and Chainlets

Unlike Bitcoin's UTXO model, Ethereum uses an account-based model where transactions involve two addresses: one input address and one output address. Therefore, the concept of chainlets needs to be adapted to this model where 2-chainlets are not naturally defined. We define chainlet orbits for Ethereum by focusing on the temporal transaction patterns within the network.

Specifically, we look at the k=3-hop neighborhoods of nodes (accounts) in the Ethereum transaction graph to capture local substructures. This choice allows us to both apply the vanilla orbit definitions to the account network without modification and also standardize orbit roles across UTXO and account-based blockchains.

In order to apply the UTXO orbit models to the Ethereum network, we employ ordinary nodes as anchors to serve a role similar to that of a UTXO transaction node. An anchor node is an address involved in a specific transaction pattern within the 3-hop neighborhood.

Definition 6 (Anchor 1). *An anchor 1 is an address node that is part of an orbit initiating at least one transaction such that its output is used in two subsequent transactions.*

Definition 7 (Anchor 2). *An anchor 2 node is an address node that is part of an orbit receiving coins from at least one other node and forwarding to at least one other address.*

We show two example anchors in Figure 7.

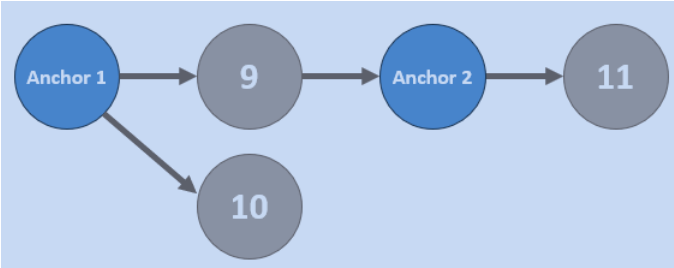


Figure 7: Orbit example in Ethereum network. Anchors of the figure act as the transaction nodes in Orbits 9, 10 and 11.

Once anchors 1 and 2 are defined, we extract 2-chainlets and compute orbits of address nodes as was done in Definition 3. By analyzing these orbit patterns, we effectively capture the dynamic transaction behaviors characteristic of account-based blockchain networks such as Ethereum (see Section E.1).

Limitations. We note that account-based transactions inherently involve simpler, direct transactions between two parties (one sender and one receiver). This simplicity restricts the complexity of the transaction graph and subsequently limits the depth of analysis that can be performed using orbits. Without the natural formation of complex chainlets as in UTXO models, it's challenging to capture rich, multi-step transaction patterns.

B Proof of the Theorem

In this part, we provide the proof of Theorem 4.1, relating the count of orbits with the number of their stabilizers. Let $\mathcal{I}(C_k)$ be the set of all isomorphisms between any two k-chainlets in $\mathcal{E}(C_k)$. $\Lambda(C_k)$ be the group of all automorphisms (self-isomorphisms) of C_k. Then, for any node u in C_k, we define the stabilizer subgroup of u in $\Lambda(C_k)$ as $\mathcal{S}(u) = \{\varphi \in \Lambda(C_k) | \varphi(u) = u\}$. Let $|K|$ represent the cardinality of the set K. Now, we recall our theorem.

Theorem B.1. *Let C_k be a k-chainlet in $\mathcal{G}$, and u be a node in C_k, and $I(C_k), \Lambda(C_k), S(u)$ be as defined above. Then, we have*

$$|\mathcal{I}(C_k)| = |\mathcal{E}(C_k)|^2 \cdot |\Lambda(C_k)| \quad \textit{and} \quad |O_u| = |\mathcal{E}(C_k)| \cdot \frac{|\Lambda(C_k)|}{|\mathcal{S}(u)|}$$

Proof. We start with proving the first equation $|\mathcal{I}(C_k)| = |\mathcal{E}(C_k)|^2 \cdot |\Lambda(C_k)|$. Let $|\mathcal{E}(C_k)| = m$ and $\mathcal{E}(C_k) = \{C_k^1, C_k^2, \ldots, C_k^m\}$ where $C_k = C_k^1$. For each C_k^i, fix $\varphi_i \in \mathcal{I}(C_k)$ such that $\varphi_i : C_k^1 \to C_k^i$. Let $\psi \in \mathcal{I}(C_k)$ be any isomorphism, i.e., $\psi : C_k^{j_1} \to C_k^{j_2}$. Then, $\widehat{\psi} = \varphi_{j_2}^{-1} \circ \psi \circ \varphi_{j_1}$ is an automorphism of C_k, i.e. $\widehat{\psi} : C_k \to C_k$. This implies $\widehat{\psi} \in \Lambda(C_k)$. Hence, by using fixed isomorphisms $\{\varphi_1, \ldots, \varphi_m\}$, we find a unique automorphism $\widehat{\psi} \in \Lambda(C_k)$ for any isomorphism $\psi \in \mathcal{I}(C_k)$. Notice that this process is reversible, i.e., for any $1 \le i \le j \le m$, and for any $\widehat{\psi} \in \Lambda(C_k)$, we obtain a unique isomorphism $\widetilde{\psi}_{ij} = \varphi_j \circ \psi \circ \varphi_i^{-1} : C_k^i \to C_k^j$. This implies $|\mathcal{I}(C_k)| = |\mathcal{E}(C_k)|^2 \cdot |\Lambda(C_k)|$. The first equation follows.

To prove the second equation $|O_u| = |\mathcal{E}(C_k)| \cdot \frac{|\Lambda(C_k)|}{|\mathcal{S}(u)|}$, we first show that $\frac{|\Lambda(C_k)|}{|\mathcal{S}(u)|}$ is equal to the number of similar nodes of u in C_k, i.e. $\frac{|\Lambda(C_k)|}{|\mathcal{S}(u)|} = |O_u^1| = |O_u \cap \mathcal{V}_{C_k^1}|$ where O_u^i represent the similar nodes to u belonging to C_k^i. Notice that $\Lambda(C_k)$ is a group acting on the set C_k. Hence, the size of the orbit of u in C_k^1 is equal to the cardinality of the quotient of $\Lambda(C_k)$ by the stabilizer subgroup $\mathcal{S}(u)$ by [20, Proposition 5.1]. In other words, O_u^1 has one to one correspondence with the quotient group $\Lambda(C_k)/\mathcal{S}(u)$ (# of cosets). Hence, if $O_u^1 = \{u_1, u_2, \ldots, u_r\}$, then $\Lambda(C_k)/\mathcal{S}(u) = \{\mathcal{S}(u)u_1, \ldots, \mathcal{S}(u)u_r\}$, i.e., $|O_u^1| = \frac{|\Lambda(C_k)|}{|\mathcal{S}(u)|}$.

Now, take any node $v \in O_u$. Assume $v \in C_k^i$ where $C_k^i \in \mathcal{E}(C_k)$. By definition of the whole orbit O_u, there exists an isomorphism $\psi : C_k^1 \to C_k^i$ with $\psi(u) = v$. Since $\varphi_i^{-1} \circ \psi \in \Lambda(C_k)$, $\varphi_i^{-1}(v) \in O_u^1$. This proves there is a one-to-one correspondence between the sets $O_u^1 \times \{\varphi_1, \ldots, \varphi_m\}$ and O_u. In other words, $O_u = \bigcup_{i=1}^m O_u^i$ where O_u^i is the orbit of u in C_k^i. The proof follows. □

C Orbits in E-Crime

It's important to note that while e-crime transactions can be studied for pattern identification and detection, they are illegal activities and have serious ethical implications. Our analysis of coin-mixing and ransomware transactions below aims to contribute to cybersecurity efforts and the understanding of criminal behaviors in the digital realm.

C.1 Orbits in Coin-Mixing Transactions

A coin-mixing transaction, also known as a coin-join transaction [22], is a technique used to enhance the privacy and anonymity of cryptocurrency transactions. In a typical coin-mixing transaction, multiple participants combine their coins or funds into a single transaction, making it difficult to trace the original source of the funds.

A coin-mixing transaction works in the following way: Several individuals who wish to improve the privacy of their transactions agree to participate in a coin-mixing process. Each participant selects a specific amount of their cryptocurrency holdings (coins) to contribute to the coin-mixing transaction. These individual inputs are usually of equal value to obfuscate the original ownership. All the selected inputs from the participants are combined into a mixing pool or a single transaction. This process effectively mixes the coins together, making it challenging to identify which coins belong to each participant. The mixing pool generates new outputs or addresses that receive the mixed coins. The number of outputs is usually the same as the number of inputs to maintain balance. The newly mixed coins are then distributed back to the participants, but each participant receives their coins in a randomized manner. This step further obscures the link between the original inputs and the final outputs.

The mixing process is repeated multiple times to enhance privacy. It's worth noting that while coin-mixing transactions enhance privacy, they can also be associated with illicit activities if used for money laundering or other illegal purposes.

In graph terms, the coin mixing transactions create 2-chainlets with many inputs and outputs as shown in Figure 6b. We identify several orbits that may encode coin mixing transactions. These are orbits 30, 31, 32, 39, 40, 41, 46, and 47 in Figure 10 and Figure 11.

C.2 Orbits in Ransomware Transactions

A ransomware transaction involves the malicious encryption of a victim's digital files or data by cybercriminals. The victim is then required to pay a ransom, usually in cryptocurrency, to obtain the decryption key and regain access to their files. Ransomware attacks have become a significant concern in the realm of cybersecurity [27].

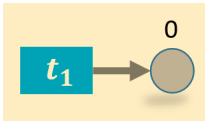
A ransomware transaction typically follows these steps. Cybercriminals exploit vulnerabilities to gain unauthorized access to a victim's system. They then encrypt the victim's files, rendering them inaccessible. The attackers demand a ransom from the victim, usually in cryptocurrency, in exchange for the decryption key. If the victim agrees to pay the ransom, they transfer the specified amount of

cryptocurrency to the attacker's address. After receiving the payment, the attackers provide the victim with the decryption key, allowing them to regain access to their files. Ransomware attacks often involve cryptocurrencies due to their pseudonymous nature, making it difficult to trace the flow of funds and the identities of the attackers. Ransomware transactions can also be represented graphically using orbits. In this context, a ransomware transaction would create a 2-chainlet with a specific set of inputs and outputs. These 2-chainlets can be identified as orbits within the larger graph of transactions. We identify certain orbits, such as orbits 30, 32, 39, and 41, within ransomware transactions. These orbits indicate the presence of specific patterns associated with ransomware activities.

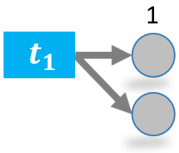
D Applicability of Orbits to Monero and Zcash

The concept of orbits can be applied to heterogeneous graphs like the UTXO transaction networks that consist of transaction and address nodes. In the case of Monero and ZCash, which both utilize the UTXO data model [4], orbits can be directly applied to the unshielded transactions pool of ZCash without any modifications.

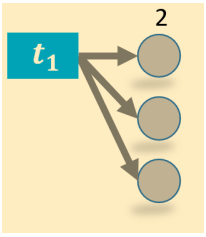
In Monero, the association between addresses and transactions is concealed in all transactions, but the input and output counts are not hidden. With orbits defined for both input and output addresses of transactions, these counts can be tracked. However, instead of representing addresses as nodes, output hash IDs would be represented as nodes in this context.



(a) Orbit 0: if $|\Gamma_1^o| = 1$ and $a \in \Gamma_1^o$ then $a \in o_0$.

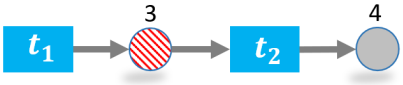


(b) Orbit 1: if $|\Gamma_1^o| = 2$ and $a \in \Gamma_1^o$ then $a \in o_1$.

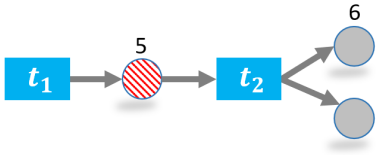


(c) Orbit 2: if $|\Gamma_1^o| = 3$ and $a \in \Gamma_1^o$ then $a \in o_2$.

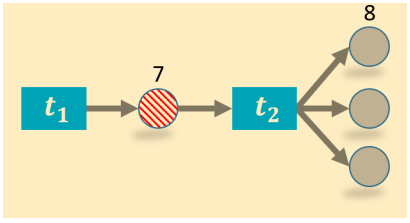
Figure 8: Orbits 0 to 2 involve addresses who do not create a 2-chainlet for the daily network snapshot. These addresses keep the received coins dormant for the given day. All addresses have passive orbits.



(d) Orbit 3: if $|\Gamma_1^o| = 1, |\Gamma_2^i| = 1, |\Gamma_2^o| = 1$ and $a \in \Gamma_1^o$, $a \in \Gamma_2^i$ then $a \in o_3$. Orbit 4: if $|\Gamma_1^o| = 1, |\Gamma_2^i| = 1, |\Gamma_2^o| = 1$ and $a \in \Gamma_2^o$, then $a \in o_4$.



(e) Orbit 5: if $|\Gamma_1^o| = 1, |\Gamma_2^i| = 1, |\Gamma_2^o| = 2$, and $a \in \Gamma_1^o$, $a \in \Gamma_2^i$ then $a \in o_5$. Orbit 6: if $|\Gamma_1^o| = 1, |\Gamma_2^i| = 1, |\Gamma_2^o| = 2$, and $a \in \Gamma_2^o$ then $a \in o_6$.



(f) Orbit 7: if $|\Gamma_1^o| = 1, |\Gamma_2^i| = 1, |\Gamma_2^o| = 3$, and $a \in \Gamma_1^o$, $a \in \Gamma_2^i$ then $a \in o7$. Orbit 8: if $|\Gamma_1^o| = 1, |\Gamma_2^i| = 1, |\Gamma_2^o| = 3$, and $a \in \Gamma_2^o$ then $a \in o8$.

Figure 9: X-1-N addresses create orbits 3 to 8 where the first transaction has a single output address and the second transaction receives coins from the address. Red addresses have active orbits, whereas gray addresses have passive orbits.

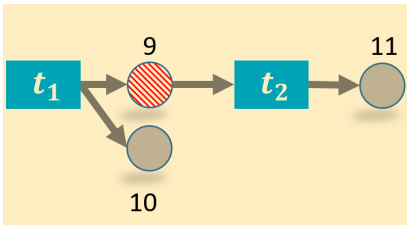
E Additional Classification Experiments

In Section 6, we report the average accuracy of five runs along with their standard deviation, using an 80/20 training and test split. For the BitcoinHeist data experiments, we utilize binary classification (white/ransomware) and a Random Forest with 300 trees. In experiments on the full orbit data, we employ a one-versus-rest Random Forest classifier.

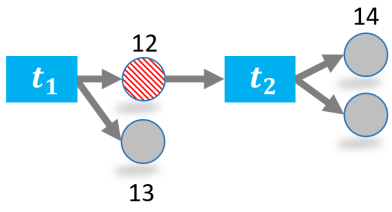
Classifiers and Hyperparams

With orbits, we use `OneVsRestClassifier` of `RandomForestClassifier` of the Scikit library [29]. The default parameters of the `sklearn.ensemble` `RandomForestClassifier` are as follows: the model used is a OneVsRestClassifier employing a RandomForestClassifier.

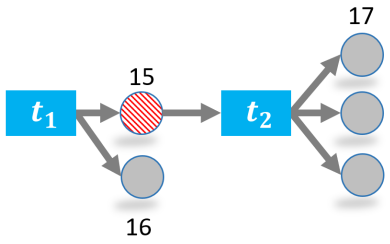
In Section 6, we report the average accuracy from five runs, along with their standard deviation, using an 80/20 training and test split. For the BitcoinHeist data experiments, we apply binary classification (white/ransomware) using a RandomForest with 300 trees. Additionally, we



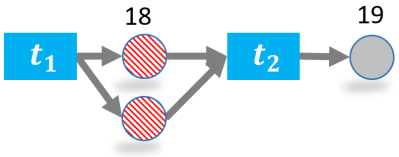
(g) Orbit 9: if $|\Gamma_1^o| = 2, |\Gamma_2^i| = 1, |\Gamma_2^o| = 1, a \in \Gamma_1^o, a \in \Gamma_2^i$ **then** $a \in$ o9. **Orbit 10: if** $|\Gamma_1^o| = 2, |\Gamma_2^i| = 1, |\Gamma_2^o| = 1, a \in \Gamma_1^o, a \notin \Gamma_2^i$ **then** $a \in$ o10. **Orbit 11: if** $|\Gamma_1^o| = 2, |\Gamma_2^i| = 1, |\Gamma_2^o| = 1, a \in \Gamma_2^o$ **then** $a \in$ o11.



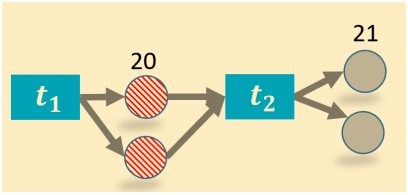
(h) Orbit 12: if $|\Gamma_1^o| = 2, |\Gamma_2^i| = 1, |\Gamma_2^o| = 2, a \in \Gamma_1^o, a \in \Gamma_2^i$ **then** $a \in$ o12. **Orbit 13: if** $|\Gamma_1^o| = 2, |\Gamma_2^i| = 1, |\Gamma_2^o| = 2, a \in \Gamma_1^o, a \notin \Gamma_2^i$ **then** $a \in$ o13. **Orbit 14: if** $|\Gamma_1^o| = 2, |\Gamma_2^i| = 1, |\Gamma_2^o| = 2, a \in \Gamma_2^o$ **then** $a \in$ o14.



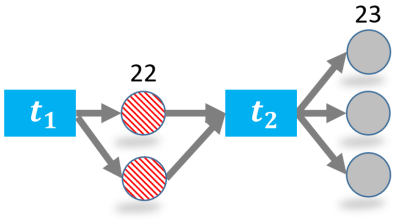
(i) Orbit 15: if $|\Gamma_1^o| = 2, |\Gamma_2^i| = 1, |\Gamma_2^o| = 3, a \in \Gamma_1^o, a \in \Gamma_2^i$ **then** $a \in$ o15. **Orbit 16: if** $|\Gamma_1^o| = 2, |\Gamma_2^i| = 1, |\Gamma_2^o| = 3, a \in \Gamma_1^o, a \notin \Gamma_2^i$ **then** $a \in$ o16. **Orbit 17: if** $|\Gamma_1^o| = 2, |\Gamma_2^i| = 1, |\Gamma_2^o| = 3, a \in \Gamma_2^o$ **then** $a \in$ o17.



(j) Orbit 18: if $|\Gamma_1^o| = 2, |\Gamma_2^i| = 2, |\Gamma_2^o| = 1, a \in \Gamma_1^o, a \in \Gamma_2^i$ **then** $a \in$ o18. **Orbit 19: if** $|\Gamma_1^o| = 2, |\Gamma_2^i| = 2, |\Gamma_2^o| = 1, a \in \Gamma_2^o$, **then** $a \in$ o19.



(k) Orbit 20: if $|\Gamma_1^o| = 2, |\Gamma_2^i| = 2, |\Gamma_2^o| = 2, a \in \Gamma_1^o, a \in \Gamma_2^i$ **then** $a \in$ o20. **Orbit 21: if** $|\Gamma_1^o| = 2, |\Gamma_2^i| = 2, |\Gamma_2^o| = 2, a \in \Gamma_2^o$, **then** $a \in$ o21.



(l) Orbit 22: if $|\Gamma_1^o| = 2, |\Gamma_2^i| = 2, |\Gamma_2^o| = 3, a \in \Gamma_1^o, a \in \Gamma_2^i$ **then** $a \in$ o22. **Orbit 23: if** $|\Gamma_1^o| = 2, |\Gamma_2^i| = 2, |\Gamma_2^o| = 3, a \in \Gamma_2^o$, **then** $a \in$ o23.

Figure 10: X-2-N create orbits 9 to 23 where the first transaction has two output addresses and the second transaction receives coins from one or two of the output addresses of the first transaction. Red striped addresses have active orbits, whereas gray addresses have passive orbits.

use a one-versus-rest RandomForest classifier from the Scikit library's `RandomForestClassifier` [29] for the three-way classification and a test size of 0.2.

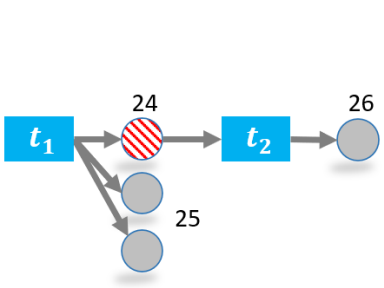
E.1 Ethereum Results

We train an address classifier on Ethereum for four types of DeFi protocols: lending, assets, derivatives, and decentralized exchanges, as well as a class for ordinary ('normal') addresses. The classification problem is significant because new protocols are introduced on the Ethereum blockchain daily, and existing protocols create new addresses. Users and blockchain data analytics companies scour offline resources to find the entity behind an address with the goal of tracking investment and portfolio decisions, detecting transaction anomalies as well as identifying smart contract hacking attempts. Automated address classification alleviates the need for much manual work. Hence, address type classification has appeared as an important, modern and challenging task in Graph Machine Learning [33].

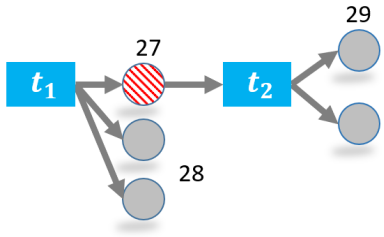
It is important to note that Ethereum addresses, whether they belong to smart contracts or externally owned accounts, share the same format: a 42-character hexadecimal string beginning with "0x." This uniformity means that, when looking at the address string alone, there is no inherent way to differentiate between an address that represents a smart contract and one that represents an externally owned account. The distinction between these two types of addresses is critical in understanding their behavior and capabilities on the Ethereum network, as smart contracts are autonomous programs with code execution capabilities, while externally owned accounts are controlled by private keys and are responsible for initiating transactions. Without additional context or metadata, simply examining the address string does not reveal this information. Protocol and normal addresses can belong to any of the two address types.

We selected the daily graph of September 20, 2021 because it contained the highest number of protocol addresses in ether transactions. However, even for the day, the data imbalance is quite high: only 89 protocol addresses appeared in over 1M ether transactions. As a result, we use a down-sampling strategy for the 'normal' addresses. Furthermore, we were constrained to use a binary classification scheme due to the limited dataset of protocol addresses: the classification was limited to distinguishing between protocol and non-protocol addresses only.

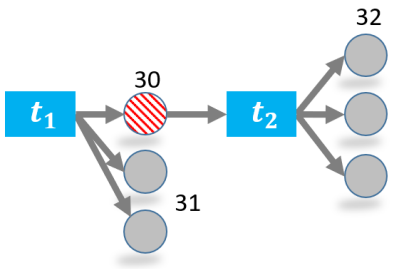
We use a random forest algorithm with 300 trees on orbits, with the most important ones being orbits 0, 1, 2, 8, and 29 as identified by the Gini importance measure [9]. These orbits are all passive orbits that are used to receive coin payments from users. The orbit model achieves an accuracy of 0.97 and as shown in Figure 12 an Area Under the Curve of 0.86. As the figure shows, adding more 'ordinary' nodes first improves, then deteriorate the classifier performance. These results are promising, yet the collection of additional labeled data, specifically unique addresses, is essential for further validation.



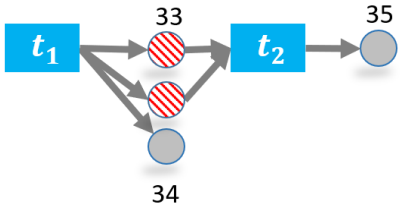
(m) Orbit 24: if $|\Gamma_1^o| = 3, |\Gamma_2^i| = 1, |\Gamma_2^o| = 1, a \in \Gamma_1^o, a \in \Gamma_2^i$ **then** $a \in o24$. **Orbit 25: if** $|\Gamma_1^o| = 3, |\Gamma_2^i| = 1, |\Gamma_2^o| = 1, a \in \Gamma_1^o, a \notin \Gamma_2^i$ **then** $a \in o25$. **Orbit 26: if** $|\Gamma_1^o| = 3, |\Gamma_2^i| = 1, |\Gamma_2^o| = 1, a \in \Gamma_2^o$, **then** $a \in o26$.



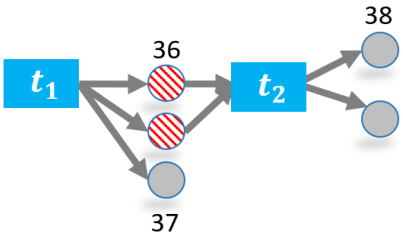
(n) Orbit 27: if $|\Gamma_1^o| = 3, |\Gamma_2^i| = 1, |\Gamma_2^o| = 2, a \in \Gamma_1^o, a \in \Gamma_2^i$ **then** $a \in o27$. **Orbit 28: if** $|\Gamma_1^o| = 3, |\Gamma_2^i| = 1, |\Gamma_2^o| = 2, a \in \Gamma_1^o, a \notin \Gamma_2^i$ **then** $a \in o28$. **Orbit 29: if** $|\Gamma_1^o| = 3, |\Gamma_2^i| = 1, |\Gamma_2^o| = 2, a \in \Gamma_2^o$ **then** $a \in o29$.



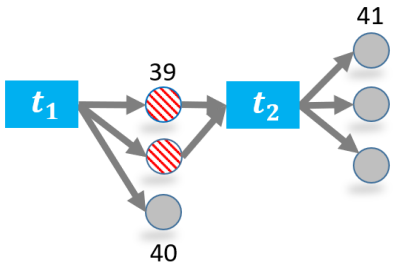
(o) Orbit 30: if $|\Gamma_1^o| = 3, |\Gamma_2^i| = 1, |\Gamma_2^o| = 3, a \in \Gamma_1^o, a \in \Gamma_2^i$ **then** $a \in o30$. **Orbit 31: if** $|\Gamma_1^o| = 3, |\Gamma_2^i| = 1, |\Gamma_2^o| = 3, a \in \Gamma_1^o, a \notin \Gamma_2^i$ **then** $a \in o31$. **Orbit 32: if** $|\Gamma_1^o| = 3, |\Gamma_2^i| = 1, |\Gamma_2^o| = 3, a \in \Gamma_2^o$ **then** $a \in o32$.



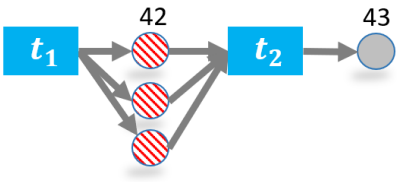
(p) Orbit 33: if $|\Gamma_1^o| = 3, |\Gamma_2^i| = 2, |\Gamma_2^o| = 1, a \in \Gamma_1^o, a \in \Gamma_2^i$ **then** $a \in o33$. **Orbit 34: if** $|\Gamma_1^o| = 3, |\Gamma_2^i| = 2, |\Gamma_2^o| = 1, a \in \Gamma_1^o, a \notin \Gamma_2^i$ **then** $a \in o34$. **Orbit 35: if** $|\Gamma_1^o| = 3, |\Gamma_2^i| = 2, |\Gamma_2^o| = 1, a \in \Gamma_2^o$ **then** $a \in o35$.



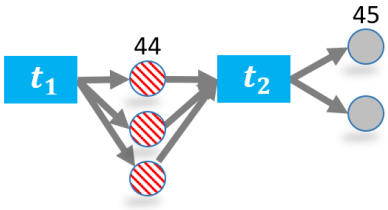
(r) Orbit 36: if $|\Gamma_1^o| = 3, |\Gamma_2^i| = 2, |\Gamma_2^o| = 2, a \in \Gamma_1^o, a \in \Gamma_2^i$ **then** $a \in o36$. **Orbit 37: if** $|\Gamma_1^o| = 3, |\Gamma_2^i| = 2, |\Gamma_2^o| = 2, a \in \Gamma_1^o, a \notin \Gamma_2^i$ **then** $a \in o37$. **Orbit 38: if** $|\Gamma_1^o| = 3, |\Gamma_2^i| = 2, |\Gamma_2^o| = 2, a \in \Gamma_2^o$ **then** $a \in o38$.



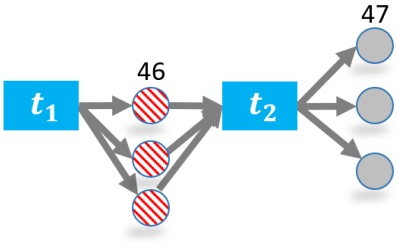
(s) Orbit 39: if $|\Gamma_1^o| = 3, |\Gamma_2^i| = 2, |\Gamma_2^o| = 3, a \in \Gamma_1^o, a \in \Gamma_2^i$ **then** $a \in o39$. **Orbit 40: if** $|\Gamma_1^o| = 3, |\Gamma_2^i| = 2, |\Gamma_2^o| = 3, a \in \Gamma_1^o, a \notin \Gamma_2^i$ **then** $a \in o40$. **Orbit 41: if** $|\Gamma_1^o| = 3, |\Gamma_2^i| = 2, |\Gamma_2^o| = 3, a \in \Gamma_2^o$ **then** $a \in o41$.



(t) Orbit 42: if $|\Gamma_1^o| = 3, |\Gamma_2^i| = 3, |\Gamma_2^o| = 1, a \in \Gamma_1^o, a \in \Gamma_2^i$ **then** $a \in o42$. **Orbit 43: if** $|\Gamma_1^o| = 3, |\Gamma_2^i| = 3, |\Gamma_2^o| = 1, a \in \Gamma_2^o$ **then** $a \in o43$.



(u) Orbit 44: if $|\Gamma_1^o| = 3, |\Gamma_2^i| = 3, |\Gamma_2^o| = 2, a \in \Gamma_1^o, a \in \Gamma_2^i$ **then** $a \in o44$. **Orbit 45: if** $|\Gamma_1^o| = 3, |\Gamma_2^i| = 3, |\Gamma_2^o| = 2, a \in \Gamma_2^o$ **then** $a \in o45$.



(v) Orbit 46: if $|\Gamma_1^o| = 3, |\Gamma_2^i| = 3, |\Gamma_2^o| = 3, a \in \Gamma_1^o, a \in \Gamma_2^i$ **then** $a \in o46$. **Orbit 47: if** $|\Gamma_1^o| = 3, |\Gamma_2^i| = 3, |\Gamma_2^o| = 3, a \in \Gamma_2^o$ **then** $a \in o47$.

Figure 11: X-3-N create orbits 24 to 47 where the first transaction has three output addresses and the second transaction receives coins from one, two or three of the output addresses of the first transaction. Red striped addresses have active orbits, whereas gray addresses have passive orbits.

E.2 Litecoin Results

Due to the absence of labeled address data, our analysis of Litecoin is confined to orbit discovery and the examination of orbit characteristics. This analysis is crucial to demonstrate that useful orbits can be identified on non-Bitcoin blockchains.

Litecoin differs from Bitcoin in certain aspects such as block generation time (2.5 minutes on Litecoin vs 10 minutes on Bitcoin), hashing algorithm, and transaction volume. However Litecoin shares the same UTXO transaction structure with Bitcoin, as such, orbits are applicable without any modifications.

A notable characteristic of Litecoin transactions is their tendency to contain fewer address nodes compared to Bitcoin transactions. This results in a prevalence of simpler transaction patterns, typically involving one-to-one or one-to-two coin transfers as we show in Figure 13b. This simplicity can be attributed to Litecoin's intended use as a faster and more efficient medium for smaller, everyday transactions. Unlike

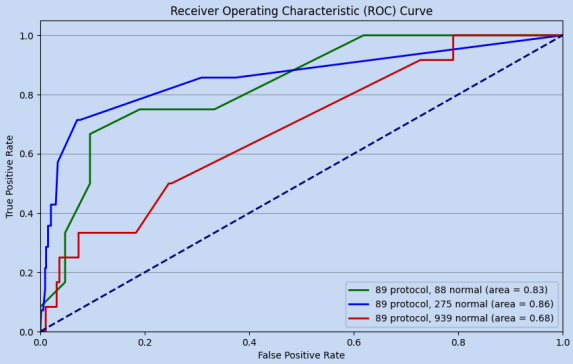
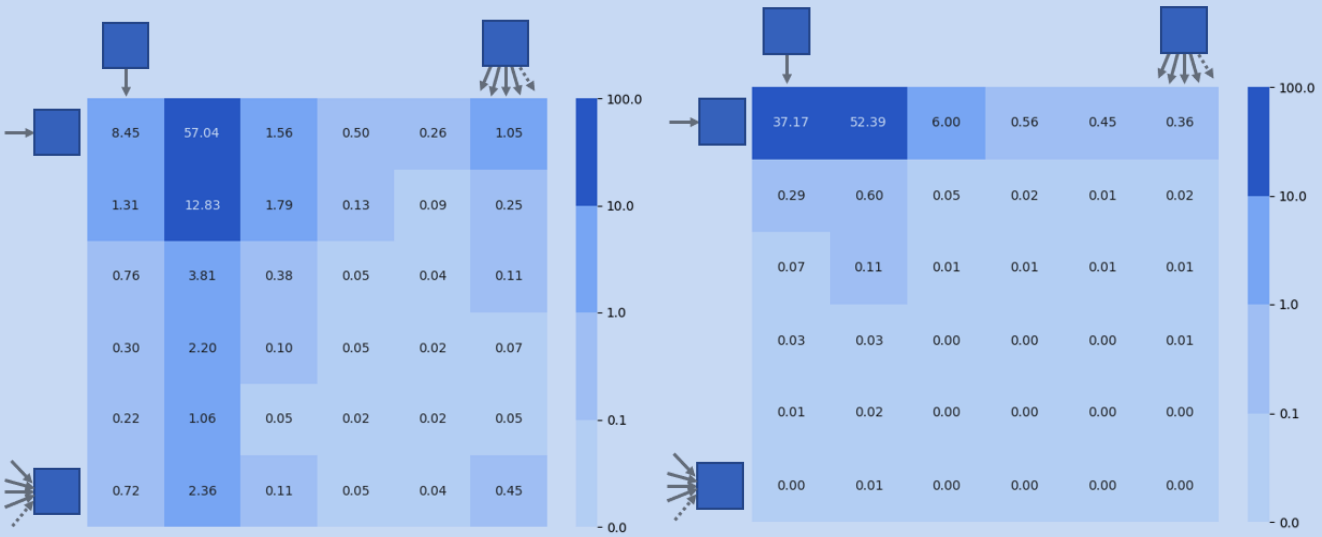


Figure 12: Summary of Model Performance Metrics and Dataset Composition. The orbit model achieves an accuracy of 0.97 **and an Area Under the Curve (AUC) of** 0.86**. The dataset consists of** 275 **ordinary address nodes and** 89 **address nodes labeled as 'Protocol'.**



(a) 1-Chainlet occurrence matrix of Bitcoin.

(b) 1-Chainlet occurrence matrix of Litecoin.

Figure 13: Comparative analysis of 1-chainlet occurrence matrices for Bitcoin and Litecoin. In the figure rows represent the number of input addresses whereas columns represent the number of output addresses in transactions. Darker colors indicate higher percentages. As the figure shows, 57.04% **of all Bitcoin transactions have one input and two output addresses. On Litecoin,** 90.45% **of all transactions have** 2 **or less inputs and outputs (the four top-left cells).**

Bitcoin, which is often used for larger transfers and is more subject to complex multi-input and multi-output transactions, Litecoin's transaction patterns are designed for speed and cost-efficiency.

The dominance of one-to-one and one-to-two transactions in Litecoin means that the complex substructures and patterns identified in Bitcoin's transaction graph are less prevalent. Specifically, the orbits 0—6, 9—14, 18—21 are prevalent.

The core principle of chainlet orbits is to capture the structural roles of addresses within the transaction graph. Even in the Litecoin network dominated by one-to-one and one-to-two transactions, these structural roles can reveal significant insights. One-to-one and one-to-two transactions, while simpler, can still exhibit consistent patterns that are indicative of certain behaviors. For example, frequent one-to-one transactions from a single address may suggest automated micro-payments or faucet activities, which can be significant for network analysis. The uniformity of transactions in Litecoin makes anomalies stand out more prominently. Chainlet orbits can effectively

identify deviations from the norm, such as unexpected multi-output transactions or irregular transaction timings, which may signal e-crime activities or attempts to obfuscate transaction trails.