MPOCryptoML: Multi-Pattern based Off-Chain Crypto Money Laundering Detection

Yasaman Samadi, Hai Dong, *Senior Member, IEEE,* Xiaoyu Xia

***Abstract*—Recent advancements in money laundering detection have demonstrated the potential of using graph neural networks to capture laundering patterns accurately. However, existing models are not explicitly designed to detect the diverse patterns of off-chain cryptocurrency money laundering. Neglecting any laundering pattern introduces critical detection gaps, as each pattern reflects unique transactional structures that facilitate the obfuscation of illicit fund origins and movements. Failure to account for these patterns may result in under-detection or omission of specific laundering activities, diminishing model accuracy and allowing schemes to bypass detection. To address this gap, we propose the *MPOCryptoML* model to effectively detect multiple laundering patterns in cryptocurrency transactions. MPOCryptoML includes the development of a multi-source Personalized PageRank algorithm to identify random laundering patterns. Additionally, we introduce two novel algorithms by analyzing the timestamp and weight of transactions in high-volume financial networks to detect various money laundering structures, including fan-in, fan-out, bipartite, gather-scatter, and stack patterns. We further examine correlations between these patterns using a logistic regression model. An anomaly score function integrates results from each module to rank accounts by anomaly score, systematically identifying high-risk accounts. Extensive experiments on public datasets including Elliptic++, Ethereum fraud detection, and Wormhole transaction datasets validate the efficacy and efficiency of MPOCryptoML. Results show consistent performance gains, with improvements up to 9.13% in precision, up to 10.16% in recall, up to 7.63% in F1-score, and up to 10.19% in accuracy.**

***Index Terms*—Off-chain crypto money laundering, multi-pattern detection, graph anomaly detection.**

I. Introduction

CRYPTO money laundering involves disguising the origins of illegally obtained cryptocurrency to make it appear legitimate [1]. This process leverages complex transaction chains and obfuscation strategies, often exploiting the decentralized, borderless, and pseudonymous nature of blockchain technologies [2]. As cryptocurrency adoption accelerates across financial ecosystems, so does its abuse for illicit purposes, including money laundering, fraud, and sanctions evasion. The global financial and legal implications are profound, necessitating advanced approaches to track and disrupt these activities at scale. A typical example of a crypto money laundering process is illustrated in Figure 2, where layered transactions obscure the flow from source to destination addresses.

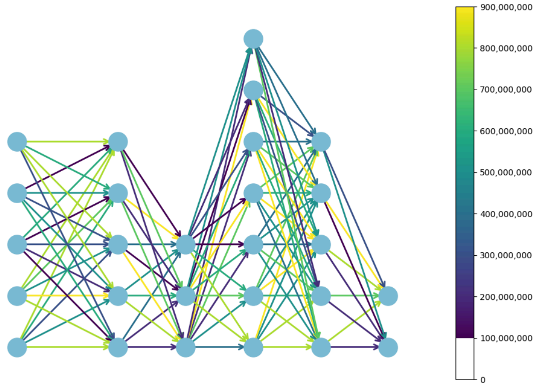


Fig. 1. An illustration of cryptocurrency money laundering, which is a series of digital wallet transactions resulting in a complex, multi-pattern subgraph. Sources on the left are used to initiate the money laundering process, which is then carried out through a series of digital wallet stages in the middle to conceal the money's origin and destinations on the right. The amount of money transferred is indicated by the color of the edge.

While crypto laundering retains the classical stages of placement, layering, and integration, it departs significantly from traditional bank-based laundering in both execution and traceability. Traditional anti money laundering (AML) frameworks rely heavily on centralized oversight, customer verification, and transaction monitoring, typically confined to one jurisdiction. In contrast, cryptocurrency networks are decentralized, span multiple regulatory zones, and operate pseudonymously [3]. Furthermore, crypto transactions can be automated and executed at scale using scripts and bots, with minimal barriers to entry. These features fundamentally undermine the effectiveness of conventional banking AML mechanisms, rendering many legacy solutions obsolete in crypto contexts.

Crypto transactions can occur both *"on-chain"* (visible and verifiable on the blockchain ledger) and *"off-chain"* (occurring outside the blockchain, often through centralized exchanges, peer-to-peer platforms, or fiat conversion mechanisms). While "on-chain" activities provide transparency and immutability, "off-chain" transactions remain opaque, fragmented, and jurisdictionally complex. "Off-chain" laundering lacks standardized monitoring, making it harder to detect than its "on-chain" counterpart. Due to these unique characteristics, "off-chain" laundering often becomes the gateway for illicit crypto assets to re-enter the legitimate financial system.

Existing crypto AML tools primarily target "on-chain" activities using blockchain analysis, anomaly detection, and graph-based learning models [4], [5]. However, these methods struggle with obfuscation techniques, lack visibility into centralized platforms, and are largely ineffective against laundering strategies that span "off-chain" domains. Conventional

All authors are with the School of Computing Technologies, RMIT University, Melbourne, VIC, Australia.

Manuscript received July 10, 2025.

models often depend on static laundering patterns or transaction thresholds and fail to adapt to evolving laundering tactics, especially in environments without direct blockchain traceability. Moreover, most prior works focus on wallet security [6], protocol-level vulnerabilities, or generalized fraud detection without addressing the nuanced nature of "off-chain" laundering.

To address these challenges, we propose **MPOCryptoML**, a novel "off-chain" money laundering detection framework. Our approach models laundering behaviors as multi-source heterogeneous graphs and applies a multi-stage detection architecture. It begins by using a multi-source Personalized PageRank (PPR) algorithm to trace random laundering flows. Then, it applies temporal, structural, and transactional features captured via a Timestamp algorithm, a Weight algorithm, and a Logistic Regression model to identify suspicious patterns such as fan-in/fan-out, bipartite, stack, and gather-scatter. Lastly, an anomaly scoring function aggregates and ranks entities by laundering likelihood.

This paper makes the following contributions:

- We present the first end-to-end method, MPOCryptoML, tailored for identifying multiple laundering patterns in "off-chain" cryptocurrency laundering, bridging a critical research gap.
- We develop a multi-source Personalized PageRank (PPR) algorithm that captures hidden and random laundering paths in cross-platform transaction graphs.
- We introduce a multi-pattern laundering detection algorithm capable of identifying complex transaction patterns such as fan-in/fan-out, bipartite, gather-scatter, and stack formations.
- We design a novel anomaly score function for ranking laundering suspects based on multiple behavioral and topological features in graph space.
- We demonstrate the scalability and effectiveness of our approach on real-world datasets from Ethereum fraud detection, Wormhole, and Elliptic++, achieving significant improvements of up to 9.13% in precision, 10.16% in recall, 7.63% in F1-score, and 10.19% in accuracy, compared to seven state-of-the-art baselines.

The rest of this paper is structured as follows: Section II provides an extensive review of related works, while Section III covers preliminaries on "off-chain" crypto money laundering workflow and patterns. Section IV introduces the problem formulation, followed by Section V, which details *MPOCryptoML* architecture. Section VI presents the experiments, results, and an ablation study to evaluate the model's performance. Section VII discusses the application scenarios. Finally, Section VIII concludes the paper with key insights and future directions.

II. Related Work

In this section, we review various methods for detecting anomalies and outliers, as well as identifying patterns of money laundering.

Anomaly Detection Approaches: Anomaly detection aims to identify data instances that deviate significantly from established normal patterns. Given the scarcity of labeled data in real-world scenarios, unsupervised and semi-supervised methods are widely employed. Unsupervised techniques encompass a range of strategies, each based on different assumptions about the nature of anomalies. Isolation-based methods (e.g., Isolation Forest [7]) detect anomalies by recursively partitioning data and isolating points that require fewer splits, assuming that anomalies are few and different. Density-based methods (e.g., Local Outlier Factor [8]) consider instances in sparse regions as anomalies, relying on the assumption that normal points lie in dense clusters. Distance-based methods (e.g., Deep SVDD [9]) measure how far a point deviates from others or a learned center, identifying distant points as anomalies. Probability-based approaches (e.g., ECOD [10]) model the underlying data distribution and flag instances with low probability density as outliers. Reconstruction-based methods (e.g., Deep Autoencoding Gaussian Mixture Model [11]) train models to reconstruct normal instances accurately, and instances with high reconstruction error are treated as anomalies. Despite their flexibility, these unsupervised methods often suffer from high false positive rates due to the absence of ground truth and variability in real-world data.

To address these limitations, semi-supervised methods have emerged as a promising alternative by incorporating limited labeled data to guide the anomaly detection process. For example, positive-unlabeled (PU) learning [12], [13] assumes that labeled anomalies are rare and attempts to identify outliers from a large pool of unlabeled data, though it often relies on assumptions (e.g., uniformity of anomalies) that may not hold in practice. GAN-based techniques [14], [15] learn to distinguish real data from data generated by a generative adversarial network, with anomalies detected based on the discrepancy. Notably, PIAWAL [15] enhances this by introducing weighted adversarial learning to amplify anomalous signal strength. Other approaches, such as MACE [16], utilize spectral representations of transaction behavior, enabling efficient and robust anomaly detection through frequency-domain feature extraction.

Anti money Laundering approaches (AML): Machine learning algorithms are also employed for the detection of money laundering activities. In [17], Support Vector Machines (SVM) were utilized to process large datasets, achieving higher accuracy. The authors of [18] applied fuzzy matching to identify subgraphs containing suspicious accounts. In [19], the authors determined the involvement of capital flow in ML activities using Radial Basis Function (RBF) neural networks calculated over time. Paula et al. [20] demonstrated some success in using deep neural networks for this purpose. However, these algorithms generally detect ML activities in a supervised manner, which can be problematic due to imbalanced labels and limited adaptability. Moreover, ML detection is often an adversarial task, and deep models may lack robustness when facing adversarial attacks.

Graph-based models are particularly effective in detecting complex and non-obvious laundering patterns by capturing structural and temporal relationships between entities. These models typically represent entities (e.g., accounts, addresses) as nodes and interactions (e.g., transactions, fund transfers) as edges, enabling the detection of anomalous substructures within the graph. FlowScope [21] models financial transactions as directed graphs and applies flow tracking to trace the movement of funds across accounts, identifying suspicious routing behaviors indicative of layering stages in money laundering. DiGA [22] leverages graph mining techniques to extract subgraphs with characteristics typical of anti-money laundering (AML) scenarios, such as fan-in/fan-out or circular patterns, which are often manually curated in traditional systems. AMAP [23] utilizes attributed heterogeneous graphs to detect illicit sub-networks by learning node embeddings that reflect both structural roles and attribute similarities, enabling the detection of laundering patterns involving identity obfuscation or account reuse. MoNLAD [3] integrates real-time transaction streams into a dynamic graph model and applies temporal anomaly detection to flag accounts that suddenly exhibit behaviors inconsistent with their historical profile.

While traditional graph-based AML models demonstrate strong performance in capturing relational structures, they are primarily designed for regulated banking systems and face limitations when applied to decentralized, large-scale cryptocurrency environments. To address this gap, several graph-based methods have been proposed specifically for cryptocurrency anti-money laundering (AML). ComGA [24] integrates deep multi-layer perceptrons (MLPs) with graph neural networks (GNNs) to learn node representations that capture both transaction features and structural context in crypto networks. AEGIS [25] employs adversarial learning to detect anomalies by modeling node behaviors under distributional shifts, but its application is hindered by high computational costs in large transaction graphs. Clustering-based approaches, such as DeepFD [26] and semi-GCN [27], utilize GNN embeddings to group similar nodes and detect outliers. However, these methods struggle with defining clear cluster boundaries in highly dynamic and noisy crypto networks, often resulting in elevated false positive rates. Despite these advancements, no existing method has demonstrated robust and scalable detection of crypto money laundering patterns in real-world, high-volume blockchain data, highlighting the need for more effective graph-based models tailored to the unique characteristics of decentralized financial systems.

Decentralized transactions in blockchain systems present unique challenges for money laundering detection due to their irreversible execution, lack of centralized oversight, and pseudonymous identities. These properties hinder traditional anomaly detection methods, which often rely on centralized monitoring, identifiable entities, and reversible audit trails. Additionally, standard AML frameworks are typically tailored for regulated environments, where transaction semantics, user identities, and activity histories are accessible assumptions that do not hold in permissionless blockchain ecosystems. Tracing illicit fund flows across decentralized networks requires not only anomaly detection but also pattern recognition across multi-hop, temporally dynamic graphs. Existing models in both anomaly detection and AML are often designed to detect only limited laundering patterns (e.g., fan-out, fan-in, stack, gather-scatter, random walks, or bipartite flows). These models typically encode strong prior assumptions, making them ineffective at generalizing to unseen or evolving laundering strategies in the wild. As a result, they exhibit limited accuracy and high false negative rates, leaving significant detection gaps when applied to complex, multi-pattern, and “off-chain” laundering behaviors common in real-world crypto environments.

III. PRILIMINARIES

Crypto money laundering, driven by blockchain technology, exhibits unique features while sharing similarities with traditional laundering techniques [1], [28], [4], [29], [30], [21], [22], [3]. The decentralized and pseudonymous nature of blockchain introduces new challenges, such as anonymity of addresses, decentralized networks complicating oversight, and the speed and irreversibility of transactions, which reduce intervention opportunities. Criminals often use mixing services or tumblers to obscure transaction trails, while rapid cross-border transactions and the emergence of DeFi platforms and smart contracts create additional laundering opportunities. Centralized and decentralized exchanges are key for converting illicit funds into other assets, necessitating monitoring of trading and withdrawal behaviors. Effective detection requires advanced methods, including graph-based analysis, machine learning, and blockchain forensics, to process large volumes of data in real time and adapt to evolving laundering tactics. Developing robust cryptoAML systems tailored to these challenges is essential for addressing cryptocurrency laundering effectively.

A. Crypto Money Laundering Workflow

Cryptocurrency money laundering typically follows three core stages that mirror conventional laundering processes: **placement**, **layering**, and **integration**. In the placement stage, illicit funds are introduced into the cryptocurrency ecosystem by converting fiat currency into digital assets or vice versa. The layering stage involves transferring these assets across multiple accounts, wallets, or platforms often across jurisdictions to obscure the origin of the funds. This can include complex patterns such as rapid trades, mixing services, and cross-asset swaps. Finally, the integration stage entails reintroducing the laundered funds into the legitimate economy, either by converting them back to fiat or using them for legal purchases in digital form [28].

Exchange platforms play a pivotal role throughout this cycle, facilitating both the conversion of fiat to cryptocurrency and the disbursement of laundered assets. Digital wallet addresses serve as intermediaries, allowing illicit actors to move tainted funds through multiple hops before arriving at a final deposit address associated with an exchange. Notably, such laundering activity can be confined entirely within a single blockchain ecosystem, underscoring how decentralised platforms can be exploited for illicit financial flows [28].

To mitigate these risks, cryptocurrency exchanges and forensic analytics firms actively monitor trading activity and trace the flow of suspicious funds. For instance, Chainalysis provides investigative support by labeling wallet addresses involved in reported suspicious activity, allowing law enforcement and financial institutions to follow the trail of illicit assets [28], [31]. Regulatory frameworks increasingly require platforms to implement address monitoring and reporting mechanisms, though limitations persist depending on the type of cryptocurrency, the exchange infrastructure, and jurisdictional compliance requirements [29], [32]. These complexities make detection and prevention of cryptocurrency money laundering a critical challenge for financial crime investigators and compliance professionals alike.

B. Difference Between On-chain and Off-chain Transactions

Understanding the fundamental distinctions between “on-chain” and “off-chain” transactions is essential for analysing the behavior of illicit financial activities in blockchain ecosystems. These two transaction types differ significantly in terms of transparency, scalability, execution mechanisms, and susceptibility to abuse. On-chain transactions are inherently traceable and governed by the blockchain’s consensus protocol, whereas “off-chain” transactions bypass direct ledger recording, enabling faster and more private exchanges. This difference is particularly consequential in the context of crypto money laundering, where criminals may exploit “off-chain” mechanisms to evade detection. Table I summarises the key characteristics that differentiate “on-chain” and “off-chain” transactions, highlighting their respective implications for forensic analysis and anti-money laundering efforts.

C. Off-chain Crypto Money Laundering Patterns

Malicious node behaviors in cryptocurrency networks can be modeled as distinct graph problems, which are crucial for addressing money laundering in real-world scenarios and on exchange platforms. These behaviors often manifest as anomalies within groups of nodes in local subgraphs, deviating from typical single-node or individual transaction patterns. User groups typically form discrete but internally dense subgraphs, which helps address the issue of intrinsic sparsity. Transactions can be represented as weighted directed graphs, where nodes represent accounts or parties, and edges represent asset transfers. The direction of an edge indicates the source and destination of the transfer, while the weight quantifies the value of the transfer. Analyzing transactions as weighted directed graphs provides a robust framework for understanding asset movement, detecting anomalies, and identifying money laundering patterns in cryptocurrency networks. This approach reveals various patterns of cryptocurrency money laundering, as illustrated in Figure 2.

When exploring deeper into the intricacies of crypto money laundering, it is imperative to comprehend the laundering patterns, that criminals employ to hide their operations. **Fan-in** is a prevalent pattern in which money is sent from several sources to one location, combining money from different sources into one account. On the other hand, the **fan-out** pattern entails a single source sending money to several locations, distributing money among different accounts to make tracing more difficult. These strategies are combined in the **gather-scatter** pattern, which distributes the aggregated amount to several accounts after sending several tiny amounts to one account in the gathering phase (scattering phase). The **bipartite** pattern is another advanced technique that isolates direct contacts and hides account links. It consists of two sets of nodes with transactions happening only between nodes of distinct sets. Furthermore, the **random** pattern involves transactions that appear to happen at random, which makes it difficult to detect using conventional analytical techniques. Last, the **stack** pattern is continuously moving money through a sequence of intermediate accounts in a stacked, linear manner, producing a trail of transactions that essentially hides the original source of the money. The mentioned laundering patterns -fan-in, fan-out, bipartite, gather-scatter, and stack— are suggested by IBM [30]. Comprehending these trends is essential to creating effective techniques to identify and stop cryptocurrency money laundering.

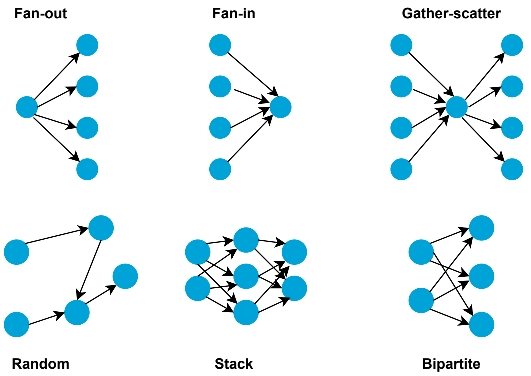


Fig. 2. Crypto money laundering patterns [30]

IV. Problem Definition

For the crypto money transfers, let $G = (V, E, W, T)$ be a directed graph, with V representing the vertex set. The digital wallets’ accounts are represented by V, the set of edges by E, the total amount of money in each account by W, and the timestamp at which the edges are added to each node by T. In our case, we refer to the vertex and account interchangeably throughout the study.

Definition 1 (*Off-chain crypto money laundering Detection*). Given a graph of transactions $G = (V, E, W, T)$, the primary objective is to detect and identify multiple “off-chain” crypto money laundering patterns. These patterns are specific subgraphs $G\,' = (V\,', E\,')$, where $V\,' \subseteq V$ and $E\,' \subseteq E$, that exhibit particular structural characteristics indicative of “off-chain” crypto money laundering activities. The detection of these patterns involves analysing the topological features such as fan-in where a vertex $v \in V$ is the number of incoming edges to v. It is defined as fan-in$(v) = |\{u \in V|(u, v) \in E\}|$.

$$\text{fan-in}(v) = {d_i}^{-}(S) = \Sigma_{v_k \in M_{l-1} \wedge (k,v) \in E} e_{kv} \tag{1}$$

The fan-out of a vertex $v \in V$ is the number of outgoing edges from v. It is defined as fan-out$(v) = |\{w \in V|(v, w) \in E\}|$.

$$\text{fan-out}(v) = {d_i}^{+}(S) = \Sigma_{v_j \in M_{l+1} \wedge (v,j) \in E} e_{vj} \tag{2}$$

The gather-scatter metric for a vertex $v \in V$ combines both the fan-in and fan-out characteristics to evaluate the overall

TABLE I
MAJOR DIFFERENCES BETWEEN ON-CHAIN AND OFF-CHAIN TRANSACTIONS

Aspect	On-Chain Transactions	Off-Chain Transactions
Execution Location	Performed directly on the blockchain ledger	Occurs outside the blockchain ledger
Consensus Involvement	Requires consensus (e.g., PoW, PoS) for validation	Does not require blockchain consensus
Transparency	Publicly visible and auditable	Hidden from public view; often private
Immutability	Immutable once confirmed on the blockchain	Can be modified or reversed before final settlement
Scalability	Limited by block size and network throughput	Highly scalable and faster
Fees	Subject to transaction fees (e.g., gas fees)	Typically low or no blockchain fees
Latency	Can be delayed due to network congestion	Enables near-instant settlement
Security Guarantees	Secured by cryptographic consensus protocols	Relies on trust, escrow, or external enforcement
Use in Illicit Activities	Easier to trace using blockchain analytics	Commonly exploited for laundering due to reduced traceability

connectivity of v. It is defined as gather-scatter(v) = fan-in(v) + fan-out(v).

$$\text{gather-scatter}(v) = d_i^-(S) + d_i^+(S) = \left(\sum_{v_k \in M_{l-1} \wedge (k,v) \in E} e_{kv} \right) + \left(\sum_{v_j \in M_{l+1} \wedge (v,j) \in E} e_{vj} \right)$$

The random refers to the randomness in the occurrence of transactions. For a vertex $v \in V$, the randomness of edges can be represented by the probability of an edge occurrence as random(v) = Probability $((v, w) \in E)$.

$$\text{random}(v) = \frac{\sum_{w \in M_{l+1} \wedge (v,w) \in E} e_{vw}}{|M_{l+1}|} \tag{3}$$

The stack feature is represented by a directed path of nodes connected in a specific order. For a path $P = (v_1, v_2, \ldots, v_k)$ of length k, where each $(v_i, v_{i+1}) \in E$, $v_i \in M_l$, $v_{i+1} \in M_{l+1}$, and P is defined as:

$$P = (v_1, v_2, \ldots, v_k)$$

A graph is bipartite if its vertex set V can be partitioned into two disjoint sets V_1 and V_2 such that every edge connects a vertex in V_1 to a vertex in V_2 and is defined as

$$\forall (u, v) \in E, u \in M_l \Rightarrow v \in M_{l+1} \text{ and vice versa}$$

Table II presents the notations and their meanings used in algorithms.

TABLE II
TABLE OF NOTATIONS AND MEANINGS

Notation	Description
u_i	A node of user account
M_l	Set of nodes at layer l
α	Decay factor (the probability that a random walk terminates at a step)
$K(s)$	Total number of random walks for source node
$d(s)$	Number of out-degrees of the node s
$d(u_i)$	Number of out-degrees of the node u_i
δ, ϵ, P_f	Parameters of an approximate PPR where, δ is threshold, ϵ is error bound, p_f is failure probability
N_s^m	Number of m-hop out-neighbors of the source node s
$r(s, u_i)$	Residue value of node u_i w.r.t source node s
$\sigma(v_i)$	Anomaly score
$N\theta(v_i)$	Normalized θ score for node v_i
$N\omega(v_i)$	Normalized ω score for node v_i
$F_{(\theta,\omega)}(v_i)$	Set of new features

V. MPOCRYPTOML

This section presents the general framework of the proposed model MPOCryptoML, as shown in Figure 3. The primary goal of MPOCryptoML is to identify multiple patterns of "off-chain" crypto money laundering. To track potential laundering routes in a large transaction graph, the Multi-source PPR algorithm is applied to source nodes [22], [33], [34], uncovering random connections and paths between nodes and sources. The PPR scores of each visited node are stored in a tuple called the set of PPR scores (SPS), which is later used in the anomaly score function. Next, the model evaluates the set of normalized timestamp scores (NTS) and normalized weight scores (NWS) to detect additional laundering patterns. Timestamp scores capture anomalies in transaction timing, while weight scores highlight imbalances in transaction volumes. Normalizing these scores ensures comparability and accurate anomaly detection. The NTS and NWS values are analyzed using the multi-pattern crypto money laundering detection algorithm, which trains a logistic regression (LR) model to identify laundering patterns. The resulting pattern scores, denoted as $F_{(\theta,\omega)}$, are then input into the Anomaly Score Function. This comprehensive feature set, including both SPS and $F_{(\theta,\omega)}$, enables the detection of suspicious accounts by identifying various laundering patterns such as fan-in, fan-out, gather-scatter, bipartite, and stack structures, providing a robust method for spotting potential money laundering activities within cryptocurrency transactions.

A. Multi-Source Personalized PageRank Algorithm

Detecting "off-chain" cryptocurrency money laundering is critical due to its significant economic and legal implications. To uncover these hidden laundering techniques, identifying random patterns in transaction graphs is essential, as launderers often use complex strategies to obscure illicit payments. The PPR algorithm excels at detecting such random patterns by highlighting potential laundering paths that might otherwise go unnoticed. PPR ranks nodes based on their relevance to source nodes, focusing on the likelihood of visiting nodes from multiple starting points specifically, the nodes initiating transactions. This probabilistic approach identifies nodes with strong transactional links to the source nodes, assigning them higher relevance scores within dense transactional clusters. By iteratively updating residue and reserve values through random walks, PPR effectively identifies nodes frequently vis-

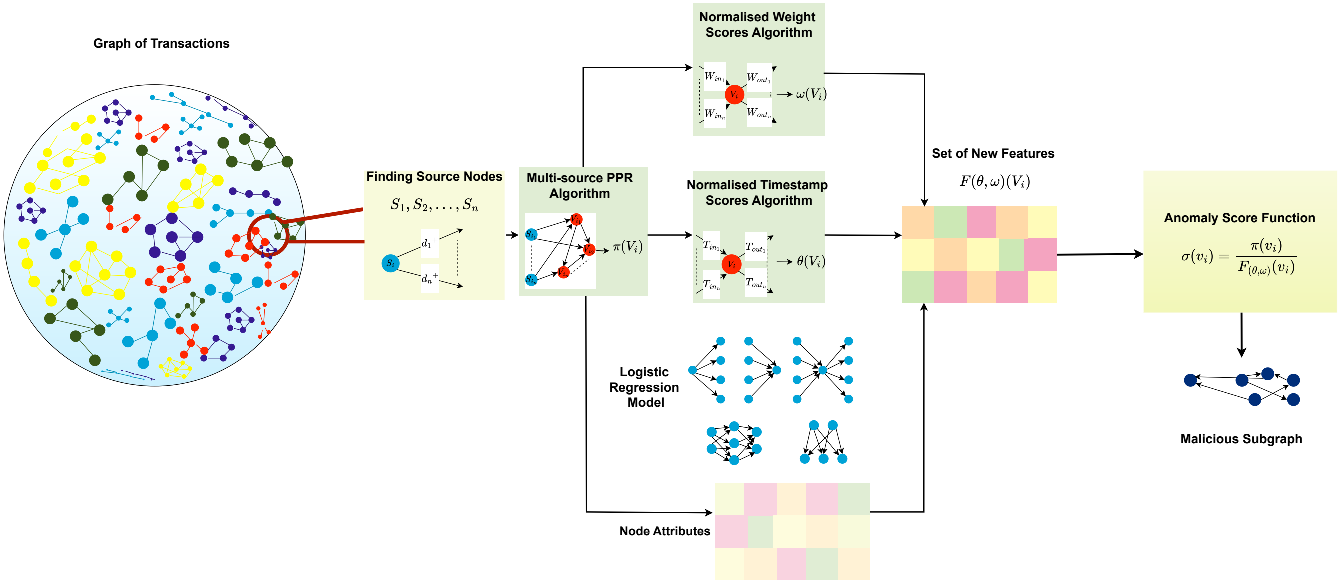


Fig. 3. Crypto-AML inference with MPOCryptoML involves several steps: The multi-source Personalised PageRank algorithm first identifies random laundering patterns. Nodes identified in this stage are then evaluated using a normalized timestamp score algorithm and a normalized weight score algorithm. The scores from these components, along with laundering patterns, are input into a logistic regression model to generate new features for each node to detect additional laundering patterns. These PPR scores and new features are then used by an anomaly score function to compute the anomaly score for each node, ultimately identifying those involved in off-chain cryptocurrency money laundering.

ited, increasing the likelihood of detecting money laundering activities [22] [34].

The primary objective of using the PPR algorithm is to detect malicious nodes that randomly connect to user accounts (source nodes s). Given the computational expense of performing random walks on large transaction networks, we optimize the process by reducing the number of random walks and employing forward pushes to compute PPR scores more efficiently through a k-hop PPR approach. To identify **random** patterns, Algorithm 1 assigns personalised probabilities to multiple sources, representing suspicious nodes within the transaction network. Nodes with high PageRank scores are likely involved in laundering activities, as their close connections to the initial suspicious nodes enable the identification of hidden and randomly dispersed laundering patterns across the network.

To address the CryptoAML detection problem, Algorithm 1 is applied to estimate the multi-source Personalized PageRank (PPR) scores across the transaction network. The algorithm is initiated by identifying all source nodes—i.e., addresses that initiate transactions but do not receive any—by locating nodes with outgoing edges but zero in-degree (lines 3–4). These nodes are central to laundering behaviors and are used as seed points for propagation.

Following source identification, Algorithm 1 initializes two vectors: a residual vector $r(s, \cdot)$ and an intermediate score vector $\pi^{\circ}(s, \cdot)$ for each source node s and its neighbors (lines 5–7). This initialization is necessary to prepare for propagation, where the residual vector reflects the remaining probability mass to be distributed and the score vector accumulates PPR scores.

For each source node s, the walk budget $K(s)$ is computed using the Monte Carlo approximation method from [34] (line 9):

$$K(s) = \frac{(2/3\epsilon + 2) \cdot d(s) \cdot \log\left(\frac{2}{p_f}\right)}{\epsilon^2 \cdot \alpha(1-\alpha)} \tag{4}$$

where $d(s)$ denotes the out-degree of node s, ϵ is the relative accuracy guarantee, p_f is the failure probability, and $\alpha \in (0, 1)$ is the teleport probability. This formulation ensures that highly connected nodes receive a proportionally higher number of walks, increasing statistical robustness in score estimation.

The algorithm's core propagation stage (lines 10–14) executes a forward push from each node u_i whose residual exceeds a dynamic threshold $\frac{d(u_i)}{\alpha \cdot K(s)}$. When this condition is met, the residual at u_i is distributed to its out-neighbors (line 12), and a portion (proportional to α) is added to its own score vector (line 13), followed by resetting the residual at u_i to zero (line 14). This stage reflects a localized score propagation mechanism, ensuring nodes with strong transactional ties accumulate higher scores.

Once no more nodes exceed the residual threshold, the temporary scores are copied to the final score vector $\hat{\pi}(s, \cdot)$ and stored in the result set SPS (lines 15–18). All nodes with nonzero scores are added to the visited node set SVN, which tracks coverage of the transaction subgraph.

To account for remaining residuals, the algorithm proceeds with a Monte Carlo refinement phase (lines 19–24). For each node with residual mass, $r(s, v_i) \cdot K(s)$ random walks are simulated. If a random walk terminates at node $v_i \in N_s^m$, a score increment of $\frac{1}{K(s)}$ is added to $\hat{\pi}(s, v_i)$ (line 24). This reinforces the ranking of nodes frequently reached by random walks, capturing diffusion-based anomalies not propagated in earlier steps.

The final PPR scores (SPS) and visited nodes (SVN) are returned (line 28), completing the estimation procedure. The pseudocode is shown in Algorithm 1 and follows the original PPR formulation with adaptations for multi-source laundering detection [22], [34].

Algorithm 1: Multi-Source PPR Algorithm

```
Input: Graph $G = (V, E, W, T)$, source node $s$, PPR relative accuracy
       guarantee $\epsilon$, failure probability $p_f$, proportion allocation $\alpha$, Hop $K$
Output: A set of PPR scores (SPS), a set of visited nodes (SVN)
$s_{1,2,..,n} \leftarrow 0$;
foreach $v_i \in V$ do
    if in-degree $(v_i) = 0$ then
        Add $v_i$ to the list of source nodes ;     // Identify source nodes with no incoming edges
$r(s, s) \leftarrow 1$; $\pi^\circ(s, s) \leftarrow 0$ ; // Initialize residual and score for source node $s$
foreach $u_i \in N_s^m$ do
    $r(s, u_i) \leftarrow 0$, $\pi^\circ(s, u_i) \leftarrow 0$ ; // Initialize neighbors of $s$ with 0 residual and score
foreach $s$ do
    $K(s) \leftarrow \frac{(\frac{2}{3}\epsilon+2)d(s)\log\left(\frac{2}{p_f}\right)}{\epsilon^2\alpha(1-\alpha)}$ ;     // Compute walk count based on accuracy, $\alpha$, and degree
    while exists $u_i$ such that $r(s, u_i) > \frac{d(u_i)}{\alpha \cdot K(s)}$ and $d(u_i) > 0$ do
        foreach $v_j$ that is an out-neighbor of $u_i$ do
            $r(s, v_j) \leftarrow r(s, v_j) + (1 - \alpha) \cdot \frac{r(s,u_i)}{d(u_i)}$ ;     // Push residual to out-neighbors
            $\pi^\circ(s, u_i) \leftarrow \pi^\circ(s, u_i) + \alpha \cdot r(s, u_i)$ ;     // Update score with $\alpha$ portion
            $r(s, u_i) \leftarrow 0$ ;     // Reset residual after push
    foreach $v_i \in N_s^m$ do
        $\hat{\pi}(s, v_i) \leftarrow \pi^\circ(s, v_i)$ ;     // Copy temporary score to final score
        Add $\hat{\pi}(s, v_i)$ to $SPS$ ;  // Store score in the result set
        Add $v_i$ to $SVN$ ;     // Mark node as visited
    foreach $v_i \in N_s^m$ and $r(s, v_i) > 0$ do
        for $i = 1$ to $(r(s, v_i) \cdot K(s))$ do
            Conduct a random walk from $v_i$: $(1 - \alpha) \cdot r(s, u_i) \cdot K(s)$ ;     // Simulate diffusion via random walk
            if the random walk terminates at $v_i$ and $v_i \in N_s^m$ then
                $K(u_i) \leftarrow r(s, u_i) \cdot K(s)$ ;     // Update walk budget
                $\hat{\pi}(s, v_i) \leftarrow \hat{\pi}(s, v_i) + \frac{1}{K(s)}$ ;     // Increment score based on walk result
                Add $\hat{\pi}(s, v_i)$ to $SPS$ ; // Store updated score
                Add $v_i$ to $SVN$ ;     // Add node to visited list
return $SPS$ and $SVN$ ;  // Return final scores and visited nodes
```

B. Multi-Pattern Crypto Money Laundering Algorithm

Money laundering accounts often exhibit distinctive behavior, such as receiving a large number of transactions within a short period and quickly transferring the funds to other accounts. This pattern is indicative of attempts to obscure the origins and destinations of illicit funds. Unlike conventional accounts, money laundering accounts tend to show high transactional activity, both in volume and frequency. Additionally, these accounts typically transfer nearly the same amount of money in and out, creating a symmetry in transaction amounts. This symmetry is a deliberate strategy to maintain the appearance of regular, balanced activity while avoiding detection. By closely matching the amounts of incoming and outgoing transactions, these accounts make it difficult to identify suspicious activity based solely on volume discrepancies. Moreover, the high transaction turnover and volume contribute to the high density of these nodes within the transaction graph, complicating their detection.

Algorithm 2 computes the *Normalized Timestamp Score (NTS)* for each node by capturing the temporal asymmetry between its incoming and outgoing transactions. This metric is particularly effective for identifying transient intermediary accounts—commonly used in money laundering—where funds are rapidly cycled through addresses to conceal origin and destination. These accounts typically exhibit minimal temporal separation between incoming and outgoing activity.

The algorithm operates on the set of visited nodes SVN, identified by the Personalized PageRank (PPR) algorithm, along with two auxiliary mappings: the in-degree timestamp set SITS and the out-degree timestamp set SOTS. For each node $v_i \in \text{SVN}$, the goal is to derive a normalized timestamp deviation score $N_\theta(v_i) \in [0, 1]$ representing how temporally unbalanced the node's transaction behavior is.

The procedure begins by computing the temporal spread of the in-degree timestamps $T_{s_\text{in}}(v_i) \in \text{SITS}(v_i)$ using the difference between maximum and minimum timestamps (lines 3–8):

$$\theta_\text{in}(v_i) = \max T_{s_\text{in}}(v_i) - \min T_{s_\text{in}}(v_i) \tag{5}$$

Similarly, it computes the temporal spread of the out-degree timestamps $T_{s_\text{out}}(v_i) \in \text{SOTS}(v_i)$ (lines 9–14):

$$\theta_\text{out}(v_i) = \max T_{s_\text{out}}(v_i) - \min T_{s_\text{out}}(v_i) \tag{6}$$

The core measure of temporal irregularity is then calculated as the absolute difference between the two spreads (line 15):

$$\theta(v_i) = |\theta_\text{out}(v_i) - \theta_\text{in}(v_i)| \tag{7}$$

This raw difference score $\theta(v_i)$ is stored in a temporary set STS (line 16) for later normalization. To facilitate comparison across nodes with diverse activity patterns, the algorithm applies min-max normalization over all nodes (lines 18–21):

$$N_\theta(v_i) = \frac{\theta(v_i) - \min_v \theta(v)}{\max_v \theta(v) - \min_v \theta(v)} \tag{8}$$

The resulting score $N_\theta(v_i)$ indicates how temporally asymmetric a node's behavior is. Nodes with low scores are indicative of quick-turnaround behavior—characteristic of laundering intermediaries—while high scores suggest delayed redistribution or asset holding, which are less indicative of illicit flow.

The full pseudocode for this timestamp scoring mechanism is presented in Algorithm 2.

Algorithm 3 computes the *Normalized Weight Score (NWS)* for each node by capturing the discrepancy between the aggregated weights of incoming and outgoing transactions. This score quantifies how unbalanced a node's transaction behavior is in terms of transaction volume or value, providing a structural signal for identifying potential laundering intermediaries.

The algorithm operates on the set of visited nodes SVN obtained from the Personalized PageRank (PPR) stage, along with two associated mappings: the set of in-degree weights SIW and the set of out-degree weights SOW. For each node $v_i \in \text{SVN}$, the algorithm first sums the total weight of all incoming transactions (lines 3–6):

$$\omega_\text{in}(v_i) = \sum W_\text{in}(v_i) \tag{9}$$

Algorithm 2: Normalised Timestamp Score Algorithm

```
Input: A set of visited nodes $SVN$, a set of in-degree timestamps for all
       nodes $SITS$, a set of out-degree timestamps for all nodes $SOTS$,
       where $SITS(v_i) \in SITS$ represents the SITS for node $v_i$, and
       $SOTS(v_i) \in SOTS$ represents the SOTS for node $v_i$
Output: A set of normalized timestamp scores
       $NTS = \{N\theta(v_1), N\theta(v_2), \ldots, N\theta(v_i)\}$
1  foreach $v_i \in SVN$ do
                                   // Iterate over each visited node
       // --- Compute in-degree timestamp range ---
2      $\forall TS_{in}(v_i) \in SITS(v_i)$ Max $TS_{in}(v_i) \leftarrow -\infty$ Min
         $TS_{in}(v_i) \leftarrow +\infty$;
3      foreach $TS_{in}(v_i) \in SITS(v_i)$ do
                              // Iterate over in-degree timestamps if
           $TS_{in}(v_i) >$ Max $TS_{in}(v_i)$ then
4              Max $TS_{in}(v_i) \leftarrow TS_{in}(v_i)$
5          if $TS_{in}(v_i) <$ Min $TS_{in}(v_i)$ then
6              Min $TS_{in}(v_i) \leftarrow TS_{in}(v_i)$
       // --- Compute out-degree timestamp range ---
7      $\forall TS_{out}(v_i) \in SOTS(v_i)$ Max $TS_{out}(v_i) \leftarrow -\infty$ Min
         $TS_{out}(v_i) \leftarrow +\infty$
8      foreach $TS_{out}(v_i) \in SOTS(v_i)$ do
                             // Iterate over out-degree timestamps if
           $TS_{out}(v_i) >$ Max $TS_{out}(v_i)$ then
9              Max $TS_{out}(v_i) \leftarrow TS_{out}(v_i)$
10         if $TS_{out}(v_i) <$ Min $TS_{out}(v_i)$ then
11             Min $TS_{out}(v_i) \leftarrow TS_{out}(v_i)$
12     $\theta_{in}(v_i) \leftarrow$ Max $TS_{in}(v_i)-$ Min $TS_{in}(v_i)$ ;        // Compute
         in-degree timestamp spread
13     $\theta_{out}(v_i) \leftarrow$ Max $TS_{out}(v_i)-$ Min $TS_{out}(v_i)$ ;     // Compute
         out-degree timestamp spread
14     $\theta(v_i) \leftarrow |\theta_{out}(v_i) - \theta_{in}(v_i)|$ ;     // Absolute difference
         between out- and in-degree spreads
15     Add $\theta(v_i)$ to $STS$ ;     // Store the unnormalized score
16 foreach $\theta(v_i) \in STS$ do
                                   // Normalize all timestamp scores
       $N\theta(v_i) \leftarrow \frac{\theta(v_i) - \text{Min } \theta(v)}{\text{Max } \theta(v) - \text{Min } \theta(v)}$ ;                // Apply min-max
       normalisation
17     Add $N\theta(v_i)$ to $NTS$ ;     // Store the normalised score
18 return $NTS$ ;   // Return the set of normalized timestamp
     scores
```

Algorithm 3: Normalised Weight Score Algorithm

```
Input: Set of visited nodes $SVN$, set of in-degree weights
       $SIW = \{W_{in}(v_1), W_{in}(v_2), \ldots, W_{in}(v_i)\}$, set of out-degree
       weights $SOW = \{W_{out}(v_1), W_{out}(v_2), \ldots, W_{out}(v_i)\}$
Output: Set of normalised weight scores
       $NWS = \{N\omega(v_1), N\omega(v_2), \ldots, N\omega(v_i)\}$
1  foreach $v_i \in SVN$ do
                                   // Iterate over each visited node
2      foreach $W_{in}(v_i) \in SIW$ do
                               // Iterate over in-degree weights of $v_i$
           $\omega_{in}(v_i) = \sum W_{in}(v_i)$ ;           // Sum all in-degree
           weights for node $v_i$
3      foreach $W_{out}(v_i) \in SOW$ do
                              // Iterate over out-degree weights of $v_i$
           $\omega_{out}(v_i) = \sum W_{out}(v_i)$ ;        // Sum all out-degree
           weights for node $v_i$
4      $\omega(v_i) = |\omega_{in}(v_i) - \omega_{out}(v_i)|$ ;      // Calculate absolute
         difference between in- and out-degree weights
5      Add $\omega(v_i)$ to $SWS$ ;   // Store the unnormalszed weight
         score
6  foreach $\omega(v_i) \in SWS$ do
                                   // Normalise all weight scores
       $N\omega(v_i) = \frac{\omega(v_i) - \min \omega(v)}{\max \omega(v) - \min \omega(v)}$ ;            // Apply min-max
       normalisation
7      Add $N\omega(v_i)$ to $NWS$ ; // Store normalised weight score
8  return $NWS$ ;     // Return all normalised weight scores
```

and separately sums the total weight of all outgoing transactions:

$$\omega_{\text{out}}(v_i) = \sum W_{\text{out}}(v_i) \tag{10}$$

To measure the imbalance, the absolute difference between incoming and outgoing weights is calculated for each node (line 7):

$$\omega(v_i) = |\omega_{\text{in}}(v_i) - \omega_{\text{out}}(v_i)| \tag{11}$$

This raw difference score $\omega(v_i)$ is stored in a temporary set SWS (line 8). To normalize and standardize the scores across the entire set of nodes, min-max normalization is applied (lines 9–11):

$$N\omega(v_i) = \frac{\omega(v_i) - \min_v \omega(v)}{\max_v \omega(v) - \min_v \omega(v)} \tag{12}$$

The resulting value $N\omega(v_i) \in [0, 1]$ captures how uneven a node's transaction volume is, with low values indicating balanced flow and high values potentially flagging nodes that act as transactional funnels or relays—common roles in money laundering circuits.

The full pseudocode of this weight scoring mechanism is provided in Algorithm 3.

Algorithm 4 introduces a supervised learning framework for detecting diverse money laundering patterns by leveraging two anomaly-based indicators: the *Normalized Timestamp Score (NTS)* and the *Normalized Weight Score (NWS)*. These features, derived from Algorithms 2 and 3, are combined into a unified representation:

$$F_{(\theta,\omega)}(v_i) = [N_\theta(v_i), N_\omega(v_i)] \tag{13}$$

This fused feature vector captures both temporal irregularities and transactional imbalances for each node v_i in the transaction network. A logistic regression (LR) classifier is then trained on this feature set to distinguish between normal and suspicious node behaviors. The trained model is capable of recognizing a wide range of laundering patterns: **Fan-in / Fan-out:** Detected via sharp disparities in $\omega_{\text{in}}(v_i)$ vs. $\omega_{\text{out}}(v_i)$ and in $\theta_{\text{in}}(v_i)$ vs. $\theta_{\text{out}}(v_i)$. **Bipartite:** Identified through intermediate NTS/NWS values in nodes bridging disjoint sets. **Gather-Scatter:** Indicated by high variance in both $\theta(v_i)$ and $\omega(v_i)$ scores. **Stack:** Revealed through sequences of nodes with consistent θ and ω values representing rapid fund relay. By encoding structural and temporal behavior into an interpretable model, Algorithm 4 provides a scalable and effective mechanism for identifying complex laundering activities in transaction networks.

Algorithm 4: Multi-pattern Crypto Money-laundering Detection Algorithm

```
Input: Set of normalized timestamps $NTS$, set of normalized weight scores
       $NWS$
Output: New set of features $F_{(\theta,\omega)}(v_i)$
1  Function TrainModel($NTS$, $NWS$):
2      Combine $NTS$ and $NWS$ into a single feature set $F$
3      Train a logistic regression model LR using $F$ to identify laundering
         patterns
4      Use the trained model LR to generate predictions and identify laundering
         patterns in the data
5      return $F_{(\theta,\omega)}(v_i)$
```

C. Anomaly Score Function

The Anomaly Score Algorithm (AS) aims to identify malicious nodes involved in cryptocurrency money laundering by assigning an anomaly score to each node in the network. This score aggregates data from multiple sources, including random scores from the multi-source PPR algorithm $\pi(v_i)$

and money laundering scores from the multi-pattern crypto money laundering algorithm $F_{(\theta,\omega)}$, based on NWS and NTS values. The goal is to compute an anomaly score $\sigma(v_i)$ for each node. For each node $v_i \in SVN$, the algorithm retrieves the suspicious pattern score $\sigma(v_i)$ from SPS which is a set of PPR scores for each visited node in the transaction graph and a set of features from normalised timestamp, normalised weight scores, and money laundering patterns $F_{(\theta,\omega)}$. It then calculates the anomaly score $\sigma(v_i)$ using the equation (14). A set of anomaly scores (SAS) identifying nodes possibly involved in illegal activity is eventually generated.

$$\sigma(v_i) = \frac{\pi(v_i)}{F_{(\theta,\omega)}(v_i)} \tag{14}$$

This formula integrates the calculated scores, amplifying the anomaly score for nodes that exhibit strong indications of suspicious behaviour across multiple metrics.

VI. Experiments and Results

This section outlines the experimental parameters and compares the proposed method and state-of-the-art techniques.

A. Experimental Settings

*1) **Dataset**:* We use three real-world “off-chain” datasets: Bitcoin payment flows in wallets from “Elliptic++”[1], wallet addresses from “Ethereum Fraud Detection”[2], and the “Wormhole”[3] dataset on Ethereum. We use three real-world datasets as described below:

- **Elliptic++:** Bitcoin payment flows recorded in the ”Elliptic++” dataset, utilizing wallet addresses provided by Elliptic to track “off-chain” transactions. We select the transaction period before time step 42 to mitigate the effects of a whole reconfiguration of the network topology, which results from an abrupt closing of the dark market at time step 43 (see [22] and [35]).
- **Ethereum Fraud Detection:** The ”Ethereum Fraud Detection Dataset” is another dataset that provides wallet addresses recording “off-chain” transactions. This imbalanced dataset includes rows including legitimate and known fraudulent Ethereum transactions and is unbalanced when modeling. The number of unique addresses (nodes) is to be greater than the number of edges (transactions) in the Ethereum transaction graph. This relationship is characteristic of many real-world networks and datasets.
- **Wormhole:** The ”Wormhole Hack” on Ethereum occurred in February 2022, targeting the Wormhole cross-chain bridge, which facilitates interoperability between Ethereum and other blockchains. Since cross-chain transactions involve both “on-chain” and “off-chain” components, we specifically extracted the “off-chain” transactions for our analysis. The hacker exploited a vulnerability in the bridge’s smart contract to mint 120,000 wrapped Ethereum (wETH) on Solana without having the required Ethereum backing it, resulting in a loss of about 325 million. This exploit exposed the security risks of cross-chain protocols, as they rely on the integrity of wrapped tokens to ensure assets are properly backed. Jump Crypto, Wormhole’s parent company, later replenished the stolen funds to restore the bridge’s liquidity.

All three real-world datasets are labeled, with ground truth available for analysis. The statistical data from the three datasets are listed in Table III.

TABLE III
Statistics of datasets

Dataset	Elliptic++	Ethereum	Wormhole
Nodes	203,769	9816	219581
Edges	234,555	9265	236295
Node features	166	51	9
Anomalies	4,545	2179	158
Avg. Degree	1.15	1.85	1.076

*2) **Evaluation Metrics**:* We employ five metrics that are frequently utilised in AML detection: The probability that the model will give a randomly chosen anomaly sample a higher score than a randomly chosen normal sample is represented by the Area Under the ROC Curve or AUC [36] [37]. When the class is unbalanced, AUC has more robustness and is not affected by threshold selection [38]. As a result, we follow the majority of earlier research and employ AUC for ranking-based evaluation [39]. Precision$@K$ is the proportion of correctly identified anomaly samples out of the top$@K$ samples predicted as anomalies, ranked by the anomaly score. Here, K corresponds to the total number of anomalies predicted by the model. Precision is crucial in anomaly detection scenarios where the cost of false positives such as wrongful accusations in fraud detection or unnecessary treatment in disease diagnosis can be substantial. Thus, high precision ensures the model’s predictions are reliable and minimizes false alarms [40] [41]. Recall$@K$ refers to the fraction of correctly identified anomaly samples within the top$@K$ ranked anomalies based on their anomaly scores [42]. In this context, K represents the total number of anomalies in the dataset. The consequences of missing anomalies are critical in fields like fraud detection and disease diagnosis [36], [43]. Therefore, achieving a high recall rate is crucial for the effectiveness of a detection model. We calculated the F1-score and accuracy to evaluate the model’s performance. The F1-score balances precision and recall, making it especially useful for imbalanced datasets where anomalies are rare [44]. It helps assess the trade-off between correctly identifying anomalies and minimizing false positives and negatives. Accuracy, while measuring overall correctness, may not fully capture the model’s ability to detect anomalies in such cases, so the F1-score provides a more nuanced evaluation [45], [46].

*3) **Baselines**:* The proposed model is compared to seven cutting-edge baselines, including traditional supervised machine learning models and advanced GNN-based anomaly detection methods. Specifically, we evaluate it against XGBOOST, DeepFD, OCGTL, ComGA, FlowScope, GUDI, and MACE.

[1] https://www.elliptic.co

[2] https://www.kaggle.com/datasets/vagifa/ethereum-frauddetection-dataset

[3] https://rekt.news/wormhole-rekt/

XGBooST [47] is a boosting supervised technique based on decision trees. By analyzing the importance of features, XGBoost exhibits great classification accuracy and interpretability when used as the supervised machine learning baseline.

DeepFD [26] is a baseline for unsupervised learning. Subgraph-level anomaly detection implies that anomaly-dense blocks in the embedding space can be found to address fraud detection issues such as AML. As a result, it employs Autoencoder to embed the graph and DBSCAN to cluster the dense areas.

OCGTL [48] is the benchmark for graph-level anomaly detection, which achieves the SOTA performance in social network and molecular datasets by combining graph transformation learning and one-class pooling.

ComGA [24] is a deep graph convolution network incorporated as the state-of-the-art GNN-based baseline to encapsulate the finished attributed graph and reconstruct the network using attribute and structure decoders. As a result, it may calculate the anomalous score using the feature matrix and the reconstructed neighboring matrix.

Flowscope [21] can identify the entire money flow from source to destination by modeling transactions with a multipartite graph.

GUDI [49] is a pioneering supervised framework that combines self-supervised pre-training to capture broad graph patterns with few-shot learning to classify domain-specific anomalies. Its guided diffusion mechanism synthesizes pretrained and domain-specific model outputs during inference, bypassing fine-tuning and enabling efficient domain adaptation.

MACE [16] is a multi-normal-pattern unsupervised anomaly detection approach for time series in the frequency domain, characterized by a pattern extraction mechanism for diverse normal patterns, a dualistic convolution mechanism for amplifying short-term anomalies and hindering reconstruction.

B. Experimental Setup

All experiments are conducted on an AWS EC2 `g4dn.2xlarge` instance equipped with 1 NVIDIA T4 Tensor Core GPU, 32 GiB memory, and 225 GB NVMe SSD local storage, running Ubuntu Linux. The proposed model is implemented in Python using the PyTorch and Scikit-learn libraries.

C. Hyperparameter Tuning

To ensure a fair and reproducible evaluation, we initialise key hyperparameters in accordance with the empirical design guidelines proposed in [34]. Specifically, we set the restart probability $\rho = 1$, the failure probability $p_f = 1$, and the random walk weighting factor $\alpha = 0.5$ in the multi-source Personalized PageRank (PPR) algorithm. These values were determined via sensitivity analysis to balance detection performance against computational cost.

Effect of Random Walk Weighting Factor α: The parameter α governs the balance between global exploration and local reinforcement in the multi-source PPR. Lower values (e.g., $\alpha = 0.3$) promote deeper exploration by allowing longer random walks, while higher values (e.g., $\alpha = 0.7$) favor returning to the source node, reinforcing the local neighborhood. We evaluated $\alpha \in \{0.3, 0.5, 0.7\}$ across the Elliptic++, Ethereum, and Wormhole datasets. As shown in Table IV, $\alpha = 0.5$ consistently yields the highest Precision@K and AUC scores. This choice offers a balanced trade-off, effectively capturing laundering behaviors that are neither strictly local nor globally diffused.

Effect of Maximum Iterations max_{iter}: We varied the maximum number of iterations for logistic regression training, $max_{iter} \in \{500, 1000, 2000\}$, to analyse the effect on convergence and performance. While increasing max_{iter} from 500 to 1000 improves both AUC and Precision@K, further increase to 2000 yields only marginal gains with a notable increase in training time. Thus, we select $max_{iter} = 1000$ as a computationally efficient and stable configuration.

Effect of Solver Choice: We compared three solvers for logistic regression: `liblinear`, `lbfgs`, and `saga`. The `liblinear` solver, designed for binary classification and sparse data, outperforms both `lbfgs` and `saga` across all datasets in terms of Precision@K and AUC, as shown in Table IV. This superior performance stems from its compatibility with high-dimensional sparse features derived from transaction subgraphs. Conversely, `saga` shows slower convergence and lower accuracy, indicating limited suitability for the laundering detection task.

Effect of Sampling Ratios: To evaluate robustness under limited supervision, we trained the model using varying sampling ratios of the labeled training data: 50%, 60%, 80%, and 100%. While performance slightly degrades at 50% due to reduced supervision, the proposed `MPOCryptoML` model maintains competitive results across all settings. This robustness demonstrates the model's inductive generalisation ability in sparse and partially labeled environments, which is critical for real-world deployments.

A detailed ablation study is presented in Table IV, summarising the influence of each hyperparameter across multiple datasets. The optimal configuration—$\alpha = 0.5$, $max_{iter} = 1000$, and `solver = liblinear` offers a balanced approach for effective money laundering detection in complex off-chain transaction graphs, combining detection accuracy with computational efficiency.

TABLE IV
HYPERPARAMETER TUNING ACROSS DATASETS

Parameter Setting	**Elliptic++**		**Ethereum**		**Wormhole**	
	Pre@K	**AUC (%)**	**Pre@K**	**AUC (%)**	**Pre@K**	**AUC (%)**
$\alpha = 0.3$	93.11%	93.80	95.02%	96.10	90.12%	92.15
$\alpha = 0.5$	**95.43%**	**94.21**	**97.39%**	**97.40**	**92.55%**	**93.75**
$\alpha = 0.7$	94.05%	93.90	95.88%	96.42	91.06%	92.10
$maxiter = 500$	94.32%	93.45	96.40%	96.85	91.02%	92.35
$maxiter = 1000$	**95.43%**	**94.21**	**97.39%**	**97.40**	**92.55%**	**93.75**
$maxiter = 2000$	95.50%	94.22	97.45%	97.42	92.60%	93.79
`solver = saga`	93.60%	93.55	96.11%	96.90	90.15%	91.70
`solver = lbfgs`	94.85%	93.88	96.72%	97.05	91.33%	92.44
`solver = liblinear`	**95.43%**	**94.21**	**97.39%**	**97.40**	**92.55%**	**93.75**

D. Overall Performance Comparison

Table V compares the overall performance of anomaly detection. We used 80% for training, 10% for validation, and

10% for testing. All results were averaged over three runs. The proposed methodology outperforms several benchmarks across different datasets, with varying degrees of success. The following observations are made from various perspectives:

Table V compares the overall performance of anomaly detection. Over three runs, all of the results are averaged. The suggested methodology beats seven benchmarks with varying degrees of success. The findings from various aspects including overall comparative performance, supervised vs. semi-supervised approaches, and off-Chain laundering detection performance are summarised as follows:

(1) Overall Comparative Performance: MPOCryptoML consistently outperforms existing benchmarks across the Elliptic++, Ethereum, and Wormhole datasets. This superior performance stems from its multi-pattern and node-level focus, which enables fine-grained detection of anomalous behaviors. In contrast, OCGTL performs well on Elliptic++ due to its global graph-level modeling but struggles with Ethereum and Wormhole, where sparse topologies reduce the effectiveness of broad graph transformations. ComGA, a subgraph-level GNN, similarly underperforms on sparse graphs, indicating challenges in applying GNN-based detection to low-density networks.

XGBoost achieves strong performance (85.70% precision, 84.20% recall on Ethereum) but lags behind MPOCryptoML in the Precision@K metric due to its weaker anomaly ranking capability. DeepFD, based on unsupervised clustering, underperforms on Ethereum and Wormhole, as its assumption that anomalies form dense clusters fails in dynamic laundering contexts. GUDI achieves moderate precision on Elliptic++ (74.69%) but shows limited generalizability due to reliance on pre-trained graph patterns. Flowscope yields the weakest results across all datasets. Although MACE achieves high precision on Wormhole (96.03%), it is less effective than MPOCryptoML at ranking anomalies.

(2) Supervised vs. Semi-supervised and Unsupervised Approaches: Semi-supervised methods, particularly MPOCryptoML, outperform both unsupervised (DeepFD, MACE) and supervised (XGBoost, GUDI) baselines in AML anomaly detection. MPOCryptoML leverages weak supervision combined with graph reconstruction to support inductive learning and effectively mitigate label imbalance, a common issue in fraud detection. DeepFD's reliance on density-based clustering fails to align with the irregular structure of laundering behaviors. GUDI, while moderately effective on Elliptic++, struggles with Ethereum and Wormhole due to its general-purpose pre-trained graph templates.

Although XGBoost performs well overall, it ranks anomalous accounts less effectively than MPOCryptoML. The latter's hybrid learning strategy combining structure-aware graph encoding with label guidance demonstrates robustness in modeling transaction complexity and outperforming both conventional classifiers and unsupervised techniques in Precision@K and detection accuracy.

(3) Off-Chain Laundering Detection Performance: MPOCryptoML demonstrates strong capabilities in detecting off-chain laundering behaviors through its integration of the Personalized PageRank algorithm and multi-pattern analysis. This enables the model to rank suspicious accounts with high fidelity, achieving superior performance in Precision@K. In comparison, ComGA, which uses GANs for anomaly detection, offers limited effectiveness due to its generalized learning objective. MACE, despite showing high precision and recall, lacks detailed multi-pattern modeling, leading to reduced effectiveness in prioritizing anomalies.

Overall, MPOCryptoML's ability to capture both random and structured laundering behaviors across heterogeneous transaction patterns enables superior identification of illicit activities when compared to state-of-the-art methods.

E. Ablation Study

To assess each module's functionality in the proposed model, we run an ablation study on anomaly detection performance in this section. We create two ablation models and assess how well they work from an effective and efficient standpoint. Each multi-pattern and random cryptoML account detection module is executed independently.

In the multi-pattern cryptoML model, we remove the logistic regression component from the Normalised Time and Weight algorithm, resulting in the model called the **Normalised T/W model**. Additionally, we modify the multi-pattern cryptoML model to only test the effect of finding random accounts recognized by the Personalised PageRank algorithm, resulting in the **Random cryptoML model**. This assessment primarily assesses each module's ability to identify hidden abnormal patterns in a strictly supervised setting. Table VI displays a comparison of their results with the entire proposed model architecture.

Overall, our design aims are validated by the average gain of the proposed "off-chain" cryptoML when compared to each ablation. The combination of two modules has a considerable improvement over finding cryptoML accounts as each separately can only address specific patterns of cryptoML in the graph of transactions.

VII. Application Scenarios

For financial firms like JPMorgan Chase or Goldman Sachs, MPOCryptoML model can be extremely helpful in combating illegal activity in the cryptocurrency market. These multinational financial behemoths are subject to strict anti-money laundering (AML) standards and are increasingly handling transactions involving cryptocurrencies. The model can spot suspicious trends that point to money laundering by examining particular blockchain nodes, such as wallets or addresses. For instance, the model flags activities for more examination if a wallet often exchanges money with several addresses without a clear, valid reason or conducts a lot of little transactions that might be an attempt to hide the source of funds. By automating the identification of suspicious transactions, financial institutions can reduce the risk of regulatory fines and maintain their standing as complying organisations.

Due to their significant user numbers and the decentralised nature of cryptocurrency transactions, exchange platforms like Binance and Coinbase are especially susceptible to money

TABLE V
OVERALL GRAPH ANOMALY DETECTION PERFORMANCE COMPARISON

Dataset	**Elliptic++**					**Ethereum**					**Wormhole**				
Metrics	**Pre@K**	**Recall@K**	**F1-score**	**ACC(%)**	**AUC(%)**	**Pre@K**	**Recall@K**	**F1-score**	**ACC(%)**	**AUC(%)**	**Pre@K**	**Recall@K**	**F1-score**	**ACC(%)**	**AUC(%)**
XGBoosT	81.30%	80.35%	80.05%	81.30%	81.80%	85.70%	84.20%	84.50%	85.70%	93.90%	82.50%	81.20%	82.50%	82.50%	85.30%
DeepFD	68.80%	67.30%	67.50%	68.20%	72.30%	57.10%	55.24%	56.01%	57.10%	67.46%	62.50%	61.47%	61.52%	62.49%	69.05%
OCGTL	91.20%	90.28%	90.57%	91.23%	90.03%	75.42%	74.27%	74.56%	75.01%	85.05%	79.05%	78.12%	78.56%	79.93%	86.11%
ComGA	85.50%	84.34%	84.53%	85.22%	85.20%	73.21%	72.22%	72.55%	73.02%	74.07%	78.55%	74.41%	77.61%	78.49%	79.50%
Flowscope	51.90%	50.29%	51.03%	51.77%	68.30%	49.65%	48.18%	48.48%	49.58%	67.95%	35.85%	34.49%	34.50%	35.51%	63.55%
GUDI	74.69%	73.19%	73.52%	74.24%	88.01%	64.84%	63.42%	63.05%	65.01%	86.50%	52.25%	51.39%	51.52%	52.24%	60.55%
MACE	94.23%	91.10%	92.02%	93.05%	**95.02%**	96.03%	89.03%	92.03%	93.04%	95.03%	90.02%	84.05%	86.01%	87.03%	92.03%
MPOCrpytoML	**95.43%**	**94.23%**	**94.51%**	**95.41%**	94.21%	**97.39%**	**96.16%**	**96.48%**	**97.39%**	**97.40%**	**92.55%**	**91.11%**	**90.89%**	**92.57%**	**93.75%**

TABLE VI
PERFORMANCE COMPARISON OF ABLATION MODELS

Dataset	**Ethereum**	**Elliptic++**	**Wormhole**
Normalised T/W model	74.90%	69.10%	66.75%
Random cryptoML model	67.81%	67.97%	63.50%
MPOCrpytoML	**97.40%**	**94.21%**	**91.85%**

laundering activities. Wallet operations can be monitored using the MPOCryptoML, which can identify suspicious activity such as transactions to high-risk jurisdictions or large-scale, quick fund transfers in chains. A model like this would be beneficial in detecting and stopping behaviours that are frequently used in money laundering schemes, such as the usage of mixers, decentralised exchanges (DEXs), or unlawful asset conversions. By doing this, these platforms would be able to improve their AML procedures and better identify laundering strategies like integration and layering before illegal monies enter the legal economy.

To improve their oversight of cryptocurrency transactions, regulatory bodies like the Financial Action Task Force (FATF) or national organisations like the U.S. Securities and Exchange Commission (SEC) could incorporate node-level crypto money laundering detection models into their blockchain analytics platforms. These agencies can use these models to track illegal assets that have been laundered through cryptocurrency gambling sites like Stake.com, exchanges, or mixing services like Tornado Cash. For instance, the model may be able to identify the source of stolen funds if they pass through several cryptocurrency exchanges. This would help law enforcement spot illicit activities, trace the source of funding, and take the necessary legal action to stop criminal enterprises from operating in the crypto space.

VIII. CONCLUSION

In this study, we present MPOCryptoML, a novel "off-chain" crypto-AML method that is the first to address crypto money laundering using an "off-chain" strategy. Our approach is designed to handle large-scale, benign cryptocurrency transactions while accurately identifying various money laundering patterns. Due to the confidential nature of transaction data, a comprehensive statistical evaluation of the explanation quality will be deferred, as a thorough human-involved assessment is currently impractical. Future work will extend this approach to fraud detection in cross-chain financial transactions and emerging events on social networks.

ACKNOWLEDGMENT

This work was fully supported by the CloudTech RMIT Green Bitcoin Joint Research Program.

REFERENCES

[1] G. Yang, X. Liu, and B. Li, "Anti-money laundering supervision by intelligent algorithm," *Computers and Security*, p. 103344, 2023.

[2] U. Nations. (2023) Money Laundering Through Cryptocurrencies. [Online]. Available: https://syntheticdrugs.unodc.org/syntheticdrugs/en/cybercrime/launderingproceeds/moneylaundering.html

[3] X. Sun, W. Feng, S. Liu, Y. Xie, S. Bhatia, B. Hooi, W. Wang, and X. Cheng, "Monlad: Money laundering agents detection in transaction streams," in *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining (WSDM)*, 2022, pp. 976–986.

[4] F. Alliance. (2022) Criminal Abuse of Digital Currencies. [Online]. Available: https://www.austrac.gov.au/sites/default/files/2022-04/AUSTRAC_FCG_PreventingCriminalAbuseOfDigitalCurrencies_FINAL.pdf

[5] C. Team. (2024) Case Study: How We Track Crypto Money Laundering for Off-Chain Crime. [Online]. Available: https://www.chainalysis.com/blog/2024-crypto-money-laundering/#:~:text=The%20goal%20of%20money%20laundering,where%20the%20funds%20came%20from.

[6] K. Kolachala, E. Simsek, M. Ababneh, and R. Vishwanathan, "Sok: Money laundering in cryptocurrencies," in *Proceedings of the 16th International Conference on Availability, Reliability and Security*, 2021, pp. 1–10.

[7] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation-based anomaly detection," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 6, no. 1, pp. 1–39, 2012.

[8] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "Lof: Identifying density-based local outliers," in *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, 2000, pp. 93–104.

[9] L. Ruff, R. Vandermeulen, N. Goernitz, L. Deecke, S. A. Siddiqui, A. Binder, E. Müller, and M. Kloft, "Deep one-class classification," in *International conference on machine learning*. PMLR, 2018, pp. 4393–4402.

[10] Z. Li, Y. Zhao, X. Hu, N. Botta, C. Ionescu, and G. H. Chen, "Ecod: Unsupervised outlier detection using empirical cumulative distribution functions," *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 12, pp. 12 181–12 193, 2022.

[11] B. Zong, Q. Song, M. R. Min, W. Cheng, C. Lumezanu, D. Cho, and H. Chen, "Deep autoencoding gaussian mixture model for unsupervised anomaly detection," in *International conference on learning representations*, 2018.

[12] H. Ju, D. Lee, J. Hwang, J. Namkung, and H. Yu, "Pumad: Pu metric learning for anomaly detection," *Information Sciences*, vol. 523, pp. 167–183, 2020.

[13] H. Mu, R. Sun, G. Yuan, and G. Shi, "Positive unlabeled learning-based anomaly detection in videos," *International Journal of Intelligent Systems*, vol. 36, no. 8, pp. 3767–3788, 2021.

[14] B. Tian, Q. Su, and J. Yin, "Anomaly detection by leveraging incomplete anomalous knowledge with anomaly-aware bidirectional gans," *arXiv preprint arXiv:2204.13335*, 2022.

[15] W. Zong, F. Zhou, M. Pavlovski, and W. Qian, "Peripheral instance augmentation for end-to-end anomaly detection using weighted adversarial learning," in *International Conference on Database Systems for Advanced Applications*. Springer, 2022, pp. 506–522.

[16] F. Chen, Y. Zhang, Z. Qin, L. Fan, R. Jiang, Y. Liang, Q. Wen, and S. Deng, "Learning multi-pattern normalities in the frequency domain for efficient time series anomaly detection," in *2024 IEEE 40th International Conference on Data Engineering (ICDE)*. IEEE, 2024, pp. 747–760.

[17] J. Tang and J. Yin, "Developing an intelligent data discriminating system of anti-money laundering based on svm," in *2005 International conference on machine learning and cybernetics(ACM)*, vol. 6. IEEE, 2005, pp. 3453–3457.

[18] K. Michalak and J. Korczak, "Graph mining approach to suspicious transaction detection," in *2011 Federated conference on computer science and information systems (FedCSIS)*. IEEE, 2011, pp. 69–75.

[19] L.-T. Lv, N. Ji, and J.-L. Zhang, "A rbf neural network model for anti-money laundering," in *2008 International conference on wavelet analysis and pattern recognition*, vol. 1. IEEE, 2008, pp. 209–215.

[20] E. L. Paula, M. Ladeira, R. N. Carvalho, and T. Marzagão, "Deep learning anomaly detection as support fraud investigation in brazilian exports and anti-money laundering," in *2016 15th ieee international conference on machine learning and applications (icmla)*. IEEE, 2016, pp. 954–960.

[21] X. Li, S. Liu, Z. Li, X. Han, C. Shi, B. Hooi, H. Huang, and X. Cheng, "Flowscope: Spotting money laundering based on graphs," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 04, 2020, pp. 4731–4738.

[22] X. Li, Y. Li, X. Mo, H. Xiao, Y. Shen, and L. Chen, "Diga: Guided diffusion model for graph recovery in anti-money laundering," in *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2023, pp. 4404–4413.

[23] Z. Chai, Y. Yang, J. Dan, S. Tian, C. Meng, W. Wang, and Y. Sun, "Towards learning to discover money laundering sub-network in massive transaction network," 2023.

[24] X. Luo, J. Wu, A. Beheshti, J. Yang, X. Zhang, Y. Wang, and S. Xue, "Comga: Community-aware attributed graph anomaly detection," in *Proceedings of the Fifteenth ACM International Conference on Web Search and Data Mining (WSDM)*, 2022, pp. 657–665.

[25] J. Wyatt, A. Leach, S. M. Schmon, and C. G. Willcocks, "Anoddpm: Anomaly detection with denoising diffusion probabilistic models using simplex noise," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 650–656.

[26] H. Wang, C. Zhou, J. Wu, W. Dang, X. Zhu, and J. Wang, "Deep structure learning for fraud detection," in *2018 IEEE international conference on data mining (ICDM)*. IEEE, 2018, pp. 567–576.

[27] A. Kumagai, T. Iwata, and Y. Fujiwara, "Semi-supervised anomaly detection on attributed graphs," in *2021 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2021, pp. 1–8.

[28] C. Team. (2023) Case Study: How We Track Crypto Money Laundering for Off-Chain Crime. [Online]. Available: https://www.chainalysis.com/blog/japan-cryptocurrency-money-laundering-case-study/

[29] E. S. Clare Sullivan. (2023) Trade-based Money Laundering: Risks and Regulatory Responses. [Online]. Available: https://www.aic.gov.au/sites/default/files/2020-05/rpp115.pdf

[30] E. Altman, B. Egressy, J. Blanuša, and K. Atasu, "Realistic synthetic financial transactions for anti-money laundering models," *arXiv preprint arXiv:2306.16424*, 2023.

[31] D. Dupuis and K. Gleason, "Money laundering with cryptocurrency: Open doors and the regulatory dialectic," *Journal of Financial Crime*, vol. 28, no. 1, pp. 60–74, 2020.

[32] S. Jones. (2023) CRegulation of Digital and Crypto Assets. [Online]. Available: https://ministers.treasury.gov.au/ministers/stephen-jones-2022/media-releases/regulation-digital-and-crypto-assets

[33] S. Wang, R. Yang, X. Xiao, Z. Wei, and Y. Yang, "Fora: Simple and effective approximate single-source personalized pagerank," in *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2017, pp. 505–514.

[34] S. Luo, X. Xiao, W. Lin, and B. Kao, "Baton: Batch one-hop personalized pageranks with efficiency and accuracy," *IEEE Transactions on Knowledge and Data Engineering*, vol. 32, no. 10, pp. 1897–1908, 2019.

[35] M. Weber, G. Domeniconi, J. Chen, D. K. I. Weidele, C. Bellei, T. Robinson, and C. E. Leiserson, "Anti-money laundering in bitcoin: Experimenting with graph convolutional networks for financial forensics," *arXiv preprint arXiv:1908.02591*, 2019.

[36] H. Kim, B. S. Lee, W.-Y. Shin, and S. Lim, "Graph anomaly detection with graph neural networks: Current status and challenges," *IEEE Access*, vol. 10, pp. 111 820–111 829, 2022.

[37] X. Ma, J. Wu, S. Xue, J. Yang, C. Zhou, Q. Z. Sheng, H. Xiong, and L. Akoglu, "A comprehensive survey on graph anomaly detection with deep learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 12, pp. 12 012–12 038, 2021.

[38] W. Ma and M. A. Lejeune, "A distributionally robust area under curve maximization model," *Operations Research Letters*, vol. 48, no. 4, pp. 460–466, 2020.

[39] K. Liu, Y. Dou, X. Ding, X. Hu, R. Zhang, H. Peng, L. Sun, and S. Y. Philip, "Pygod: A python library for graph outlier detection," *Journal of Machine Learning Research*, vol. 25, no. 141, pp. 1–9, 2024.

[40] K. Ding, J. Li, N. Agarwal, and H. Liu, "Inductive anomaly detection on attributed networks," in *Proceedings of the twenty-ninth international conference on international joint conferences on artificial intelligence (IJCAI)*, 2021, pp. 1288–1294.

[41] Y. Zheng, M. Jin, Y. Liu, L. Chi, K. T. Phan, and Y.-P. P. Chen, "Generative and contrastive self-supervised learning for graph anomaly detection," *IEEE Transactions on Knowledge and Data Engineering*, vol. 35, no. 12, pp. 12 220–12 233, 2021.

[42] K. Liu, Y. Dou, Y. Zhao, X. Ding, X. Hu, R. Zhang, K. Ding, C. Chen, H. Peng, K. Shu *et al.*, "Bond: Benchmarking unsupervised outlier node detection on static attributed graphs," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 35, pp. 27 021–27 035, 2022.

[43] J. Wolleb, F. Bieder, R. Sandkühler, and P. C. Cattin, "Diffusion models for medical anomaly detection," in *International Conference on Medical image computing and computer-assisted intervention*. Springer, 2022, pp. 35–45.

[44] H. Xu, W. Chen, N. Zhao, Z. Li, J. Bu, Z. Li, Y. Liu, Y. Zhao, D. Pei, Y. Feng *et al.*, "Unsupervised anomaly detection via variational auto-encoder for seasonal kpis in web applications," in *Proceedings of the 2018 world wide web conference (WWW)*, 2018, pp. 187–196.

[45] Y. Su, Y. Zhao, C. Niu, R. Liu, W. Sun, and D. Pei, "Robust anomaly detection for multivariate time series through stochastic recurrent neural network," in *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining (KDD)*, 2019, pp. 2828–2837.

[46] S. Tuli, G. Casale, and N. R. Jennings, "Tranad: Deep transformer networks for anomaly detection in multivariate time series data," *arXiv preprint arXiv:2201.07284*, 2022.

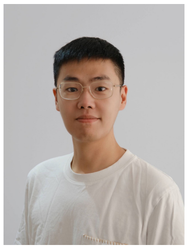
[47] M. Jullum, A. Løland, R. B. Huseby, G. Ånonsen, and J. Lorentzen, "Detecting money laundering transactions with machine learning," *Journal of Money Laundering Control*, vol. 23, no. 1, pp. 173–186, 2020.

[48] C. Qiu, M. Kloft, S. Mandt, and M. Rudolph, "Raising the bar in graph-level anomaly detection," *arXiv preprint arXiv:2205.13845*, 2022.

[49] X. Li and L. Chen, "Graph anomaly detection with domain-agnostic pre-training and few-shot adaptation," in *2024 IEEE 40th International Conference on Data Engineering (ICDE)*. IEEE, 2024, pp. 2667–2680.

Yasaman Samadi Yasaman Samadi is a Ph.D. Candidate at RMIT University, a cybersecurity researcher, and a former quantum computer engineer. Her research focuses on cybersecurity, blockchain fraud detection, and machine learning in financial systems.

Hai Dong (Senior Member, IEEE) received a Ph.D. degree from Curtin University, Perth, Australia, and a Bachelor's degree from Northeastern University, Shenyang, China. He is currently a Senior Lecturer at the School of Computing Technologies, RMIT University, Melbourne, Australia. He also serves as the Chair for the IEEE Computational Intelligence Society Task Force on Deep Edge Intelligence. His primary research interests include Cloud/Edge Computing, Edge Intelligence, Blockchain, Cyber Security, and AI Security. His publications appeared in TOSEM, TIFS, TMC, TSC, TSE, AAAI, ACM MM, ASE, ICML, etc.



Xiaoyu Xia received his PhD degree with the Alfred Deakin Medal from Deakin University, Australia. Currently, he is a lecturer at RMIT University, Australia. He has published over 60 peer-reviewed papers in international journals and conferences, including IEEE S&P, ACM WWW, ACM SIGIR, IEEE TPDS, IEEE TMC, IEEE JSAC, IEEE TSC, etc. He also serves as an Associate Editor for IEEE TDSC and a Review Board Member of the IEEE TPDS. His research interests include system privacy and security, AI privacy, parallel and distributed computing, and sustainable computing. More details about his research can be found at https://xiaoyushawxia.github.io/homepage/.