Expert Systems With Applications 296 (2026) 128900

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

ETTracker: A fund tracking framework for anti-money laundering on Ethereum

Changhao Wu[a,b], Junwei Xu[a], Kai Wang[a], Weili Han[a,b,*], Hongfeng Chai[a,b]

[a] *Computation and Artificial Intelligence Innovative College, Fudan University, Shanghai, China*
[b] *Institute of Financial Technology, Fudan University, Shanghai, China*

ARTICLE INFO

Keywords:
Blockchain
Ethereum
Anti-money laundering
GNN
Taint analysis

ABSTRACT

The anonymity and cross-border features of Ethereum's decentralized finance platform create significant challenges for anti-money laundering (AML) efforts, particularly in tracking illicit fund flows involving service providers such as exchanges, mixers, and decentralized protocols. Existing AML solutions rely primarily on transaction-level heuristics, which terminate tracking at service providers and high-degree addresses, leaving major laundering paths uninvestigated and key service providers unrecognized. To address these limitations, we present ETH-Token Tracker (ETTracker), a novel taint analysis-based framework that enables comprehensive fund flow tracking for AML on Ethereum. ETTracker introduces: (1) Unified Multi-edge Directed Graphs (UMDG), supporting graph-level tracking across service providers during internal transfers; (2) a Service Provider Label Dataset (SP-Dataset) with over 1.22 million categorized addresses, enabling labeled service provider identification in the money laundering path; and (3) a Service Provider Detector Graph Neural Network (SPD-GNN), trained to identify unlabeled service providers and reduce irrelevant transaction noise. Extensive experiments on 11 real-world money laundering cases demonstrate that SPD-GNN achieves a 98.7 % F1-score in service provider identification, outperforming existing models by 1.3 %. ETTracker detects significantly more laundering addresses than state-of-the-art methods, with the largest case achieving a 153-fold increase. ETTracker also tracks 2804 cross-service provider laundering paths and identifies 1001 previously unlabeled service providers, improving the coverage and accuracy of AML investigations.

1. Introduction

Ethereum (Buterin, 2013) has emerged as a transformative blockchain-based decentralized finance platform, distinguished by its smart contract capabilities and versatile application scenarios. Its decentralized nature and anonymous transaction model provide privacy protection for users, but these characteristics simultaneously create opportunities for illicit activities. Consequently, Ethereum has witnessed a surge in illegal operations, including fraud, ransomware attacks, and narcotics trafficking (Farrugia et al., 2020), presenting significant challenges for regulatory authorities. Despite existing research (Fu et al., 2024; Luo et al., 2024) efforts to uncover the patterns of these criminal activities and associated money laundering operations, a comprehensive analysis of fund tracking remains inadequate.

Unlike traditional financial systems, Ethereum's decentralized finance (DeFi) environment enables users to interact with a wide range of service providers (SPs) through smart contracts. These SPs include centralized exchanges (CEXs), decentralized exchanges (DEXs), wallets, cross-chain bridges, mixers, and token services. This flexibility allows malicious actors to engage multiple SPs within a single transaction, thereby obscuring the origins and destinations of illicit funds. Because both legitimate and suspicious accounts often transact through the same SPs, distinguishing between normal and illicit fund flows becomes highly challenging. As is shown in the money laundering process in Fig. 1, this overlap significantly complicates effective AML efforts, especially given the presence of various SPs in the Ethereum ecosystem.

A common approach for AML on Ethereum is taint-based fund tracking (Gómez et al., 2022; Wu et al., 2024), which constructs fund flow graphs originating from blacklisted addresses and serves as the foundation for subsequent precise money laundering detection (Li et al., 2020; Lin et al., 2024). In this paradigm, any recipient of a transaction from a blacklisted address is considered suspicious (Möser et al., 2014). As is shown in the taint-based tracking in Fig. 1, the approach queries all outgoing transactions from a blacklisted address and recursively

* Corresponding author at: Interdisciplinary Building No.2, Jiangwan Campus, Fudan University, Yangpu District, Shanghai, China.
E-mail addresses: chwu22@m.fudan.edu.cn (C. Wu), jwxu24@m.fudan.edu.cn (J. Xu), wangk20@fudan.edu.cn (K. Wang), wlhan@fudan.edu.cn (W. Han), hfchai@fudan.edu.cn (H. Chai).

https://doi.org/10.1016/j.eswa.2025.128900
Received 21 March 2025; Received in revised form 23 May 2025; Accepted 2 July 2025
Available online 5 July 2025

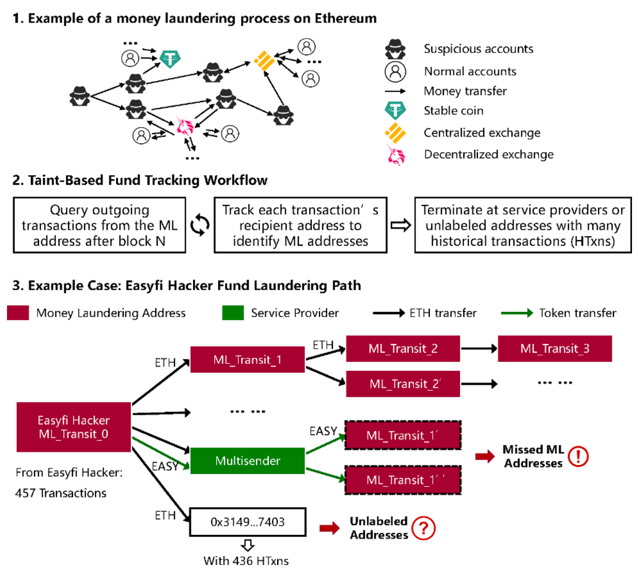

1. Example of a money laundering process on Ethereum
Suspicious accounts
Normal accounts
Money transfer
Stable coin
Centralized exchange
Decentralized exchange
2. Taint-Based Fund Tracking Workflow
Query outgoing transactions from the ML address after block N
Track each transaction' s recipient address to identify ML addresses
Terminate at service providers or unlabeled addresses with many historical transactions (HTxns)
3. Example Case: Easyfi Hacker Fund Laundering Path
Money Laundering Address
Service Provider
ETH transfer
Token transfer
Easyfi Hacker ML_Transit_0
From Easyfi Hacker: 457 Transactions
ML_Transit_1
ML_Transit_2
ML_Transit_2'
ML_Transit_3
Multisender
ML_Transit_1'
ML_Transit_1''
Missed ML Addresses
0x3149...7403
With 436 HTxns
Unlabeled Addresses
ETH
EASY

Fig. 1. Overview and limitations of taint-based fund tracking on Ethereum. (1) Example of a money laundering process. (2) Standard taint-based tracking workflow. (3) Easyfi Hacker case, showing that this approach may miss ML addresses and encounter difficulties in identifying recipient unlabeled SPs.

traverses the recipient addresses. However, when encountering an address with a large number of outgoing transactions, the fund flow graph can expand rapidly, leading to computational challenges or erroneous tracking. Furthermore, when the traversal reaches an SP, it becomes difficult to distinguish between legitimate and illicit outgoing flows. Therefore, the approach treats all SPs equally and terminates tracking at recipient addresses labeled as SPs or with high transaction volumes, as these are presumed to be SPs.

However, as is shown in the Easyfi Hacker case in Fig. 1, this transaction-level tracking approach exhibits two key limitations. First, tracking is prematurely terminated when it encounters a labeled SP such as Multisender in the transaction `0x3dd7…d65f`.[1] The money laundering through Multisender is executed within a single contract call during internal transfers, so downstream money laundering addresses (ML addresses) can in principle still be identified. However, the algorithm halts at this node, thereby missing subsequent ML addresses along the path. Second, for terminal recipient address `0x3149…7403`[2] with a high volume of historical transactions, the approach uses a transaction count threshold to infer SP status. This heuristic can misclassify ML addresses as SPs, resulting in missed ML addresses, or mistakenly identify SPs as ML addresses, leading to the inclusion of legitimate transactions.

In summary, the taint-based fund tracking approach treats all SPs uniformly as terminal nodes, resulting in missed laundering paths and increased noise due to inaccurate endpoint identification. To overcome these limitations, we propose ETTracker (**ETH**-Token **Tracker**), which enables cross-SP tracking during internal transfers and accurate identification of terminal SPs. This approach captures complete laundering paths while minimizing irrelevant transactions. ETTracker constructs a comprehensive money laundering fund flow graph using taint analysis and consists of three main components. The first is the Unified Multi-edge Directed Graph (UMDG), which integrates all transactions with the same transaction hash, including EOA, internal ETH, and token transfers, into a unified structure, enabling the detection of laundering outputs by shifting from transaction-level tracking to graph-based recipient identification. The second component is the Service Provider Label Dataset (SP-Dataset), which integrates labeled addresses from Etherscan, Xblock, and Dune, and categorizes each SP by its function (e.g., exchange, mixer, bridge). New labels are continuously retrieved from Etherscan during tracking to ensure up-to-date information. The third component is the Service Provider Detector Graph Neural Network (SPD-GNN), trained on Service Provider Transaction Graphs (SP-Graphs) extracted from laundering paths involving labeled SPs. By learning SP transactional behavior patterns, SPD-GNN identifies unlabeled SPs, further reducing irrelevant transaction noise and improving tracking precision.

In summary, the main contributions are as follows.

1. **ETTracker:** We propose a UMDG-based fund tracking framework that shifts from transaction-level tracking to fund recipient identification in graphs, thereby enabling cross-SP tracking during internal transfers.
2. **SP-dataset:** We construct a dynamically updated SP-Dataset with over 1,220,000 labeled Ethereum addresses, publicly available online.[3]
3. **SPD-GNN:** We design an SPD-GNN, trained on SP-Graphs extracted from money laundering paths with labeled SPs, which identifies previously unlabeled SPs with a 98.7 % F1-score.

The remainder of this paper is organized as follows. Section 2 introduces Ethereum transactions, SPs, and presents the taint analysis-based AML fund tracking model. Section 3 describes the ETTracker framework, including UMDG construction, SP-Dataset development, and SPD-GNN methodology. Section 4 evaluates the performance of ETTracker against state-of-the-art approaches. Section 5 discusses the implications of ETTracker, followed by related work in Section 6. Section 7 concludes the paper.

2. Background and problem formulation

2.1. Background

2.1.1. Ethereum transactions

Ethereum transactions underpin value transfer and smart contract execution, and fall into three main categories, each serving distinct operational roles.

EOA transaction: The externally owned account (EOA) transaction is the most basic type, initiated by accounts controlled via private keys rather than smart contracts. EOA transactions form the primary interface through which users interact with the Ethereum network, enabling both ETH transfers and smart contract calls.

Internal ETH transfer: Internal transfers occur when smart contracts programmatically redistribute ETH during execution. Although not directly initiated by EOAs, they are central to complex contract logic, supporting automated payments and advanced decentralized financial operations.

Token transaction: Token transactions involve standardized digital assets built on Ethereum, typically adhering to protocols such as ERC-20 for fungible tokens or ERC-721 (Chen et al., 2020) for non-fungible tokens (NFTs). Unlike ETH transfers, which affect native balances, token transfers modify contract-level storage states.

2.1.2. Service provider

Service providers in the blockchain ecosystem deliver a wide range of financial and technical services that support cryptocurrency transactions and protocol-level operations. Accurate SP identification and classification are fundamental to effective on-chain AML analysis. Based on core functionalities, SPs are grouped into the following categories.

[1] 0x3dd7f956a6ee7a47223c0de1cc050933b2d606f1201767e0e66433e57923d65f

[2] 0x31499e03303dd75851a1738e88972cd998337403

[3] https://github.com/fduwch/ETTracker

Exchange: Centralized exchanges (CEXs), such as Binance and Coinbase, act as custodial intermediaries, providing high liquidity and user accessibility. Decentralized exchanges (DEXs), including Uniswap and SushiSwap, employ automated market maker protocols to enable trustless peer-to-peer trading.

Cross-chain bridge: Cross-chain services facilitate interoperability between heterogeneous blockchain networks by enabling asset transfers across chains (Harris, 2023). Examples include Polkadot's relay chain, Portal Bridge, and LayerZero. These systems adopt mechanisms such as lock-and-mint and atomic swaps, balancing security with cross-chain validation efficiency.

Token: Stablecoins such as USDT, USDC, and DAI maintain price stability via collateralization-off-chain for USDT and USDC, and on-chain for DAI. These assets serve as stable exchange media in DeFi. In contrast, non-stable tokens, including governance, utility, and liquidity provider tokens, support trading, staking, and liquidity mining, and often appear in complex fund flows.

Mixer: Mixing services, such as Tornado Cash (Du et al., 2024) and Mixer.cash, use zero-knowledge proofs to enhance transaction privacy via anonymity sets that obscure fund origins. Their evolution highlights ongoing tension between privacy and regulatory compliance in blockchain systems.

NFT: Non-fungible token (NFT) marketplaces enable the creation, transfer, and exchange of uniquely identifiable digital assets. NFTs are widely used in art, gaming, and collectibles, forming a distinct asset class within Ethereum.

Others: This category includes wallets, lending protocols, and other core infrastructure for asset custody, collateralized lending, and smart contract interaction. Wallets function as endpoints for fund transfers, while platforms like Aave and Compound facilitate decentralized lending and borrowing.

2.1.3. Money laundering in blockchain

Money laundering on Ethereum exhibits distinct patterns across three phases: placement, layering, and integration. Each phase exploits specific features of Ethereum and its DeFi infrastructure to conceal the origin and movement of illicit funds.

Placement: This phase involves converting illicit funds into cryptocurrencies via DEXs lacking strict Know Your Customer (KYC) protocols (Rattanabunno et al., 2023) or peer-to-peer platforms enabling fiat-to-ETH exchange. It often leverages compromised wallets or specialized smart contracts to facilitate initial fund injection.

Layering: In this phase, perpetrators employ complex transaction patterns to obscure fund flows, using AMM-based token swaps, yield farming, and cross-chain bridges. Smart contract interactions and internal transfers further reduce traceability as funds circulate through intermediate addresses and undergo repeated token conversions.

Integration: This final stage returns laundered funds to the legitimate economy. Typical methods include converting crypto to fiat through compliant exchanges, using stablecoins as value reserves, or acquiring NFTs as alternative stores of value. It may also involve decentralized lending or yield-generating strategies to simulate legitimate income.

2.2. Problem formulation

Taint analysis has been widely applied in Bitcoin for AML investigations, with two representative policies being Poison and Haircut (Möser et al., 2014). As is shown in Fig. 2, the Poison policy designates any transaction as tainted if it contains at least one tainted input, whereas the Haircut policy assigns taint proportionally based on the ratio of tainted to clean inputs. Both methods rely on the UTXO model, in which the flow of funds can be explicitly tracked through transaction inputs and outputs.

However, such strategies are not directly transferable to Ethereum's account-based model, where transactions lack explicit input–output mappings. To accommodate this structural difference, we adopt the Truncated Poison Policy (TPP) (Wu et al., 2024), under which a transaction recipient is considered suspicious if the sender is blacklisted, unless the recipient behaves similarly to an ordinary user or is a known SP. This policy forms the basis for taint propagation in Ethereum-based AML tracking.

Building on this foundation, we formalize Ethereum-based AML fund tracking as a taint propagation process to construct money laundering paths originating from illicit addresses. The motivation is to support cross-SP tracking during internal transfers and to accurately identify terminal recipient SPs, enabling the capture of fund flow paths while minimizing noise from irrelevant SP transactions. The tracking framework is defined as a four-tuple system:

$$\mathcal{M} = (\mathcal{A}, \mathcal{T}, \mathcal{S}, \mathcal{R}), \tag{1}$$

where $\mathcal{A} \subset \mathbb{A}$ denotes the set of initial taint sources, with $\mathbb{A}$ being the global set of Ethereum addresses; $\mathcal{T} = \{t_e, t_i, t_k\}$ represents the set of considered transactions, including EOA transactions (t_e), internal ETH transfers (t_i), and token transactions (t_k); $\mathcal{S} \subset \mathbb{A}$ denotes the set of SP addresses encountered during propagation; $\mathcal{R} : \mathcal{S} \times \mathcal{T} \to 0, 1$ defines the binary rule determining whether taint propagation continues through a given SP.

The tracking process adheres to the following taint-based principles:

Definition 1 (Taint source set): The process is initiated from $\mathcal{A}$, where each $a \in \mathcal{A}$ is treated as a source of tainted funds. All outgoing transactions originating from these addresses are considered taint carriers, subject to filtering based on structural and semantic constraints.

Definition 2 (Propagation via transactional paths): For each $a \in \mathcal{A}$, the tracking process explores transactions $t \in \mathcal{T}$ to construct a set of taint propagation paths $P = \{p_1, p_2, \dots, p_n\}$, where each path p_i corresponds to a sequential flow of value through (t_e, t_i, t_k) transactions.

Definition 3 (Service provider interaction policy): When a service provider $s \in \mathcal{S}$ is encountered along a path, the rule function $\mathcal{R}(s, t)$ determines the continuation of taint propagation as follows:

$$\mathcal{R}(s,t) = \begin{cases} 1, & \text{if } s \text{ operates as an intermediary within internal fund transfers} \\ 0, & \text{if } s \text{ represents a terminal recipient in the transaction sequence} \end{cases} \tag{2}$$

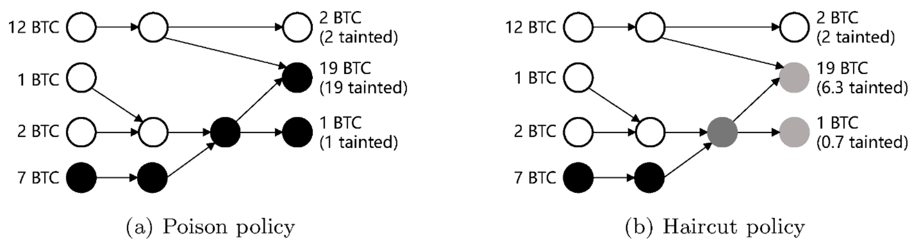


Fig. 2. Taint propagation strategies in Bitcoin under different blacklisting policies.

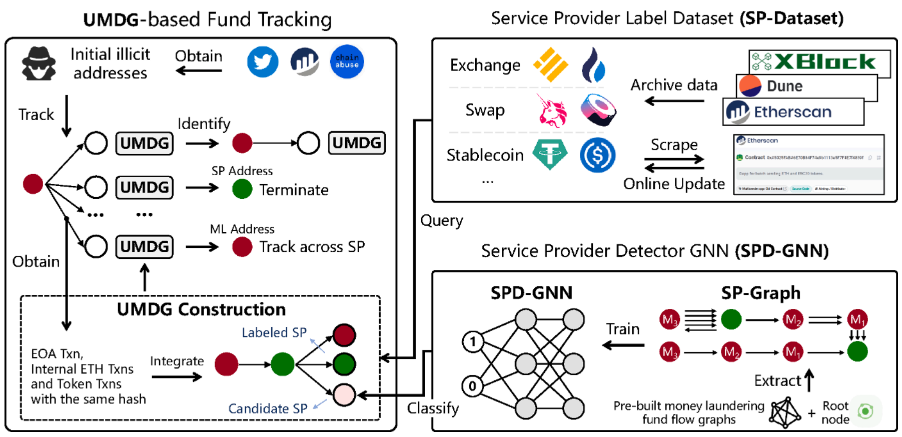


Fig. 3. Overview of the ETTracker framework, consisting of: (i) UMDG-based fund tracking, where the UMDG is built from any transaction originating from an ML address and all transactions sharing the same hash; (ii) a dynamically updated SP-Dataset, sourced from XBlock, Dune, and Etherscan, for identifying labeled SPs within the UMDG; (iii) an SPD-GNN, trained on SP-Graphs (subgraphs extracted from pre-built ML fund flow graphs) to detect unlabeled candidate SPs among downstream addresses.

$\mathcal{R}(s,t) = 1$ indicates that the SP is a non-terminal relay, and taint propagation should continue. Conversely, $\mathcal{R}(s,t) = 0$ denotes a functional endpoint in the laundering flow, terminating taint tracking at that node.

3. ETTracker

As is shown in Fig. 3, the ETTracker framework consists of three major components. First, we construct a UMDG to aggregate all transactions sharing the same transaction hash and to identify downstream ML addresses. Second, we build an SP-Dataset by continuously acquiring and categorizing known SP addresses from multiple sources. Third, we develop an SPD-GNN model to detect previously unlabeled SPs encountered during tracking.

3.1. UMDG

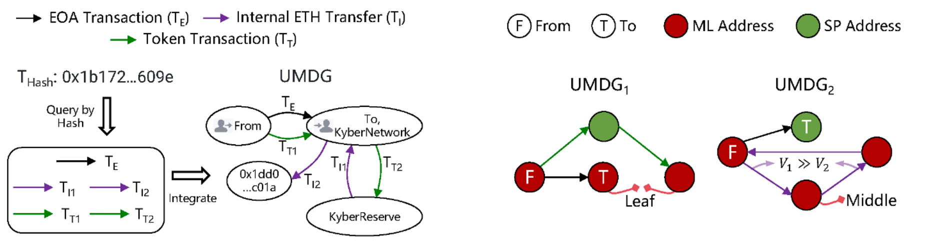
Traditional AML tracking algorithms operate at the transaction level, evaluating each outgoing transaction from a known ML address independently. Tracking typically terminates when a recipient address either has a high number of outgoing transactions or is labeled as an SP, based on the assumption that such activity reflects legitimate user behavior. As a result, high-degree addresses are uniformly treated as terminal nodes, making it difficult to isolate laundering outputs from numerous benign transactions.

However, this approach overlooks a key feature of Ethereum: a single smart contract call from an EOA transaction can trigger multiple internal and token transfers, often involving intermediaries such as SPs. The transaction-level model fails to capture these intra-transaction dependencies, especially when funds are routed through SPs participating in laundering activities. To overcome this limitation, we adopt a graph-based tracking paradigm that aggregates all transaction types sharing the same transaction hash into a unified representation. This approach enables comprehensive tracking of fund flows across multi-step interactions while filtering out irrelevant SP activity.

To implement this approach, we introduce the Unified Multi-edge Directed Graph (UMDG), shifting from transaction-level to graph-level tracking. As is shown in Fig. 4a, the UMDG integrates the initiating EOA transaction with all related internal ETH and token transfers, capturing the complete value movement from a single transaction call. UMDG construction depends on the observed transaction type: for an EOA transaction from an ML address, all associated internal and token transfers are aggregated to enable analysis from the ML address entry point; for an internal or token transfer, the corresponding EOA transaction and all related transfers are combined to form the UMDG, with tracking beginning at the relevant transfer.

As is shown in Fig. 4b, this unified view enables identification of two types of next-hop ML addresses. The Leaf pattern denotes terminal nodes with no subsequent outgoing transfers, while the Middle pattern indicates intermediate nodes where outgoing value is less than incoming value, signaling ongoing laundering activity and prompting continued tracking. Not all SPs allow further taint propagation: mixers, cross-chain bridges, and CEXs generally serve as terminal nodes due to their high user volume or off-chain settlement. In the UMDG-based framework, propagation continues only when an SP acts as an intermediary within internal fund transfers.

Each UMDG corresponds to a unique transaction hash originating from an ML address. By iteratively constructing UMDGs and identifying next-hop ML addresses, the tracking process progressively expands. All UMDGs are then merged into a comprehensive money laundering fund flow graph, capturing multi-hop paths across different transaction types



(a) Integration of all transaction types with the same hash into a UMDG.

(b) Identification of Leaf and Middle ML addresses in UMDGs.

Fig. 4. UMDG construction and fund recipient identification.

Algorithm 1 UMDG-based fund tracking.

```
Require: Initial ML address set $\mathcal{A}$, SP label set $\mathcal{S}$, starting block $b_0$
Ensure: Fund flow graph $G_{\text{flow}}$
 1: Initialize $Q \leftarrow \{(\alpha_0, b_0) | \alpha_0 \in \mathcal{A}\}$, $G_{\text{flow}}$, $\mathcal{V}$
 2: while $Q \neq \emptyset$ and $|\text{inflow}(Q)| < \Psi$ do
 3:     Dequeue $(\alpha, b)$ from $Q$; extract $\mathcal{T}$ from $\alpha$ after $b$
 4:     for each $\tau$ in $\mathcal{T}$ do
 5:         if $\tau$.hash $\notin \mathcal{V}$ then
 6:             Retrieve all related transactions $(\tau_e, \tau_i, \tau_t)$
 7:             Construct UMDG; merge into $G_{\text{flow}}$; add $\tau$.hash to $\mathcal{V}$
 8:             for each $\alpha'$ in middle/leaf recipients $\mathcal{N}(\alpha)$ do
 9:                 if $\alpha' \notin \mathcal{S}$ and isTrackable($\alpha'$) then
10:                     Enqueue $(\alpha', \tau.\text{block})$ to $Q$; add $\alpha'$ to $\mathcal{A}$
11:                 end if
12:             end for
13:         end if
14:     end for
15: end while
16: return $G_{\text{flow}}$
```

and SP interactions. Algorithm 1 summarizes the overall procedure in five steps.

1. **Transaction query**: Extract all transactions $\mathcal{T}$ originating from the ML address α after block b_0, including EOA, internal ETH, and token transfers. Transactions with less than 0.01 ETH (Wu et al., 2024) or \$10 in stablecoin value are filtered out as dirty amounts to eliminate noise.
2. **UMDG construction**: As is shown in Fig. 4a, for each $\tau \in \mathcal{T}$, the associated EOA (τ_e), internal ETH (τ_i), and token (τ_t) transfers sharing the same hash are retrieved and a UMDG is constructed by integrating these transactions.
3. **ML address identification**: As is shown in Fig. 4b, middle and leaf recipients in each UMDG are identified through graph analysis and aggregated as next-hop ML candidates $\mathcal{N}(\alpha)$.
4. **ML address evaluation**: Filter candidates α' based on the SP label set $\mathcal{S}$ and trackability criteria. Labeled SPs are excluded to reduce noise, and addresses failing criteria (e.g., excessive transaction volume or likely unlabeled SPs) are pruned to prevent the explosion of G_{flow}.
5. **Iterative tracking**: Repeat the above steps for all addresses in queue Q until it is empty or the cumulative inflow $|\text{inflow}(Q)|$ exceeds threshold Ψ. The final output is the ML fund flow graph G_{flow}.

In our implementation, the SP label set referenced in the condition $\alpha' \notin \mathcal{S}$ is detailed in Section 3.2. The criteria employed in the isTrackable function for identifying candidate unlabeled SPs are elaborated in Section 3.3.

3.2. SP-dataset

The identification of SPs constitutes a fundamental component of money laundering fund tracking on Ethereum, as these entities often serve as either tracking termination points (such as CEXs and cross-chain bridges) or fund obfuscation mechanisms (such as mixing services). However, comprehensive SP detection remains a significant challenge in blockchain forensics, primarily due to a critical data gap. Over 99 % of Ethereum addresses remain unlabeled (Etherscan, 2025), which severely compromises the operational efficacy of tracking frameworks.

Specifically, this labeling deficiency manifests in two distinct challenges in money laundering fund tracking. First, the misclassification of ML addresses as SPs results in critical tracking omissions, as the tracking process terminates at these misidentified points. Second, the misclassification of genuine SPs as regular addresses leads to substantial noise through the inclusion of irrelevant transactions and incorrect path identifications, thereby compromising the overall efficiency of ML address detection. This limitation becomes particularly pronounced in cases involving addresses with large transaction volumes.

To overcome the label data collection problem, we adopt a universal approach that diverges from traditional methods of separately collecting address labels for individual SP categories. Instead of conducting isolated collection processes for each SP type (e.g., CEX, DEX, DeFi), our approach aggregates all SP-related addresses simultaneously, followed by categorization based on functional attributes. This method leverages three authoritative label repositories: (Etherscan, 2025), (XBlock, 2025), and (Dune, 2025), selected for their extensive coverage across the Ethereum network. Through the integration of multiple data acquisition methods, including automated web crawling, API-based extraction, and direct data downloads, we systematically collect a large-scale SP address dataset while ensuring comprehensive coverage across diverse SP categories.

For systematic integration of multiple labeling sources, we employ a formal methodology where $A = \{a_1, a_2, \dots, a_n\}$ denotes the address set, and $L(a)$ represents the label for address a. Given three sources $S = \{S_1, S_2, S_3\}$, corresponding to Etherscan, XBlock, and Dune, respectively, the label assignment is determined by the following rule:

$$L(a) = \begin{cases} L_1(a), & \text{if } L_1(a) = L_2(a) = L_3(a) \\ \operatorname{argmax}_{i \in 1,2,3}(P(S_i)), & \text{otherwise} \end{cases} \tag{3}$$

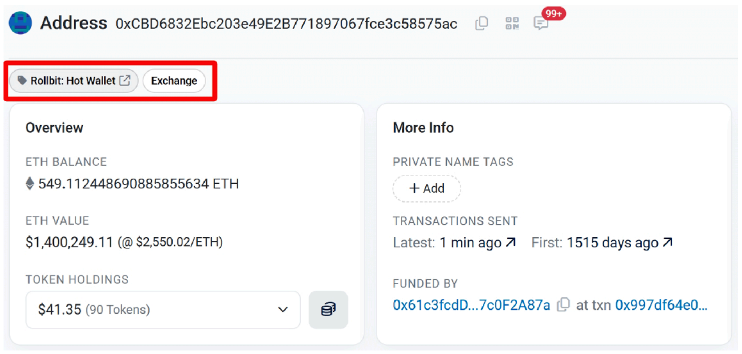
where $P(S_i)$ denotes the priority of source S_i, with $P(S_1) > P(S_2) > P(S_3)$. This priority order is established based on the reliability and comprehensiveness of each source: Etherscan (S_1) is prioritized due to its status as the most widely used Ethereum block explorer, offering verified, up-to-date, and crowdsourced annotations. XBlock (S_2) provides academically curated labels, while Dune (S_3) contributes community-driven but less standardized tags.

To further enhance label reliability during money laundering tracking, ETTracker performs an online verification step for every address identified as an ML address but absent from the static SP-Dataset. As is shown in Fig. 5, we retrieve its Etherscan webpage[4], then extract the latest tag metadata for each such address. This ensures that label acquisition is synchronized with the time of track execution, enabling timely detection of newly established or re-labeled SP addresses. The updated labels obtained through this dynamic querying process are incorporated into the SP-Dataset.

Through this rigorous integration process, our approach yields a comprehensive dataset of over 1.8 million addresses. To ensure data quality and relevance for ETTracker, we further implement a category-based filtering mechanism to exclude non-SP labels. As is shown in Table 1, the detailed categorization (see Section 2.1.2 for category descriptions) and statistical distribution of SP types and their corresponding labels are presented. The constructed SP-Dataset comprises approximately 1.22 million known labels, including 360,683 Token-related addresses, 196,583 DEX addresses, 1815 CEX addresses, 89,757 NFT-related addresses, 2513 Cross Chain addresses, and 140 Mixer addresses. About 653,000 address labels remain unknown and are reconfirmed on Etherscan when encountered during tracking.

This functional classification of SPs deepens the understanding of money laundering activities, revealing patterns such as preferred laundering tools and commonly used tokens. For example, the large number of DEX addresses (196,583) underscores their importance in the Ethereum ecosystem, while the small number of mixer addresses (140) indicates their specialized role in fund obfuscation.

[4] https://etherscan.io/address/0xCBD6832Ebc203e49E2B771897067fce3c58575ac


Address 0xCBD6832Ebc203e49E2B771897067fce3c58575ac

Rollbit: Hot Wallet | Exchange

Overview

ETH BALANCE
549.112448690885855634 ETH

ETH VALUE
$1,400,249.11 (@ $2,550.02/ETH)

TOKEN HOLDINGS
$41.35 (90 Tokens)

More Info

PRIVATE NAME TAGS
+ Add

TRANSACTIONS SENT
Latest: 1 min ago ↗ First: 1515 days ago ↗

FUNDED BY
0x61c3fcdD...7c0F2A87a at txn 0x997df64e0...

Fig. 5. Screenshot of an Etherscan address detail page. The page includes the address label and category; if no label is available, the corresponding field remains empty.

Table 1
Categorization standards and label distribution in the SP-dataset.

Category	Name	Number
CEX	Binance, Bitfinex, Kraken, KuCoin, HitBTC, Yobit, MEXC, HTX, OKX, Gate.io, etc.	1,815
DEX	Uniswap, Curve.fi, Sushiswap, MetaSwap, FixedFloat, KyberSwap, SwapLynx, CREX, etc.	196,583
Cross-Chain	Multichain, Avalanche, Zora: Bridge, CBridge: Bridge, Arbitrum: Bridge, Synapse, etc.	2,513
Token	Wrapped Ether, Wrapped Bitcoin, Tether, Shiba, Livepeer: LPT Token, ERC20, etc.	360,683
Mixer	Tornado.Cash: 100 ETH, Tornado.Cash: Proxy, MixIt, etc.	140
NFT	Seaport, Opensea, PhantaBear, Oncyber, ERC721, Gem, etc.	89,757
Others	Wallet, Other DeFi, DApp, Tool, Loan, etc.	573,088

3.3. SPD-GNN

Despite the integration of multiple public label sources and the retrieval of address tags during the tracking process, the constructed SP-Dataset may still miss service addresses that are involved in money laundering but remain unlabeled. Traditional approaches for identifying such unlabeled SPs typically rely on fixed heuristic rules based on historical transaction volumes (Wu et al., 2024) or on machine learning classifiers (Gómez et al., 2022). However, heuristic methods lack adaptability to evolving laundering strategies, and machine learning models often exhibit low performance and are sensitive to the quality and distribution of training data.

To address these limitations, we propose SP-Graphs, transaction subgraphs centered on labeled SPs extracted from laundering paths. Based on these subgraphs, we design SPD-GNN, a graph neural network with attention mechanisms that learns structural and behavioral patterns for SP identification.

SP-Graph: An SP-Graph is a local subgraph centered on a labeled SP address, representing the upstream transaction structure leading to that SP within a money laundering path. Its construction is designed to capture representative transactional behaviors of labeled SPs within money laundering cases, thereby improving the quality and diversity of the training dataset for unlabeled SPs identification. To achieve this, we first conduct a preliminary tracking phase using the UMDG-based method to construct pre-built money laundering fund flow graphs, combined with a heuristic that identifies unlabeled SP addresses as those with more than 1000 historical transactions (Wu et al., 2024).

In this phase, we apply the tracking process to real-world laundering cases and record all labeled SPs that appear along the identified paths. For each selected labeled SP, we use it as a root node and perform a breadth-first backward traversal up to n hops through the laundering path to construct its corresponding SP-Graph. To improve representativeness and reduce redundancy in the training data, we extract at most one SP-Graph per SP address per case. To further increase dataset diversity, we additionally sample SP-Graphs rooted at different ML addresses within the same case.

As is shown in Algorithm 2, the detailed construction process consists of four key steps:

Algorithm 2 SP-Graph construction.

```
Require: Fund flow graph $G_{\text{flow}}$, depth limit $n$, root node $v_{\text{root}}$
Ensure: SP-Graph $G_{\text{sp}}$
 1: Initialize queue $Q \leftarrow \{(v_{\text{root}}, 0)\}$, visited node set $\mathcal{V} \leftarrow \emptyset$
 2: while $Q \neq \emptyset$ do
 3:     Pop $(v, d)$ from $Q$; add $v$ to $\mathcal{V}$
 4:     if $d < n$ then
 5:         for each predecessor $u$ of $v$ in $G_{\text{flow}}$ do
 6:             if $u \notin \mathcal{V}$ then
 7:                 $Q \leftarrow Q \cup \{(u,\ d+1 \text{ if } u \text{ is ML address})\}$
 8:             end if
 9:         end for
10:     end if
11: end while
12: $G_{\text{sp}} \leftarrow$ subgraph of $G_{\text{flow}}$ induced by $\mathcal{V}$; annotate node features
13: return $G_{\text{sp}}$
```

1. **Initialization:** The SP-Graph is constructed based on the fund flow graph G_{flow} obtained from ETTracker without SPD-GNN. The root node is defined as a labeled SP or ML address during training, and as an unlabeled candidate address during real-world tracking. A backward traversal depth n is specified to limit the expansion scope.
2. **Backward traversal:** Starting from the root node, a backward breadth-first traversal is conducted over the fund flow graph. The traversal depth increases only when passing through ML addresses, while SP addresses are traversed without affecting depth. The process continues until each backward path contains exactly n ML addresses.
3. **SP-Graph construction:** An induced subgraph is extracted from G_{flow} based on all nodes visited during traversal. This subgraph forms the SP-Graph and serves as the structural input for classifying the root node.
4. **Feature construction:** Each node in the SP-Graph is enriched with behavioral and account-level features (Table 2), including transaction frequency, volume, stablecoin usage, and smart contract indicators.

Table 2
Node features in SP-Graphs.

Category	FID (Num)	Description
Txn Counts	F0-F17 (18)	Total counts, frequency, in-degree, out-degree, lifecycle, and associated addresses for EOA, internal ETH, and token transactions
Txn Value	F18-F26 (9)	Total value, inflow, and outflow for EOA, internal ETH, and token transactions
Stablecoin	F27-F29 (3)	Counts, amounts, and types of stablecoin transactions
IsContract	F30 (1)	Whether the node is a Smart Contract

Table 3
SPD-GNN architecture and training configuration.

Component	Details	Parameters
Layers		
Input Transform	Linear + ReLU	31→64, Dropout = 0.3
GATConv (×2)	Hidden layers with attention	64→64, Heads = 4, Dropout = 0.3
BatchNorm + ELU	After each hidden layer	Momentum = 0.1
GATConv (Output)	Final attention layer	64→32, Heads = 1
Classifier	Linear projection	32→2, Kaiming init
Optimization		
Optimizer	AdamW	lr = 0.001, weight_decay = 5e-4
Scheduler	Cosine annealing	T_0 = 10, T_{mult} = 2
Loss	BCEWithLogitsLoss	pos_weight = $n_{\text{neg}}/n_{\text{pos}}$
Regularization	Skip connections, Dropout	Grad clip norm = 1.0

As is shown in Table 2, the node-level feature design of SP-Graphs is summarized, comprising a comprehensive set of attributes that reflect both transactional behavior and structural properties of each address. The feature set includes count-based statistics (F0–F17), such as frequency, in-degree, out-degree, lifecycle, and the number of distinct counterparties, computed separately for EOA, internal ETH, and token transfers. Value-oriented features (F18–F26) quantify total inflows and outflows for each transfer type. To further capture asset-specific behavior, a subset (F27–F29) focuses on stablecoin usage, including count, volume, and token type. Finally, a binary indicator (F30) specifies whether the address is a smart contract, highlighting the widespread use of contract-based infrastructure among SPs.

SPD-GNN: SPD-GNN is a binary classifier that determines whether a candidate address encountered during tracking is an SP, especially if it is unlabeled. According to taint-based tracking principles, if an address is identified as an SP or shows typical SP transaction patterns, tracking is halted to avoid including transactions from unrelated users. This strategy effectively reduces false positives caused by service aggregation.

To support reliable decision-making in such cases, SPD-GNN leverages both transaction structure and local address features to improve classification accuracy. As is shown in Table 3, SPD-GNN is based on a two-layer GATConv architecture (Veličković et al., 2018) with multi-head attention and a final linear classifier. Given the heterogeneous and locally sparse nature of SP-Graphs, the attention mechanism selectively aggregates informative neighbor features, while skip connections (Weber et al., 2019) mitigate oversmoothing and enhance message propagation across low-degree nodes. Feature extraction incorporates dropout and batch normalization to improve generalization. The model is optimized using AdamW with cosine annealing scheduling and trained with a weighted binary cross-entropy loss to address class imbalance.

For the classification objective, we employ the binary cross-entropy loss:

$$\mathcal{L}_{\text{CE}} = -\frac{1}{N}\sum_{i=1}^{N}\left[w_1 y_i \log(\hat{y}_i) + w_0(1-y_i)\log(1-\hat{y}_i)\right]. \tag{4}$$

In taint-based AML analysis, recall is particularly critical for controlling the quality of fund tracking. Failure to identify key laundering intermediaries, particularly SPs, does not simply truncate the graph but instead causes uncontrolled propagation into irrelevant regions. When such addresses are mistakenly treated as regular nodes, the tracking process continues through them, introducing large volumes of benign transactions and unrelated entities. This leads to the graph explosion, significantly impairing the interpretability and efficiency of downstream analysis.

Since effective tracking requires focusing on major laundering paths, recall-tuning is essential for filtering out noise-inducing extensions. To address this, SPD-GNN is specifically designed to emphasize recall by incorporating differential positive weighting and an auxiliary loss term to penalize false negatives:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{CE}} + \alpha\sum_{i=1}^{N} y_i(1-\hat{y}_i)^2, \tag{5}$$

where the recall-tuning coefficient α controls the contribution of false-negative penalization to the overall optimization objective.

4. Experiments

In this section, we conduct comprehensive experiments to evaluate the proposed ETTracker. Our experimental investigation is guided by the following Research Questions (RQs):

1. **RQ1: UMDG effectiveness:** What performance advantages does our UMDG-based approach provide compared to state-of-the-art transaction-level money laundering tracking algorithms?
2. **RQ2: SPD-GNN contribution:** How does the SPD-GNN trained on SP-Graphs improve detection accuracy, and how does it compare with other models in terms of both performance and efficiency?
3. **RQ3: Service provider analysis:** What characteristic patterns emerge among SPs across different money laundering cases?

4.1. *Experimental settings*

Environment: The experiments are conducted on a machine equipped with an Nvidia GeForce RTX 2080Ti GPU, Intel(R) Xeon(R) Silver 4210 CPU, and 128GB main memory. The programs are implemented in Python, and neural network models are developed using the PyTorch framework and the Deep Graph Library (DGL) for graph neural network operations.

Competitors: Although a number of works have explored fund tracking and AML in blockchain contexts, most of them focus on Bitcoin or general transaction networks. For Ethereum, to the best of our knowledge, XBlockFlow (Wu et al., 2024) remains the most representative effort. It provides a systematic investigation of major heist incidents from 2016 to 2022 based on taint analysis. Moreover, it lays the groundwork for the application of several AML-oriented fund tracking methods, such as Li et al. (2020), Lin et al. (2024), Sun et al. (2021), in the Ethereum ecosystem. Our method is conceptually based on XBlockFlow and significantly extends its core ideas. Therefore, we use XBlockFlow as the primary baseline in our evaluation. In addition, we benchmark SPD-GNN against classical machine learning models and ablated versions of itself to further validate its effectiveness.

As is shown in Table 4, we conduct a thorough evaluation on 11 complex money laundering cases, including PlusTokenPonzi, UpbitHack (as analyzed by XBlockFlow), and the recent Bybit cold wallet hack. The SPD-GNN training dataset consists of 6196 SP-Graphs derived from labeled SPs identified during ETTracker's initial tracking in these cases, comprising a total of 8,034,883 nodes, of which 398,056 are SP nodes.

4.2. *RQ1: UMDG effectiveness*

The main advantage of the UMDG-based approach is its ability to continue tracking across SPs during internal fund transfers, rather than stopping at labeled SPs or high-volume addresses as in traditional transaction-level algorithms. This overcomes a key limitation of static

Table 4
Cryptocurrency security incidents on Ethereum.

Case Name	Case Type	Year	Amount
AscendEX Hack	Exchange Hot Wallet Breach	2021	\$77.7M
BitMart Hack	Exchange Hot Wallet Breach	2021	\$150M
Bybit Exploit	Exchange Cold Wallet Breach	2025	\$1.5B
Cream Finance	DeFi Flash Loan Exploit	2021	\$130M
EasyFi Hack	DeFi Admin Key Compromise	2021	\$80M
Fake Metadium Presale	Fraudulent Token Sale	2018	\$20M
Harvest Finance	DeFi Flash Loan Exploit	2020	\$24M
Nexus Mutual Hack	CEO Personal Wallet Compromise	2020	\$8M
PlusToken Ponzi	Cryptocurrency Ponzi Scheme	2019	\$49M
Upbit Hack	Exchange Hot Wallet Breach	2019	\$50M
Vulcan Forged	Private Key Theft	2021	\$140M

heuristics that prematurely terminate tracking at intermediary entities. To assess the effectiveness of our method, we compare ETTracker and XBlockFlow on 10 complex money laundering cases, focusing on four metrics: the number of detected ML addresses, ML transactions, SP addresses, and labeled SPs.

To isolate the impact of UMDGs, we first conduct experiments without employing SPD-GNN for SP identification among ML addresses, while maintaining the same unlabeled SP identification criterion as XBlockFlow. Addresses with historical transaction volumes ($T_a > 1000$) are classified as unlabeled service addresses. This configuration, denoted as ETTracker (w/o SPD-GNN), ensures that the primary difference in fund tracking methodologies is solely the adoption of UMDGs.

To evaluate performance, we adopt four standard metrics: Precision, Recall, F1-score, and Money Coverage Ratio (MCR). Precision and Recall measure the accuracy and completeness of detected laundering entities, while F1-score balances both, and MCR reflects the proportion of illicit funds successfully covered. For consistency, all metrics are calculated using XBlockFlow's results as pseudo-ground truth.

As is shown in Table 5, the UMDG-based ETTracker (w/o SPD-GNN) achieves consistently high Recall and MCR across the evaluated cases. In most scenarios, the Recall approaches or reaches 1.0, indicating that the majority of laundering entities are successfully identified. Similarly, MCR remains high, reflecting effective coverage of illicit funds. In the HarvestFinance and EasyfiHacker cases, for example, ETTracker identifies 27 and 1858 ML addresses, associated with 2083 and 36,307 transactions, respectively. These results benefit from its ability to capture 28 and 341 cross-SP tracking instances in the two cases, contributing to a total of 2804 such instances across all cases. In EasyfiHacker, FakeMetadiumPresale, and HarvestFinance cases, the Precision is relatively low due to the large number of additional ML addresses detected by ETTracker. These addresses exceed those labeled by XBlockFlow, leading to lower Precision when its results are used as a reference.

We further analyze the performance from a different perspective by examining the ML address coverage and expansion rates of ETTracker (w/o SPD-GNN) relative to XBlockFlow across the 10 money laundering cases. As is shown in Fig. 6a, concerning coverage rates, ETTracker (w/o SPD-GNN) achieves 100 % ML address coverage in 7 cases while maintaining at least 50 % SP address coverage in 7 cases compared with XBlockFlow. Through manual verification, we determine that the incomplete ML address coverage in certain cases stems from XBlockFlow's inclusion of transactions from before the initial attack activities. For instance, in the UpbitHack case, the initial ML activities commence at block number 9,014,194, yet XBlockFlow's tracking results incorporate numerous transactions prior to this block. This observation indicates that XBlockFlow does not strictly perform post-fund-transfer tracking but rather captures all transaction addresses associated with the identified ML addresses. Similarly, the lower SP coverage rates can be attributed to XBlockFlow's methodology of including SPs that interact with unlabeled SPs within a one-week time window, which extends beyond the direct fund tracking paths.

As is shown in Fig. 6b, the expansion rates demonstrate ETTracker's ability to identify additional money laundering activities beyond the baseline. ETTracker (w/o SPD-GNN) successfully expands both ML and SP address detection across all 10 cases, with EasyfiHacker showing the highest expansion rates. This result is primarily attributed to the ability of UMDGs to enable tracking across SPs during internal transfers, which allows the system to follow laundering flows that span multiple SP addresses. In the EasyfiHacker case, this capability leads to more than a 100-fold increase in identified ML addresses.

To gain a more comprehensive understanding of the structural characteristics of money laundering operations, we further analyze the supplementary tracking results produced by the UMDG-based fund tracking method. Specifically, as is shown in Table 6, we report several key metrics: (1) the number of SPs involved in internal fund transfers (SP_I) and as terminal recipients (SP_O); (2) the total number of UMDGs constructed; (3) the number of candidate SP addresses selected for SPD-GNN detection (SPD); (4) two categories of fund recipient addresses (ML_L and ML_M); and (5) the types of transactions immediately preceding the ML addresses (NXT_E, NXT_I, NXT_T). Together, these statistics provide a detailed view of the underlying patterns and entities present in the tracked laundering flows.

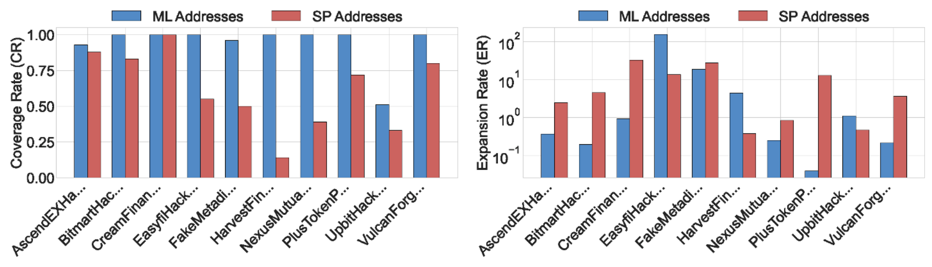
These statistics reveal distinct money laundering patterns across different cases. Compared to PlusTokenPonzi and UpbitHack, the BybitExploiter case exhibits a more complex UMDG structure, with a relatively small number of UMDGs supporting a large volume of transactions and fewer EOA transactions. Across all cases, the number of ML_L addresses consistently exceeds that of ML_M, suggesting a preference for transferring funds to leaf nodes rather than retaining them within intermediary addresses. This may indicate that launderers tend to offload funds to external endpoints to reduce traceability or to avoid using internal addresses that are less likely to be under their direct control.

In summary, the UMDG-based approach dramatically improves the coverage of ML addresses compared to traditional transaction-level tracking, capturing up to 153× more ML addresses in the largest case.

Table 5
Comparison of fund tracking performance: ETTracker and XBlockFlow.

Case	ETTracker (w/o SPD-GNN)								XBlockFlow (Wu et al., 2024)			
	Prec.	Rec.	F1	MCR	ML_A	ML_T	SP_A	SP_L	ML_A	ML_T	SP_A	SP_L
AscendEXHacker	0.71	0.91	0.8	0.89	94	3144	229	224	84	1992	197	78
BitmartHack	0.83	1.0	0.91	1.0	6	1842	122	122	5	924	24	24
BybitExploiter	–	–	–	–	19,495	474,890	321	288	–	–	–	–
CreamFinance	0.51	1.0	0.68	1.0	35	2874	265	249	18	90	20	9
EasyfiHacker	0.01	1.0	0.01	1.0	1858	36,307	909	813	12	2155	593	66
FakeMetadiumPresale	0.04	0.96	0.07	0.99	3143	60,643	777	686	160	572	29	29
HarvestFinance	0.19	1.0	0.31	1.0	27	2083	35	31	5	1813	78	72
NexusMutualHacker	0.8	1.0	0.89	1.0	10	477	50	48	8	956	42	42
PlusTokenPonzi	0.96	1.0	0.98	1.0	32,017	59,883	236	217	30,783	48,771	27	19
UpbitHack	0.59	0.52	0.55	0.42	19,766	119,231	3222	3010	17,232	890,324	8,904	7087
VulcanForged	0.82	1.0	0.9	1.0	28	2629	42	42	23	363	13	10

MCR: Money Coverage Ratio; $\mathbf{ML}_A$: ML Addresses; $\mathbf{ML}_T$: ML Transactions; $\mathbf{SP}_A$: SP Addresses; $\mathbf{SP}_L$: Labeled SP Addresses.



(a) Address coverage rate (CR) (b) Address expansion rate (ER)

Fig. 6. Coverage and expansion rates of ML and SP addresses comparing ETTracker (w/o SPD-GNN) with XBlockFlow.

Table 6
Additional fund tracking results of ETTracker (w/o SPD-GNN).

Case	UMDG	SP_I	SP_O	ML_L	ML_M	SPD	NXT_E	NXT_I	NXT_T
AscendEXHacker	761	118	111	183	0	2	102	22	151
BitmartHack	354	114	8	13	0	1	12	9	120
BybitExploiter	57,435	223	98	19,592	14	61	80,424	70,363	206,468
CreamFinance	369	230	35	68	0	1	274	238	705
EasyfiHacker	8217	341	568	2424	15	252	4966	4770	8327
FakeMetadiumPresale	21,676	228	549	3691	1	466	6920	7146	13,589
HarvestFinance	153	28	7	32	0	0	30	13	23
NexusMutualHacker	70	30	20	29	0	2	33	31	47
PlusTokenPonzi	51,647	75	161	32,176	21	21	36,705	4883	10,450
UpbitHack	73,622	769	2,453	22,219	49	1,502	22,343	7042	14,740
VulcanForged	508	27	15	42	0	3	71	44	99

UMDG: Number of UMDGs; $\mathbf{SP_I}$: SPs involved in internal fund transfers; $\mathbf{SP_O}$: SPs as terminal recipients. $\mathbf{ML_L}$, $\mathbf{ML_M}$: Leaf and Middle pattern fund recipient addresses; **SPD**: Candidate addresses for SPD-GNN. $\mathbf{NXT_E}$, $\mathbf{NXT_I}$, $\mathbf{NXT_T}$: ML_A from EOA, internal ETH, and token transactions.

Table 7
Performance comparison of different models.

Model	Accuracy	Precision	Recall	F1-Score
SPD-GNN(R)	0.9839 ± 0.0016	0.9767 ± 0.0007	**0.9933 ± 0.0003**	0.9849 ± 0.0008
SPD-GNN	**0.9849 ± 0.0006**	**0.9824 ± 0.0010**	0.9914 ± 0.0007	**0.9868 ± 0.0006**
EvolveGCN	0.9736 ± 0.0007	0.9232 ± 0.0023	0.9904 ± 0.0019	0.9556 ± 0.0009
SkipGCN	0.9691 ± 0.0008	0.9665 ± 0.0018	0.9800 ± 0.0023	0.9732 ± 0.0009
GCN	0.8426 ± 0.0027	0.7280 ± 0.0120	0.7209 ± 0.0077	0.7243 ± 0.0052
GAT	0.9660 ± 0.0012	0.9562 ± 0.0015	0.9857 ± 0.0017	0.9707 ± 0.0013
SAGE	0.9745 ± 0.0007	0.9255 ± 0.0036	0.9907 ± 0.0016	0.9570 ± 0.0015
XGBoost	0.9544 ± 0.0016	0.7925 ± 0.0047	0.9862 ± 0.0045	0.8788 ± 0.0042
Random Forest	0.9485 ± 0.0016	0.7698 ± 0.0053	0.9882 ± 0.0003	0.8654 ± 0.0033

This confirms the effectiveness of UMDG in extending fund tracing across SPs.

4.3. RQ2: SPD-GNN contribution

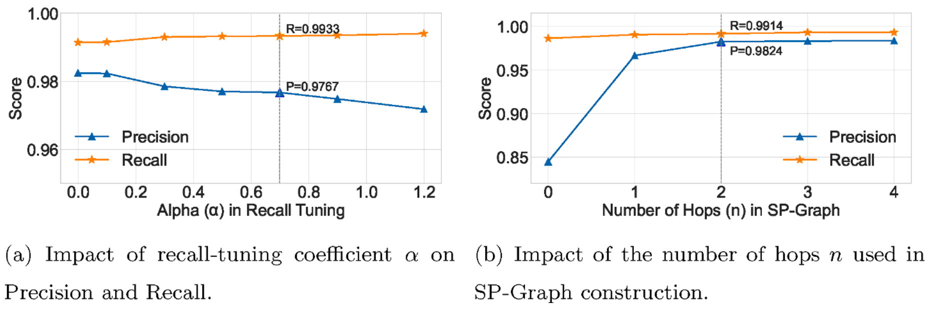
The second major contribution of ETTracker lies in the accurate identification of previously unlabeled SPs using the SPD-GNN model trained on SP-Graphs. To assess its effectiveness, we compare SPD-GNN with classical classifiers (XGBoost, Random Forest), a simplified variant (SPD-GNN(R)), and several GNN-based models. Among these, EvolveGCN (Pareja et al., 2020) incorporates temporal dependencies in evolving graphs, while SkipGCN (Weber et al., 2019) introduces skip connections to mitigate over-smoothing. As is shown in Table 7, we evaluate models using four standard metrics: Accuracy, Precision, Recall, and F1-score, with results averaged over 5-fold cross-validation.

SPD-GNN achieves the best overall performance, with an F1-score of 0.9868, precision of 0.9824, and recall of 0.9914. Compared to XGBoost and Random Forest, it yields a notable improvement in both precision and F1-score, indicating stronger control over false positives. Among GNN-based models, SPD-GNN also consistently outperforms its counterparts, surpassing EvolveGCN and SkipGCN by over 3 and 1 percentage points in F1-score, respectively.

For practical AML applications where high recall is essential, we adopt SPD-GNN(R) as the deployed model for unlabeled SP classification. Although it incurs a slight drop in precision, the gain in recall helps reduce the risk of missed detections, which is critical for avoiding untracked laundering paths and limiting the propagation of noisy transactions in downstream analysis. As is shown in Fig. 7a, we set the recall-tuning coefficient $\alpha = 0.7$ to achieve this balance, yielding a recall of 0.9933 while maintaining a relatively high precision of 0.9767.

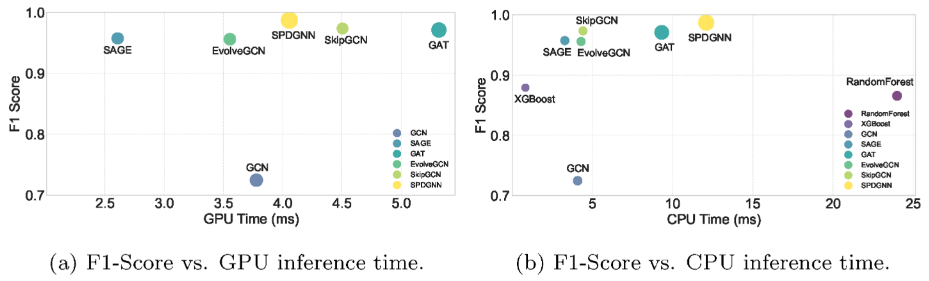
As is shown in Fig. 7b, the impact of the backward traversal depth n used in SP-Graph construction on model performance is illustrated. Increasing n from 0 to 2 significantly improves both precision and recall, as the model benefits from incorporating multi-step upstream transaction context. However, further increasing the depth beyond $n = 2$ results in diminishing returns, with minimal performance gains. Based on this trade-off, we set $n = 2$ as the default depth, which provides sufficient structural information while maintaining computational efficiency in both training and inference.

To ensure the practical applicability of SPD-GNN within ETTracker, we constrain SP-Graph construction based on the historical transaction volume of candidate addresses. Only those with 300 to 2000 transactions are passed to SPD-GNN for classification. Addresses with fewer than 300 transactions are treated as low-activity laundering addresses



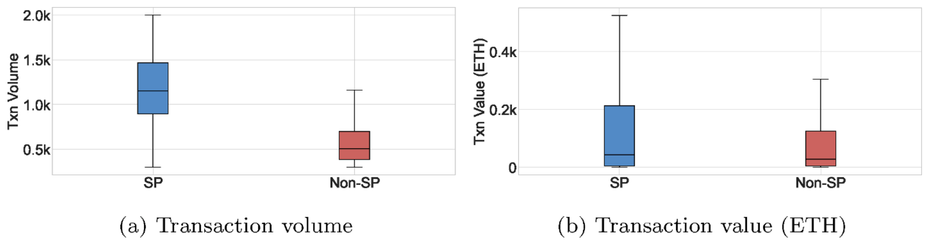
(a) Impact of recall-tuning coefficient α on Precision and Recall. (b) Impact of the number of hops n used in SP-Graph construction.

Fig. 7. Effect of key hyperparameters in SPD-GNN on model performance.



(a) F1-Score vs. GPU inference time. (b) F1-Score vs. CPU inference time.

Fig. 8. Comparison of different models in terms of performance and inference efficiency.



(a) Transaction volume (b) Transaction value (ETH)

Fig. 9. Distribution of transaction volume and value in SPD-GNN identification results.

and are excluded from further SP analysis, while those with more than 2000 transactions are omitted to limit computational overhead and avoid graph explosion. These thresholds are empirically grounded: prior work (Wu et al., 2024) on the Upbit Hack incident demonstrated that laundering addresses rarely exceed 603 transactions, and proposed a 1,000-transaction cutoff for potential SP identification.

As is shown in Fig. 8, SPD-GNN achieves the highest F1-score among all models and maintains inference times on GPU that are comparable to most baselines, outperforming models including GAT and SkipGCN. Its slightly higher CPU latency is due to the attention mechanism, which involves iterative neighborhood aggregation and weight computation. Profiling of the VulcanForged case shows that SP-Graph construction and SPD-GNN prediction together account for only 0.06 % of the total runtime, confirming that model inference is not a bottleneck. The majority of processing time is spent on UMDG construction (53.95 %, 287.24 seconds), address transaction retrieval (31.09 %, 165.54 seconds), and label lookups (12.64 %, 67.29 seconds), with a total runtime of 532.40 seconds. These findings indicate that transaction and label retrieval are the primary sources of tracking latency.

As is shown in Fig. 9, the SPD-GNN classification results from the perspectives of transaction volume and ETH transaction value are presented. Addresses identified as SPs exhibit higher historical transaction volumes, averaging approximately 1,200, while non-SPs average around 500. A similar contrast is observed in transaction value distributions. Unlike XBlockFlow, which relies on a fixed threshold of 1000 transactions to detect potential SPs, our method incorporates structural transaction patterns and richer address features for more adaptive classification. These results highlight the effectiveness of integrating SPD-GNN into the fund tracking process. By enabling more nuanced identification based on both structure and features, ETTracker improves SP detection accuracy and enhances tracking performance in complex laundering scenarios.

As is shown in Table 8, the impact of SPD-GNN on fund tracking is presented by comparing the results of ETTracker before and after SPD identification. However, the contribution of SPD-GNN is not simply reflected in the net increase or decrease of ML_A or SP_A, but rather in its ability to make more informed decisions at critical points in the tracking process. To quantify this impact, we calculate the Jaccard similarity coefficients between ML address sets (J_{ML}) and SP address sets (J_{SP}) before and after applying SPD-GNN. The Jaccard coefficient measures the proportion of shared elements between two sets, defined as $J(A, B) = |A \cap B|/|A \cup B|$. Lower similarity values indicate greater changes in address identification caused by SPD-GNN. Among the cases, FakeMetadiumPresale shows the most pronounced shift, with J_{ML} and J_{SP} dropping to 0.669 and 0.633, respectively, suggesting a significant correction in early-stage classification decisions.

In summary, the SPD-GNN model significantly enhances unlabeled SP detection accuracy (achieving 98.7 % F1-score, about 1–3 % higher than baseline models), with an acceptable runtime overhead. This confirms that SPD-GNN's inclusion improves overall tracking performance without sacrificing efficiency.

4.4. RQ3: Service provider analysis

This section addresses RQ3 by analyzing the behavioral patterns of SPs across 11 money laundering cases, based on tracking results from

Table 8
Fund tracking results of ETTracker (w/ SPD-GNN) and similarity comparison with ETTracker (w/o SPD-GNN).

Case	ETTracker (w/ SPD-GNN)						w/o SPD-GNN	
	ML_A	ML_T	SP_A	SP_L	SPD_T	SPD_F	J_{ML}	J_{SP}
AscendEXHacker	73	3144	229	224	1	2	0.986	0.902
BitmartHack	6	1842	122	122	0	1	1.0	1.0
BybitExploiter	19,357	472,336	320	286	18	130	0.991	0.997
CreamFinance	35	2874	265	249	0	1	1.0	1.0
EasyfiHacker	2302	33,669	1042	905	122	189	0.683	0.798
FakeMetadiumPresale	4178	95,026	1107	972	179	470	0.669	0.633
NexusMutualHacker	10	482	50	48	0	2	1.0	1.0
PlusTokenPonzi	31,930	58,236	92	84	10	15	0.991	0.836
UpbitHack	19,141	118,393	2748	2660	671	820	0.761	0.711
VulcanForged	28	2558	43	43	1	2	1.0	0.833

SPD_T, **SPD_F**: True positive and false positive addresses identified by SPD-GNN;
J_{ML}, **J_{SP}**: Jaccard similarity coefficients of ML and SP address sets before and after applying SPD-GNN.

Table 9
Top 3 SPs involved in internal fund transfers (SP_I) and those acting as terminal recipients across UMDGs (SP_O) in each case.

Case	Top 3 SPs involved in internal fund transfers (SP_I)	Top 3 SPs acting as terminal recipients across UMDGs (SP_O)
AscendEXHacker	Wrapped Ether, Uniswap V3: USDC 3, UniswapV3	Uniswap V3: Router 2, 0x: Exchange Proxy, Metamask: Swap Router
BitmartHack	Tornado.Cash: Proxy, Wrapped Ether, Aggregation Router V4	Tornado.Cash: 100 ETH, PolkaBridge: Deflationary Farming Token, Tornado.Cash: 10 ETH
BybitExploiter	Wrapped Ether, UniswapV3, Uniswap V3: USDC 3	THORChain Router, DEXRouter, OKX: Web3 Proxy
CreamFinance	Paraswap v5: Augustus Swapper, Wrapped Ether, CoW Protocol: GPv2Settlement	Curve.fi: DAI/USDC/USDT Pool, Uniswap V3: Router, SushiSwap: Router
EasyfiHacker	Wrapped Ether, Maestro: Router 2, UniswapV2	Uniswap V2: Router 2, Uniswap: Universal Router, Uniswap: Universal Router 2
FakeMetadiumPresale	Wrapped Ether, Fake: Metadium Presale, THORChain_Router	Blur: Bidding, Uniswap: Universal Router, Blur.io: Marketplace 3
HarvestFinance	Curve.fi: y Swap, Yearn: yUSDT Token, Yearn: yUSDC Token	Tornado Cash Recipient, Ren: BTC Gateway, Tornado.Cash: 100 ETH
NexusMutualHacker	KyberNetwork, Kyber: Proxy 2, KyberFprReserveV2	1inch v2: Aggregation Router, 0x: Exchange v3, Kyber: Reserve
PlusTokenPonzi	Wrapped Ether, Uniswap V3: USDC 3, Uniswap V3: Router 2	Uniswap: Universal Router, UniswapV3, Aggregation Router V4
UpbitHack	Wrapped Ether, MetaMask: Swaps Spender, Kyber: Contract	Uniswap V2: Router 2, Uniswap V2: USDT, Uniswap: Universal Router
VulcanForged	Wrapped Ether, UniswapV2, UniswapV3	GovernanceLeftoverExchanger, FixedFloat, Tornado.Cash: 100 ETH

Table 10
Detailed information of Top 7 SPs across all money laundering cases.

Address	Label	Category	Txns	Description
0xc02aaa39b…cc2	Wrapped Ether	Token	9283	ERC-20 compliant token that wraps ETH for DeFi compatibility and DEX trading.
0xd37bbe57…146	THORChain Router	Cross-Chain	7472	Cross-chain protocol gateway facilitating trustless asset transfers between blockchains.
0x7a250d56…88d	Uniswap V2 Router 2	DEX	7324	Core routing contract for Uniswap V2 providing optimal swap paths with slippage protection.
0x4042a04c…30b	UniswapV2	DEX	5062	Automated market maker implementing constant product formula for permissionless token swaps.
0xcb488b84…2f4	UniswapV3	DEX	3379	Next-generation DEX pool implementing concentrated liquidity within customizable price ranges.
0x3fc91a3a…fad	Uniswap Universal Router	DEX	1953	Advanced aggregation contract optimizing routes across multiple Uniswap protocol versions.
0x00000000…1ac	Blur: Bidding	NFT	1080	Specialized contract handling auction bid mechanics for the Blur NFT marketplace.

ETTracker (w/ SPD-GNN) and labeled data from SP-Dataset. As is shown in Table 9, the Top 3 SPs in each case are presented, including SPs involved in internal fund transfers (SP_I) and those acting as terminal recipients across UMDGs (SP_O). SP_I refers to SPs that appear within UMDGs but are not reached by tainted flows from labeled ML addresses, while SP_O represents SPs that receive funds directly from such addresses. These two categories reflect distinct roles within laundering structures. As is shown in Table 10, the top 7 SPs aggregated across all cases are listed, along with their categories and descriptions.

The statistical results reveal distinctive patterns in SP utilization across money laundering operations. Within SP_I, Wrapped Ether is extensively employed by money launderers to facilitate ETH-to-WETH conversions, enabling subsequent token exchanges on DEXs. The high proportion of Uniswap decentralized exchanges among SP_I addresses further corroborates this finding. In contrast, SP_O patterns reflect SPs with which launderers directly interact. The exceptionally high interaction frequency with DEX routers, such as Uniswap V2: Router 2, demonstrates launderers' preference for token exchange services rather than solely relying on direct ETH transfers. Furthermore, the significant prevalence of Tornado.Cash mixing services, cross-chain services like PolkaBridge, and NFT marketplaces such as Blur highlight the complexity of money laundering activities and the diversity of fund transfer mechanisms employed.

As is shown in Fig. 10, density plots are presented for interactions with SP_I and SP_O along the complete transaction paths of 10 money laundering cases, providing deeper insights into SP interaction frequency during different money laundering phases. The horizontal axis represents normalized UMDG IDs, which indicate the tracking sequence rather than the chronological order of ML transactions. The visualization reveals distinct patterns of SP interactions across various money laundering schemes. Cases like VulcanForged and HarvestFinance exhibit intensive SP_I interactions during placement phases followed by concentrated SP_O interactions in the integration stages. In contrast, cases like EasyfiHacker and FakeMetadiumPresale demonstrate synchronized interactions with both SP_I and SP_O throughout the entire process, indicating an integrated approach to fund obfuscation. PlusTokenPonzi represents a highly typical case where numerous obfuscation addresses are utilized during the layering phase for money laundering, followed by extensive interactions with SPs during the integration phase to facilitate fund extraction.

As is shown in Fig. 11, the distribution of various SP types involved in the 11 money laundering cases is presented, building upon our

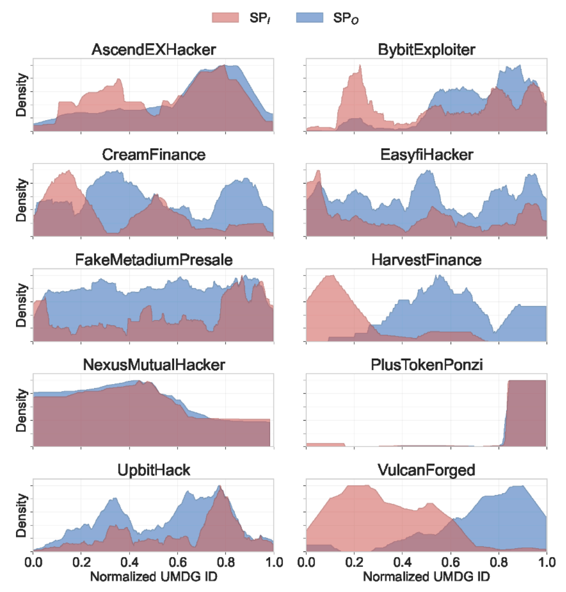


Fig. 10. SP interaction density along tracking paths across various money laundering cases.

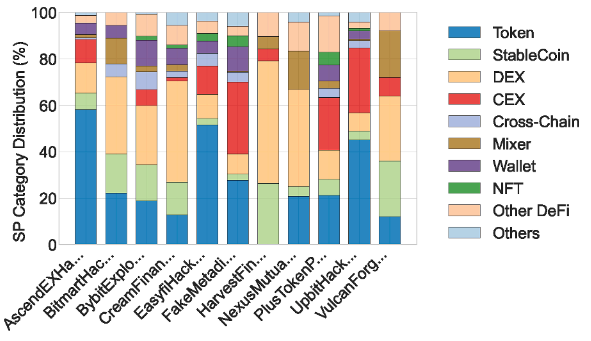


Fig. 11. Distribution of SP categories across money laundering cases.

functional classification of SPs in the SP-Dataset. To enhance clarity, we separate Stablecoin from the Token category when calculating the proportion of SP types. The visualization reveals that Token and DEX services consistently account for more than 50 % of interactions in all cases, demonstrating the prevalence of token minting and currency exchanges in money laundering operations. The substantial proportion of CEX and Wallet services further indicates launderers' intention to eventually convert cryptocurrencies into fiat currencies. The varying proportions of Mixer services across different cases reflect diverse anonymization strategies employed by different launderers, suggesting customized approaches to transaction obfuscation. Although Cross-Chain services represent a relatively small percentage of the interactions, they facilitate significant fund transfers that are particularly challenging to track due to their inter-blockchain nature.

In summary, across the 11 cases, distinct patterns emerge in SP usage: launderers consistently leverage decentralized exchanges and token bridges (e.g., Uniswap, 0x, Wrapped Ether) to obfuscate funds internally, and frequently utilize mixers, cross-chain bridges, or certain CEX platforms as terminal outlets.

5. Discussion and future work

5.1. Discussion

Our analysis identifies three principal challenges in ETTracker's implementation and the corresponding mitigation strategies: optimizing detection speed, defining effective tracking termination criteria, and ensuring model adaptability in the presence of data limitations.

Performance optimization: ETTracker's address-centric and label-intensive tracking requires frequent retrieval of EOA transactions, internal ETH transfers, token transactions, and address labels. Querying a local transaction database for a single address typically takes about 5 seconds, which is significantly slower than Etherscan's average 1-second response. To improve efficiency, ETTracker adopts a space-time tradeoff by using Etherscan as the primary data source and locally caching all fetched transactions and address labels. Subsequent queries access the cache directly, which reduces redundant network requests and tracking latency. This approach ensures that transaction data for each address is retrieved only once, optimizing both query speed and storage usage.

Label acquisition from Etherscan is inherently slower, as it involves simulating browser-based page scraping rather than direct API access. Retrieving label information for all identified ML addresses would impose unacceptable delays on the tracking process. To address this, ETTracker prioritizes Etherscan lookups only for two specific scenarios: (1) addresses with no existing label in the SP-Dataset, and (2) addresses with conflicting labels from multiple sources. Once acquired, each address's label metadata is stored locally, ensuring that repeated requests for the same address are avoided. This optimization significantly improves the overall efficiency of ETTracker while preserving label freshness and integrity in critical cases.

Model adaptability: As a data-driven learning model, SPD-GNN is susceptible to potential bias if the training dataset contains label inaccuracies or reflects class imbalance. To mitigate this, ETTracker aggregates labels from multiple verified sources and prioritizes behavioral features that generalize across service types. Additionally, to minimize the risk of false negatives (i.e., missing unidentified SPs), the model is tuned to favor higher recall, accepting some false positives that are further filtered through post-detection validation. Compared to static heuristic methods, SPD-GNN offers enhanced flexibility, as it adapts to newly emerging laundering behaviors that deviate from known patterns, making it more resilient to evolving financial tactics.

Fund tracking termination criteria: In practice, fund tracking may encounter significant convergence challenges, as observed in cases like UpbitHack. In such scenarios, the queue of ML addresses does not diminish as expected but instead grows uncontrollably. Manual analysis reveals several factors contributing to this non-convergence phenomenon.

First, the performance limitations of SPD-GNN present a fundamental challenge. Despite achieving recall rates exceeding 99.3 %, the remaining margin of error leads to occasional SP misclassification, inadvertently introducing mixed collections of thousands of laundering and legitimate transactions into the tracking process. Second, certain ML addresses operate as multi-purpose service addresses, simultaneously handling legitimate transactions alongside illicit activities. This dual functionality introduces erroneous fund volume assessments during normal tracking procedures, complicating the isolation of fund flows. Third, money laundering operations frequently combine off-chain and on-chain transaction mechanisms. For example, when money launderers directly sell assets from ML addresses to legitimate traders through external channels, subsequent tracking and visualization efforts become fundamentally disconnected from the actual money laundering fund movement patterns. Based on this analysis, we establish a termination criterion for ETTracker that halts the fund tracking process when the total number of addresses that have entered the address queue Q exceeds 100,000 entries. This threshold prevents unbounded expansion of tracking graphs that would otherwise impose excessive computational burden.

5.2. Future work

While ETTracker's taint analysis reconstructs a comprehensive graph of ML-related addresses, AML operations typically focus on dominant fund flows rather than exhaustive coverage. To address this, we plan to develop methods for detecting core laundering nodes within the graph, enabling more targeted analysis and improved operational efficiency. We also intend to compare our approach with frameworks such as FlowScope and DenseFlow to further evaluate its effectiveness and scalability in identifying high-impact laundering entities.

6. Related work

6.1. Anti-money laundering in blockchain

Detecting money laundering in blockchain environments, particularly within the Ethereum ecosystem, has become a critical research focus at the intersection of cybersecurity and financial crime prevention (Almeida et al., 2023; Kolachala et al., 2021; Zhou et al., 2024). Recent studies have predominantly leveraged network analysis (Lv et al., 2023; Marasi & Ferretti, 2024; Zhou et al., 2022), pattern recognition (Yu et al., 2023), and deep learning techniques (Li et al., 2024; Lin et al., 2022) to identify illicit transactions on blockchain platforms.

Significant advances in Ethereum-based AML detection have emerged through recent academic contributions. Wu et al. (2024) examine the evolution of laundering patterns within the Web3 ecosystem, uncovering sophisticated techniques such as counterfeit token creation and speculator impersonation. Furthermore, Fu et al. (2023) extend this understanding through their analysis of the Upbit Exchange security breach, drawing critical comparisons between traditional and blockchain-based laundering patterns. On the methodological front, Lin et al. (2024) introduce DenseFlow, combining dense subgraph detection with maximum flow algorithms to achieve a 16.34 % precision improvement across multiple datasets. Liu et al. (2023) advance the field with GTN2vec, a graph embedding algorithm that incorporates temporal transaction data and gas price information, demonstrating superior feature extraction capabilities compared to conventional approaches.

6.2. Service providers in blockchain

Privacy preservation and deanonymization in blockchain mixing services have emerged as critical research areas (Wu et al., 2021; Xu et al., 2023). Substantial efforts have focused on analyzing Tornado Cash, a prominent Ethereum mixing protocol. Recent studies have addressed this challenge primarily through two main approaches: heuristic-based analysis (Wu et al., 2022a) and machine learning methods (Hu et al., 2024). There is also growing emphasis on constructing and validating ground-truth datasets for robust evaluation (Miedema et al., 2023).

Specifically, Wu et al. (2022b) advance privacy analysis in blockchain systems by implementing expert heuristics to assess the true anonymity of Ethereum addresses through address clustering, compromise detection, and accurate computation of Tornado Cash mixer pool sizes, thus addressing limitations in pseudonymous transaction privacy. Subsequently, to investigate money laundering patterns in Tornado Cash, Youn et al. (2023) employ clustering analysis of deposit addresses and examine NFT phishing incidents, providing insights into criminal cryptocurrency flows through mixing services and quantifying associated damages. Through systematic analysis of on-chain data and transaction patterns in Tornado Cash, Tang et al. (2022) propose three heuristic clustering rules to identify linked user addresses, demonstrating how inappropriate mixing behaviors can compromise transaction privacy and reduce anonymity set sizes. Furthermore, Du et al. (2024) address the challenge of deanonymizing mixing transactions in Tornado Cash through MixBroker, a graph neural network framework that leverages Mixing Interaction Graphs and node-pair link prediction to correlate mixing addresses.

7. Conclusion

This paper presents ETTracker, a novel framework for AML fund tracking on Ethereum. ETTracker introduces three key technical contributions: (1) the UMDG structure, which enables tracking across SPs during internal transfers; (2) SP-Dataset, a dynamically updated and categorized dataset of SP addresses; and (3) SPD-GNN, a graph-based model for accurate identification of previously unlabeled SPs. Compared to existing state-of-the-art methods, ETTracker achieves superior performance in ML address coverage, SP label completeness, and SP classification accuracy. These results demonstrate the effectiveness of ETTracker in enhancing AML fund tracking. In addition, a large-scale analysis of SP activity across multiple laundering cases reveals characteristic patterns in the types of services commonly exploited for illicit financial operations.

CRediT authorship contribution statement

Changhao Wu: Conceptualization, Methodology, Software, Writing – original draft preparation; **Junwei Xu:** Data curation, Writing – original draft preparation, Visualization; **Kai Wang:** Visualization, Writing – reviewing & editing; **Weili Han:** Project administration, Supervision; **Hongfeng Chai:** Supervision.

Data availability

I have shared the link to my data at the Attach File step

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

This paper is supported by National Key Research and Development Program of China (Grant No. 2022YFC3320902).

References

Almeida, H., Pinto, P., & Vilas, A. F. (2023). A review on cryptocurrency transaction methods for money laundering. In *Proceedings of the 5th international conference on finance, economics, management and IT business (FEMIB 2023)* (pp. 114–121).
Buterin, V. (2013). Ethereum white paper: A next generation smart contract & decentralized application platform. Online. Accessed: 2024-11-15. https://github.com/ethereum/wiki/wiki/White-Paper.
Chen, W., Zhang, T., Chen, Z., Zheng, Z., & Lu, Y. (2020). Traveling the token world: A graph analysis of Ethereum ERC20 token ecosystem. In *Proceedings of the web conference 2020 (WWW 2020)* (pp. 1411–1421).
Du, H., Che, Z., Shen, M., Zhu, L., & Hu, J. (2024). Breaking the anonymity of Ethereum mixing services using graph feature learning. *IEEE Transactions on Information Forensics and Security*, *19*, 616–631.
Dune (2025). Dune analytics: Blockchain data platform. https://dune.com. Accessed: 2025-03-15.
Etherscan (2025). Etherscan: Ethereum blockchain explorer. https://etherscan.io/. Accessed: 2025-03-01.
Farrugia, S., Ellul, J., & Azzopardi, G. (2020). Detection of illicit accounts over the Ethereum blockchain. *Expert Systems with Applications*, *150*, 113318.
Fu, Q., Lin, D., Cao, Y., & Wu, J. (2023). Does money laundering on Ethereum have traditional traits? In *2023 IEEE International symposium on circuits and systems (ISCAS)* (pp. 1–5).
Fu, Q., Lin, D., Wu, J., & Zheng, Z. (2024). A general framework for account risk rating on Ethereum: Toward safer blockchain technology. *IEEE Transactions on Computational Social Systems*, *11*(2), 1865–1875.
Gómez, G., Moreno-Sanchez, P., & Caballero, J. (2022). Watch your back: Identifying cybercrime financial relationships in bitcoin through back-and-forth exploration. In *Proceedings of the 29th ACM SIGSAC conference on computer and communications security (CCS 2022)* (pp. 1291–1305).
Harris, C. G. (2023). Cross-chain technologies: Challenges and opportunities for blockchain interoperability. In *2023 IEEE International conference on omni-layer intelligent systems (COINS)* (pp. 1–6).
Hu, X., Gui, M., Cheng, G., Li, R., & Wu, H. (2024). Multi-class bitcoin mixing service identification based on graph classification. *Digital Communications and Networks*, *10*(6), 1881–1893.
Kolachala, K., Simsek, E., Ababneh, M., & Vishwanathan, R. (2021). SoK: Money laundering in cryptocurrencies. In *Proceedings of the 16th international conference on availability, reliability and security (ARES 2021)* (pp. 5:1–5:10).

Li, X., Liu, S., Li, Z., Han, X., Shi, C., Hooi, B., Huang, H., & Cheng, X. (2020). FlowScope: Spotting money laundering based on graphs. In *Proceedings of the 34th AAAI conference on artificial intelligence (AAAI 2020)* (pp. 4731–4738).

Li, Z., Yao, R., Yang, D., Zhang, Y., Mao, H., & Yuan, Y. (2024). BELFAL: A blockchain-based ensemble learning framework for anti-money laundering in crypto-currency markets. In *2024 IEEE 30th International conference on parallel and distributed systems (ICPADS)* (pp. 520–527).

Lin, D., Wu, J., Xuan, Q., & Tse, C. K. (2022). Ethereum transaction tracking: Inferring evolution of transaction networks via link prediction. *Physica A: Statistical Mechanics and its Applications*, *600*, 127504.

Lin, D., Wu, J., Yu, Y., Fu, Q., Zheng, Z., & Yang, C. (2024). DenseFlow: Spotting cryptocurrency money laundering in Ethereum transaction graphs. In *Proceedings of the ACM web conference 2024 (WWW '24)* (pp. 4429–4438).

Liu, J., Yin, C., Wang, H., Wu, X., Lan, D., Zhou, L., & Ge, C. (2023). Graph embedding-based money laundering detection for Ethereum. *Electronics*, *12*(14), 3180.

Luo, J., Qin, J., Wang, R., & Li, L. (2024). A phishing account detection model via network embedding for Ethereum. *IEEE Transactions on Circuits and Systems II: Express Briefs*, *71*(2), 622–626.

Lv, W., Liu, J., & Zhou, L. (2023). Detection of money laundering address over the Ethereum blockchain. In *2023 5th International conference on frontiers technology of information and computer (ICFTIC)* (pp. 866–869).

Marasi, S., & Ferretti, S. (2024). Anti-money laundering in cryptocurrencies through graph neural networks: A comparative study. In *2024 IEEE 21st Consumer communications & networking conference (CCNC)* (pp. 272–277).

Miedema, F., Lubbertsen, K., Schrama, V., & van Wegberg, R. (2023). Mixed signals: Analyzing ground-truth data on the users and economics of a bitcoin mixing service. In *32nd USENIX Security symposium (USENIX security 23)* (pp. 751–768).

Möser, M., Böhme, R., & Breuker, D. (2014). Towards risk scoring of bitcoin transactions. In *Financial cryptography and data security – FC 2014 workshops, bitcoin and WAHC 2014* (pp. 16–32). (*vol. 8438*). Lecture Notes in Computer Science.

Pareja, A., Domeniconi, G., Chen, J., Ma, T., Suzumura, T., Kanezashi, H., Kaler, T., & Leiserson, C. E. (2020). EvolveGCN: Evolving graph convolutional networks for dynamic graphs. In *Proceedings of the 34th AAAI conference on artificial intelligence (AAAI 2020)* (pp. 5363–5370).

Rattanabunno, S., Werapun, W., Suaboot, J., Karode, T., & Puongmanee, M. (2023). An EOA identity tracing system (AITS) on Ethereum blockchain. In *2023 8th International conference on information and network technologies (ICINT)* (pp. 61–65).

Sun, X., Zhang, J., Zhao, Q., Liu, S., Chen, J., Zhuang, R., Shen, H., & Cheng, X. (2021). CubeFlow: Money laundering detection with coupled tensors. In *Advances in knowledge discovery and data mining – 25th pacific-Asia conference (PAKDD 2021)* (pp. 78–90). (*vol. 12712*). Lecture Notes in Computer Science.

Tang, Y., Xu, C., Zhang, C., Wu, Y., & Zhu, L. (2022). Analysis of address linkability in tornado cash on Ethereum. In *Cyber security* (pp. 39–50).

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., & Bengio, Y. (2018). Graph attention networks. In *6th International conference on learning representations (ICLR 2018)*.

Weber, M., Domeniconi, G., Chen, J., Weidele, D. K. I., Bellei, C., Robinson, T., & Leiserson, C. E. (2019). Anti-money laundering in bitcoin: Experimenting with graph convolutional networks for financial forensics. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery and data mining (KDD '19)* (p. 8).

Wu, J., Lin, D., Fu, Q., Yang, S., Chen, T., Zheng, Z., & Song, B. (2024). Toward understanding asset flows in crypto money laundering through the lenses of Ethereum heists. *IEEE Transactions on Information Forensics and Security*, *19*, 1994–2009.

Wu, J., Liu, J., Chen, W., Huang, H., Zheng, Z., & Zhang, Y. (2022a). Detecting mixing services via mining bitcoin transaction network with hybrid motifs. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, *52*(4), 2237–2249.

Wu, L., Hu, Y., Zhou, Y., Wang, H., Luo, X., Wang, Z., Zhang, F., & Ren, K. (2021). Towards understanding and demystifying bitcoin mixing services. In *Proceedings of the WWW '21: The web conference 2021* (pp. 33–44).

Wu, M., McTighe, W., Wang, K., Seres, I. A., Bax, N., Puebla, M., Mendez, M., Carrone, F., De Mattey, T., Demaestri, H. O., Nicolini, M., & Fontana, P. (2022b). Tutela: An open-source tool for assessing user-privacy on Ethereum and tornado cash. arXiv preprint, abs/2201.06811.

XBlock (2025). Xblock. https://xblock.pro. Accessed: 2025-03-15.

Xu, C., Xiong, R., Shen, X., Zhu, L., & Zhang, X. (2023). How to find a bitcoin mixer: A dual ensemble model for bitcoin mixing service detection. *IEEE Internet of Things Journal*, *10*(19), 17220–17230.

Youn, M., Chin, K., & Omote, K. (2023). Empirical analysis of cryptocurrency mixer: Tornado cash. In *2023 Congress in computer science, computer engineering, & applied computing (CSCE)* (pp. 2324–2331).

Yu, Y., Wu, J., Lin, D., & Fu, Q. (2023). Money laundering detection on Ethereum: Applying traditional approaches to new scene. In *29th IEEE International conference on parallel and distributed systems (ICPADS 2023)* (pp. 1759–1766).

Zhou, F., Chen, Y., Zhu, C., Jiang, L., Liao, X., Zhong, Z., Chen, X., Chen, Y., & Zhao, Y. (2024). Visual analysis of money laundering in cryptocurrency exchange. *IEEE Transactions on Computational Social Systems*, *11*(1), 731–745.

Zhou, J., Hu, C., Chi, J., Wu, J., Shen, M., & Xuan, Q. (2022). Behavior-aware account de-anonymization on Ethereum interaction graph. *IEEE Transactions on Information Forensics and Security*, *17*, 3433–3448.