

CryptoFlowBench: Benchmarking Multi-Hop Reasoning on Blockchains for Cryptocurrency Fund Tracing

Runang He¹, Tongya Zheng^{1,2,3*}, Jing Cao^{1,2,3}, Yucheng Ying¹, Shunyu Liu¹, Bingde Hu¹, Yang Gao¹, Zeyuan Hu⁴, Jianrong Zhang⁵, Mingli Song^{1,2,3}

¹State Key Laboratory of Blockchain and Data Security, Zhejiang University

²Hangzhou City University

³Hangzhou High-Tech Zone (Binjiang) Institute of Blockchain and Data Security

⁴China Mobile Group Zhejiang Co., Ltd

⁵China Unicom Digital Technology Co., Ltd

herunang@zju.edu.cn, doujiang_zheng@163.com, jingcao@hzcu.edu.cn, yingyu@zju.edu.cn, shunyu.liu.cs@gmail.com, tonyhu@zju.edu.cn, roygao@zju.edu.cn, huzeyuan@zj.chinamobile.com, zhangjr67@chinaunicom.cn, brooksong@zju.edu.cn

Abstract

Pseudonymous accounts and complex smart-contract ecosystems substantially hinder the tracing of illicit cryptocurrency fund flows. Existing Graph Neural Networks (GNNs) have achieved promising performance in blockchain fraud detection, yet their static node- or edge-level predictions do not naturally generalize to multi-hop fund tracing. Meanwhile, existing tracing methods mainly rely on predefined heuristics to capture multi-hop relationships, limiting their adaptive reasoning capabilities. To bridge this gap, we introduce CryptoFlowBench, a benchmark constructed from authoritative reports on cryptocurrency-related illicit incidents, with four tasks spanning local to multi-hop fund tracing. These tasks pose substantial challenges to existing learning- and heuristic-based methods, particularly in complex multi-hop scenarios. We therefore propose ChainSleuth, a graph-tool-augmented agent that iteratively reasons, invokes graph tools, and updates its tracing trajectory, enabling adaptive information acquisition and recovery from erroneous paths. Experiments show that no baseline consistently dominates all four tasks, revealing substantial gaps between local fraud detection and multi-hop fund tracing. ChainSleuth achieves notable improvements on multi-hop tasks, demonstrating the potential of agentic graph reasoning for cryptocurrency fund investigation.

Code —

<https://github.com/LuoXishuang0712/CryptoFlowBench>

Introduction

Public blockchains provide transparent and append-only transaction records without directly binding accounts to real-world identities. While supporting digital payments and decentralized finance, this infrastructure can also be exploited to steal, conceal, and launder illicit funds. According to the FBI’s 2025 Internet Crime Report, cryptocurrency was involved in approximately \$11 billion of nearly \$20 billion in reported losses (Federal Bureau of Investigation 2026). To analyze such illicit activities, on-chain transfers can be represented as large, temporal, and attributed money-flow graphs (MFGs), where nodes correspond to addresses or contracts

*Corresponding Author

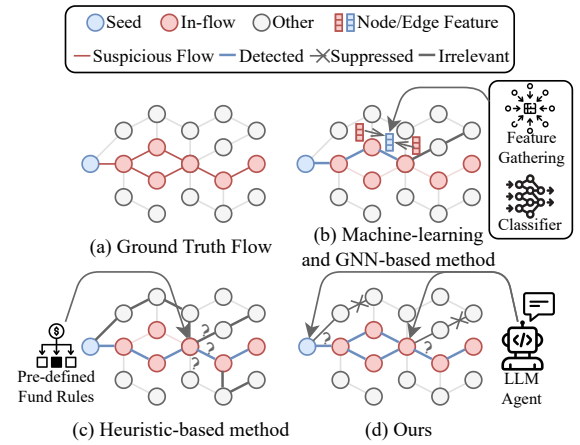


Figure 1: Comparison of fund-tracing behaviors on a case-centered transaction graph. Red edges in (a) show the verified suspicious flow. The machine-learning/GNN-based method in (b) cannot recover after selecting an irrelevant branch, shown as gray edges, while the rule-based method in (c) over-expands under a fixed propagation policy. ChainSleuth in (d) retrieves on-chain evidence, revises earlier decisions, and suppresses unsupported expansions.

and directed edges represent asset transfers (Turner, McCombie, and Uhlmann 2020; Qi et al. 2023). Within these graphs, some nodes are associated with services such as decentralized exchanges and cross-chain bridges, introducing heterogeneous transaction patterns into the tracing process (Liu et al. 2021; Bartoletti et al. 2021; Cunjak Mataković 2022; Almeida, Pinto, and Vilas 2023). As shown in Figure 1(a), practical fund tracing aims to recover the suspicious flow from limited case evidence while excluding irrelevant transfers. Existing fund-tracing methods broadly follow two paradigms. Heuristic approaches propagate funds using deterministic taint-allocation rules or graph-based relevance and flow objectives (Möser, Böhme, and Breuker 2014; Tironasakul et al. 2019; Wu et al. 2023; Lin et al.

2024). Learning-based approaches use classical classifiers or GNNs to prioritize suspicious transactions and addresses from structural and behavioral features (Weber et al. 2019; Shen et al. 2021; Huang et al. 2024). However, fixed propagation policies and independent local predictions provide limited support for revising earlier decisions, while practical tracing requires evidence-driven multi-hop reasoning and adaptive stopping.

Supporting and evaluating such practical tracing requires addressing three key challenges. **Challenge 1: Case-grounded evaluation.** Large detection benchmarks mainly provide binary address or transaction labels, whereas tracing studies often use independently constructed case graphs with inconsistent scopes, labels, and protocols. Neither setting supports unified evaluation of complete investigations across heterogeneous incidents. **Challenge 2: Error accumulation.** Machine-learning and GNN-based methods generally make independent local predictions, allowing an early false positive to propagate through a multi-hop trajectory without a mechanism for later revision, causing error accumulated along the propagation path, as illustrated in Figure 1(b). **Challenge 3: Adaptive exploration and boundary control.** Heuristic-based tracing methods rely on predefined attribution or structural policies (Möser, Böhme, and Breuker 2014; Tironsakkul et al. 2019; Wu et al. 2023; Lin et al. 2024; Wu et al. 2024b). Such policies may expand through irrelevant but highly connected services, causing common on-chain service nodes to repeatedly divert the tracing process, particularly when no reliable seed address is available, as shown in Figure 1(c).

To address **Challenge 1** and make the sequential and boundary-control requirements in **Challenges 2 and 3** measurable, we construct CRYPTOFLOWBENCH, a case-grounded benchmark that aligns heterogeneous incident reports with locally verified transactions. Each incident is represented as an evidence graph containing address roles, tracing actions, path stages, hard negatives, evidential boundaries, and source provenance. We deterministically derive four complementary tasks under a unified protocol: **transaction existence** for factual verification; **next-step action** for seeded decisions among *follow*, *inspect*, *stop*, and *ignore*; **path completion** for multi-hop tracing under accumulated errors; and **subgraph filtering** for seed localization and tracing in distractor-rich graphs. These tasks separate factual grounding, local decisions, sequential reconstruction, and seedless investigation, while exposing over-expansion and boundary violations.

To address **Challenges 2 and 3**, we propose CHAINSLEUTH, an evidence-grounded, tool-augmented LLM agent. Within a bounded ReAct loop (Yao et al. 2022), ChainSleuth maintains an evolving investigation state, selects explicit tracing actions, and retrieves transaction-level evidence through external tools. New observations can be used to revisit earlier hypotheses, suppress unsupported branches, and stop at evidential boundaries, as illustrated in Figure 1(d). Experiments show that strong local prediction does not necessarily translate into effective end-to-end tracing. Although conventional classifiers remain competitive on single-step decisions, ChainSleuth performs better on *Path* and *Subgraph* tasks, improving Path F1 by 70.0%, reducing over-

expansion by 26.6%, and improving predicted-seed F1 by 21.4% over the strongest baseline for each metric.

Our main contributions are summarized as follows:

- **CryptoFlowBench: a case-grounded benchmark for Ethereum fund-flow investigation.** We align heterogeneous incident reports with locally verified transactions to construct case-centered evidence graphs that summarizing on-chain incidents. Based on 22 incidents across five types, the benchmark derives four complementary tasks under a unified protocol, covering factual verification, seeded local decisions, multi-hop path reconstruction, and seedless suspicious-region discovery.
- **ChainSleuth: an evidence-grounded, tool-augmented LLM agent.** ChainSleuth maintains an evolving investigation state, selects explicit tracing actions, and retrieves auditable on-chain evidence through external tools. It can revisit earlier hypotheses as new evidence is acquired, thereby correcting accumulated errors, suppressing unsupported branches, and controlling tracing boundaries.
- **A comprehensive, subtask-specific evaluation.** We compare ChainSleuth with task-appropriate heuristic-based, machine-learning, GNN, and LLM baselines under matched case splits and tracing budgets. On the core tracing tasks, ChainSleuth improves Path F1 by 70.0%, reduces over-expansion by 26.6%, and improves predicted-seed F1 by 21.4% over the strongest specialized baseline for each metric.

Related Work

Cryptocurrency Illicit-Activity Detection

Existing cryptocurrency illicit-activity detection methods can be broadly grouped into several paradigms (Fouzy et al. 2026). Heuristic-based methods include rule-based approaches using predefined attribution policies and graph-based approaches using transaction topology for candidate prioritization; although interpretable, they are limited by fixed assumptions or scalability on large, densely connected graphs. Machine-learning methods, such as logistic regression, random forests, and XGBoost (Weber et al. 2019; El-mougy and Liu 2023), are generally efficient but depend heavily on handcrafted features and capture relational dependencies only indirectly. GNN-based methods, including GCN (Weber et al. 2019), I²BGNN (Shen et al. 2021), and PEA-GNN (Huang et al. 2024), model complex account interactions more directly, but their computational cost and limited explainability hinder evidence-sensitive forensic use. General graph anomaly detection methods, such as H₂GCN (Zhu et al. 2020), PMP (Zhuo et al. 2024), ConsisGAD (Chen et al. 2024), UNPrompt (Niu et al. 2024), ARC (Liu et al. 2024), and AnomalyGFM (Qiao et al. 2025), are not blockchain-specific but offer mechanisms for handling heterophily, label scarcity, and class imbalance. Their effectiveness has also been demonstrated in blockchain anomaly detection (He et al. 2026), although they remain focused on node-level detection rather than evidence-grounded, multi-hop fund tracing.

Cryptocurrency Fund-Flow Tracing

Early cryptocurrency tracing relied on deterministic taint-propagation rules, including Poison, Haircut, FIFO, LIFO, and TIHO, to associate downstream Bitcoin outputs with known illicit funds (Möser, Böhme, and Breuker 2014; Tironasakul et al. 2019). Although transparent and reproducible, these methods depend on fixed value-allocation assumptions and can produce substantially different tracing results for the same source transaction. Learning- and graph-based approaches subsequently introduced more flexible prioritization. The Elliptic study applied classical classifiers and graph convolutional network (GCN) to illicit Bitcoin transaction detection (Weber et al. 2019), while TRacer modeled tracing in account-based blockchains as relevance-guided search over directed, temporal, weighted, and multi-relational graphs (Wu et al. 2023). DenseFlow and DenseFlow+ further combined suspicious-group discovery with Ethereum asset-flow reconstruction using dense-subgraph and flow-based objectives (Lin et al. 2024; Wu et al. 2024b). These methods advance transaction classification, account prioritization, and path reconstruction, but do not explicitly model evidence-grounded investigative decisions or revision after new observations.

LLM Agents for Blockchain Security and Analysis

LLM agents extend static language-model generation by interleaving reasoning, tool use, and environmental observations, enabling adaptive multi-step decision making (Yao et al. 2022; Paranjape et al. 2023). Recent studies have applied this agentic paradigm to external environments such as web navigation (Zhou et al. 2024; Lu et al. 2025), graphical user interfaces (Xie et al. 2024; Zhai et al. 2025), and structured graphs (Jin et al. 2024; Ying et al. 2026), where agents iteratively gather evidence and refine their actions. Such capabilities are well suited to on-chain analysis, which requires querying large transaction graphs and verifying intermediate decisions with external evidence. LLM agents have also been explored for smart-contract auditing, vulnerability detection, program analysis, and automated repair (He et al. 2024; Wei et al. 2024). However, existing studies largely focus on contract code rather than evidence-grounded, multi-step fund-flow tracing.

Preliminaries

Blockchain Ledger An account-based blockchain is an append-only block sequence $\mathcal{L} = (B_0, \dots, B_L)$. Within a block window $W = [b_{\min}, b_{\max}]$, each indexed transfer is represented as

$$x_i = (h_i, b_i, t_i, s_i, d_i, a_i, m_i, \kappa_i) \in \mathcal{X}_W,$$

where h_i , b_i , and t_i denote the transaction hash, block number, and timestamp; s_i and d_i denote the source and destination addresses; and a_i , m_i , and κ_i denote the asset identifier, amount, and transfer type. The indexed transfer set is

$$\mathcal{X}_W = \{x_i \mid b_{\min} \leq b_i \leq b_{\max}\}.$$

Exact transaction-existence claims are made only with respect to the indexed chain, block window, and supported transfer types.

Case	Primary source	Type
01 Ronin	U.S. Treasury	Credential/ UI compromise
02 Harmony	Harmony	
03 CoinEx	CoinEx	
04 HTX HECO Bridge	HTX	
05 Stake.com	Stake.com	
06 WazirX	WazirX	
07 Radiant Capital	Radiant Capital	
08 PlayDapp	PlayDapp	
09 Bybit	Bybit	
10 Wormhole	CertiK	Protocol exploit
11 Nomad Bridge	Nomad	
12 BNB Bridge	BNB Chain	
13 Euler Finance	Euler Finance	
14 Curve Vyper Pools	Vyper	
15 KyberSwap Elastic	KyberSwap	
16 Cetus Protocol	Sui Foundation	
17 Prisma Finance	Prisma Finance	
18 Penpie	Penpie	
19 Dough Finance	Dough Finance	
20 UwU Lend	SlowMist	Oracle manipulation
21 Munchables	Halborn	Backdoor
22 Atomic Wallet	Atomic Wallet	Wallet compromise

Table 1: Incident types and primary sources of the benchmark cases.

Transaction-Graph Construction Raw transfers are organized into a directed address-relation graph whose nodes are addresses and whose relation edges retain the underlying transfer records. For each ordered address pair (u, v) , define

$$\mathcal{X}_{uv} = \{x_i \in \mathcal{X}_W \mid s_i = u, d_i = v\}.$$

When $\mathcal{X}_{uv} \neq \emptyset$, $e_{uv} = \text{Agg}(\mathcal{X}_{uv})$ preserves transaction identifiers and transfer metadata, with amounts aggregated only among records sharing the same asset identifier and transfer type. The resulting graph is

$$\Phi_W(\mathcal{L}) = \mathcal{G}_W = (V_W, E_W),$$

where

$$V_W = \{s_i, d_i \mid x_i \in \mathcal{X}_W\}, \quad E_W = \{e_{uv} \mid \mathcal{X}_{uv} \neq \emptyset\}.$$

Fund-Flow Tracing Given $\mathcal{G}_W$, public incident evidence x_c^{pub} , an optional seed set S_0 , and exploration budget $\mathbf{B}$, fund-flow tracing returns

$$\text{Trace}(\mathcal{G}_W, x_c^{\text{pub}}, S_0; \mathbf{B}) = (\hat{S}, \hat{E}, \hat{\mathcal{P}}),$$

where $\hat{S}$ is the supplied or inferred seed set, $\hat{E}$ is the selected tracing subgraph, and $\hat{\mathcal{P}}$ contains directed paths whose edges are contained in $\hat{E}$. For a hidden verified reference graph $\mathcal{G}_c^* = (V_c^*, E_c^*)$, the objective is to recover relevant edges under $\mathbf{B}$ while limiting irrelevant expansion $\hat{E} \setminus E_c^*$. The reference graph represents the verified evidence for case c , rather than an exhaustive reconstruction of all illicit activity.

CryptoFlowBench

We propose CryptoFlowBench, a benchmark with four complementary fund-tracing tasks: transaction existence, next-step action, path completion, and subgraph filtering. They evaluate factual verification, local tracing decisions, sequential reasoning under accumulated errors, and seedless subgraph-level tracing, respectively.

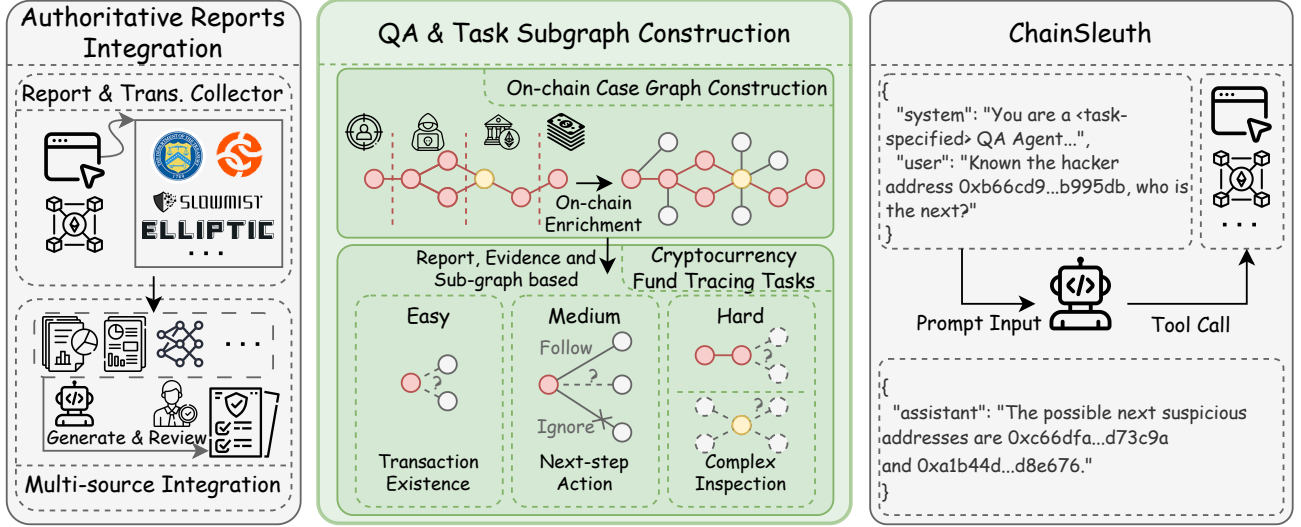


Figure 2: Overview of the benchmark construction pipeline. Public incident reports are consolidated into structured case summaries, aligned with indexed on-chain transactions, and converted into grounded question-answering tasks with task-specific verification. Where “Trans.” indicates “Transaction”.

Figure 2 summarizes the benchmark construction pipeline. Let $\mathcal{C} = \{c_i\}_{i=1}^N$ denote publicly documented cryptocurrency incidents. We use LLM-driven agents $\mathcal{A}_{\text{col}}$ and $\mathcal{A}_{\text{sum}}$ for report collection and structured summarization, while graph alignment, task generation, and verification are deterministic. For each incident c , the agents collect a document set $\mathcal{D}_c$ and normalize its claims into a structured summary S_c . Subsequent stages construct a report-evidence graph $\mathcal{H}_c$, align it with a bounded on-chain graph $\mathcal{G}_c^{(k)}$, and generate task instances $\mathcal{Q}_c$. According to this pipeline, we collect 22 incidents across five incident types, as shown in Table 1.

Authoritative Reports Integration

Existing cryptocurrency datasets mainly support binary detection and provide limited case context that can be grounded in on-chain evidence. This mismatch hinders the evaluation of practical tracing systems, especially LLM agents that require incident context and verifiable intermediate evidence. We therefore design an authoritative report collection and integration pipeline.

We first collect several on-chain incidents and utilize a web-enabled LLM agent to collect online reports. For each incident c , the collection agent produces

$$\mathcal{D}_c = \mathcal{A}_{\text{col}}(c; \theta_{\text{col}}, \mathcal{T}_{\text{col}}),$$

from public sources, including incident disclosures, technical postmortems, security analyses, and blockchain explorer pages. To further reduce the randomness of the agent, we repeat the collection process for multiple passes, which is noted as $\mathcal{D}_c^p$, where p indicates the index of pass.

Then, to verify and gather reports, we utilize a summarization agent to review all reports

$$S_c = \mathcal{A}_{\text{sum}}((\mathcal{D}_c^1, \dots, \mathcal{D}_c^p); \theta_{\text{sum}}, \mathcal{T}_{\text{sum}}).$$

The summarization agent independently verifies and distills the reports, and all sources are classified according to provenance priority.

On-chain Case Graph Construction

To align reports with on-chain evidence, we construct a case graph that maps report relations into on-chain transactions, ensuring all the evidence is case-grounded.

We first deterministically transform the structured summary S_c into a case-evidence specification

$$\mathcal{H}_c = (V_c^H, E_c^H, \rho_c, \pi_c),$$

where V_c^H contains reported addresses, contracts, protocols, and service entities; E_c^H contains reported transfers and stage-level relations; ρ_c records entity roles and tracing stages; and π_c preserves source provenance, temporal constraints, transaction identifiers, and verification status. The evidence specification provides candidate anchors and alignment constraints, but does not itself assert that reported relations exist on chain.

Then, we ground and extend the report-derived evidence using the local blockchain index, which is constructed by fetching transactions from a blockchain client and indexing according to the addresses. Given the max expand hop k , blockchain index $\mathcal{I}$, case evidence $\mathcal{H}_c$, case block window W_c , and resource bounds $\mathbf{B}$, we construct

$$\mathcal{G}_c^{(k)} = \text{Expand}_k(\mathcal{I}, \mathcal{H}_c; W_c, \mathbf{B}).$$

During the expanding process, we first derive candidate anchors from explicitly reported entities, then retrieve their bounded k -hop neighborhoods from the local index. Absence from the bounded expansion is not treated as evidence of non-existence; exact negative instances are generated only from complete index queries. The resulting representation

therefore combines report-derived semantics from $\mathcal{H}_c$ with locally verified transaction evidence from $\mathcal{G}_c^{(k)}$ through explicit alignment and provenance links.

Cryptocurrency Fund Tracing Tasks

Existing methods are typically evaluated on isolated objectives, such as node classification or fund-flow reconstruction, and therefore do not expose the full investigation process. To better match practical requirements, CryptoFlowBench defines four complementary tasks: transaction existence, next-step action, path completion, and subgraph filtering.

First, given the aligned case evidence $\mathcal{H}_c$ and bounded on-chain graph $\mathcal{G}_c^{(k)}$, we jointly construct task instances and their oracle annotations through a deterministic, graph-first procedure. Each candidate relation e is assigned an evidence label $y_e \in \mathcal{Y}$ and an investigation action

$$u_e \in \mathcal{U} = \{\text{follow}, \text{inspect}, \text{stop}, \text{ignore}\},$$

where `follow` denotes continuation along the reference fund flow, `inspect` marks relevant but insufficient evidence, `stop` identifies an evidential or operational boundary, and `ignore` denotes an irrelevant relation. This action space extends binary edge classification into an investigative decision interface, where `stop` and `ignore` explicitly represent the evidential boundary of a case.

Then we generate four types of tasks for each labeled case graph $\mathcal{G}_c^{(k)}$: **transaction existence** (*Trans*), **next-step action** (*Action*), **path completion** (*Path*), and **subgraph filtering** (*Subgraph*). *Trans* is a standalone diagnostic that asks whether an exact transaction relation exists under explicit block-window and transfer-type constraints. *Action* evaluates one seeded local decision over the outgoing relations of a known current node. *Path* composes the same local decision interface over multiple hops, exposing error accumulation under teacher-forced and free-rollout settings. *Subgraph* removes the trusted starting node and evaluates seed localization followed by tracing within a distractor-rich subgraph.

Let

$$\mathcal{R} = \{\text{Trans}, \text{Action}, \text{Path}, \text{Subgraph}\}.$$

The task set for case c is generated as

$$\mathcal{Q}_c = \bigcup_{r \in \mathcal{R}} \text{Generate}_r(\mathcal{H}_c, \mathcal{G}_c^{(k)}; \mathcal{Y}, \mathcal{U}, \mathbf{B}_r),$$

where $\mathbf{B}_r$ specifies task-dependent limits on depth, candidate size, node expansion, and evidence access. Task *Action*, *Path*, and *Subgraph* deliberately share the same edge-decision semantics, while *Action* fixes the seed and evaluates one local state, *Path* fixes the seed and evaluates a sequential rollout, and *Subgraph* additionally requires the seed to be inferred.

Every task $q \in \mathcal{Q}_c$ is separated into a public record x_q^{pub} and a disjoint oracle record y_q^{oracle} . The former contains only the task question, candidate set, permitted graph context, tool declarations, budget, and response schema. Answers, gold actions, case-path labels, and evaluation annotations remain oracle-only and will be removed from prompts, retrieval

stores, and tool outputs. For a fixed case snapshot and configuration, candidate construction, ordering, oracle generation, and public-context sanitization are deterministic.

Appendix.A presents representative benchmark cases to illustrate how incident reports are aligned with on-chain evidence and converted into case-centered graphs with tracing labels, evidential boundaries, and task instances.

ChainSleuth

Existing tracing methods rely on fixed propagation rules or independently scored local decisions, and therefore provide limited support for revising earlier hypotheses during multi-hop investigation. We propose ChainSleuth, a ReAct-based, retrieval-augmented LLM agent for evidence-grounded on-chain fund tracing (Yao et al. 2022). Through a local blockchain index, blockchain explorers, and optional pre-trained detectors, ChainSleuth adaptively retrieves evidence and updates its tracing trajectory. The tool design is detailed in Appendix.B.

For each public task state x_q^{pub} , ChainSleuth receives the task description, candidate set, response schema, and declarations of the allowed and required tools.

During execution, ChainSleuth follows a bounded interaction loop. At iteration j , given the public task input and interaction history H_j , the model either issues an allowed tool call

$$a_j = \pi_\theta(H_j, x_q^{\text{pub}}),$$

receives a sanitized observation

$$o_j = \mathcal{T}(a_j),$$

and updates

$$H_{j+1} = H_j \cup \{a_j, o_j\},$$

or terminates with a structured answer. Execution stops when the answer is produced, the evidence coverage is reached, no further progress is made, or the tool budget is exhausted.

For tasks *Action*, *Path*, and *Subgraph*, ChainSleuth retrieves transaction-level evidence for public candidate relations and then predicts an action distribution

$$\pi_\theta(u \mid s_t, e), \quad u \in \mathcal{U},$$

for each candidate relation e at state s_t . Only relations assigned `follow` with probability above the declared threshold may be expanded. Task *Action* performs this decision once, *Path* repeatedly applies it along a bounded trajectory, and *Subgraph* first estimates a seed distribution $\rho_\theta(v \mid x_q^{\text{pub}})$ before applying the same edge policy.

After evidence collection terminates, a final synthesis step produces the required JSON response with tools disabled.

Experiments

We evaluate CryptoFlowBench at two levels. Task *Trans* and *Action* assess fundamental investigation capabilities, including exact tool invocation, evidence grounding, and local action selection. Because these diagnostic tasks do not capture full multi-hop tracing, their results are reported in Appendix.C. Task *Path* and *Subgraph* constitute the core

Table 2: Task distribution by incident.

Case	Trans	Action	Path	Subgraph	Total
01 Ronin	658	11	7	13	689
02 Harmony	1,720	4	1	25	1,750
03 CoinEx	1,216	4	0	21	1,241
04 HTX HECO Bridge	220	14	5	13	252
05 Stake.com	148	10	2	13	173
06 WazirX	706	5	1	25	737
07 Radiant Capital	1,228	7	4	13	1,252
08 PlayDapp	1,688	5	1	25	1,719
09 Bybit	19,584	54	4	25	19,667
10 Wormhole	202	1	0	13	216
11 Nomad Bridge	522	5	1	17	545
12 BNB Bridge	1,594	1	0	13	1,608
13 Euler Finance	1,504	13	8	25	1,550
14 Curve Vyper Pools	440	8	3	15	466
15 KyberSwap Elastic	1,844	3	0	25	1,872
16 Cetus Protocol	662	3	0	13	678
17 Prisma Finance	466	6	3	25	500
18 Penpie	328	5	3	13	349
19 Dough Finance	1,610	5	2	25	1,642
20 UwU Lend	1,620	3	0	13	1,636
21 Munchables	238	7	2	13	260
22 Atomic Wallet	350	14	4	14	382
Total	38,548	188	51	397	39,184
Share (%)	98.4	0.5	0.1	1.0	100.0

tracing benchmark, evaluating sequential investigation under cumulative error and seedless suspicious-region discovery, respectively. For these tasks, we compare ChainSleuth with task-appropriate heuristic- and learning-based baselines under matched candidate nodes, case splits, and inference budgets.

Experimental Settings

Dataset and Splits We construct the benchmark from 22 publicly documented Ethereum incidents using the report collection, on-chain alignment, and deterministic task generation pipeline described above. Table 2 reports their distribution across incidents. To prevent case-specific information leakage, all learning-based methods are trained, validated, and tested using case-disjoint splits.

Agent Models We evaluate ChainSleuth using QWEN3.6-27B (Qwen Team 2026), DEEPSEEK-V4-FLASH (DS-V4-Flash), and DEEPSEEK-V4-PRO (DS-V4-Pro) (DeepSeek-AI 2026) as LLM.

Baselines We compare ChainSleuth with three groups of task-appropriate baselines. We group heuristic baselines into predefined rule-based and graph-based tracing methods.

Rule-based tracing. We adapt Poison, Haircut, and FIFO (Möser, Böhme, and Breuker 2014), together with LIFO and TIHO (Tironsakkul et al. 2019), to the Ethereum account-based setting. Taint is propagated separately by asset, and all methods use the same initial taint budget, candidate universe, and rollout limits.

Graph-based tracing. We include TRacer (Wu et al. 2023) and adaptations inspired by DenseFlow and DenseFlow+ (Lin et al. 2024; Wu et al. 2024b). Because these methods are not originally designed for the present action interface, we map their graph scores to candidate-edge rankings and follow decisions without access to target-case labels.

Learning-based classifiers. We train logistic regression (LR), random forest (RF), multilayer perceptron (MLP),

Table 3: *Path* results on 51 tasks from 16 cases. * indicates ChainSleuth and + indicates detector tool enabled.

Method	Comp. $\uparrow$	Path F1 $\uparrow$	TF Acc. $\uparrow$	Over-exp. $\downarrow$
Poison	0.330	0.345	0.396	0.644
Haircut	0.330	0.311	0.396	0.674
FIFO	0.422	0.340	0.560	0.628
LIFO	0.301	0.328	0.476	0.651
TIHO	0.313	0.307	0.604	0.646
TRacer	0.087	0.186	0.643	0.158
DenseFlow+	0.296	0.315	0.517	0.332
LR	0.426	0.432	0.718	0.590
RF	0.374	0.379	0.813	0.603
MLP	0.253	0.212	0.878	0.270
GCN	0.336	0.300	0.774	0.605
I ² BGNN	0.305	0.279	0.623	0.450
PEAE-GNN	0.290	0.267	0.572	0.409
TokenScout	0.307	0.286	0.659	0.414
XGB	0.276	0.279	0.854	0.325
DS-V4-Flash *	0.092	0.202	0.244	0.097
DS-V4-Pro *	0.569	0.541	0.570	0.185
Qwen3.6-27B *	<u>0.665</u>	0.734	0.824	<u>0.116</u>
Qwen3.6-27B *+	0.760	<u>0.628</u>	0.603	0.221

GCN (Weber et al. 2019), and XGBoost (Elmougy and Liu 2023) using public candidate-edge features and case-disjoint training folds. We also utilize several blockchain oriented GNN-based methods, including I²BGNN (Shen et al. 2021), PEAE-GNN (Huang et al. 2024), and TokenScout (Wu et al. 2024a). All GNN-based methods are adapted to edge-level prediction because their original formulations classify transaction nodes rather than candidate address relations.

A frozen XGBoost node classifier is shared across all edge methods for predicted-seed evaluation in the *Subgraph* task, the selection of node classifier will be discussed below.

Evaluation Metrics All results use deterministic primary scorers and are macro-averaged across incidents. We report 95% confidence intervals in Appendix D. For the *Path* task, complete-path rate (Comp.) and Path F1 evaluate trajectory reconstruction, teacher-forced accuracy (TF Acc.) isolates local decision quality without accumulated rollout error, and over-expansion (Over-exp.) measures continuation beyond the verified boundary. For the *Subgraph* task, Path F1 using predicted seeds (Pred F1) evaluates the complete pipeline of seed identification and suspicious-region expansion, while Path F1 using oracle seeds (Oracle F1) removes seed identification errors and isolates propagation quality; path precision (Path P) and recall (Path R) further characterize the trade-off between noise suppression and suspicious-region coverage. For ChainSleuth, structured-output validity and tool-protocol pass rate are also reported to distinguish tracing performance from execution and protocol failures.

Comparison Experiments

Table 3 and 4 show the result of baselines and ChainSleuth on the *Path* and *Subgraph* tasks, respectively.

ChainSleuth versus Specialized Tracing Methods

ChainSleuth shows its clearest advantage on the practical tracing objectives that require multi-step coordination and control of irrelevant expansion. On the *Path*, Qwen3.6-27B

Table 4: *Subgraph* results on the valid-seed, positive-path subset of 22 tasks from 22 cases. * indicates ChainSleuth and + indicates detector tool enabled.

Method	Pred F1 ↑	Oracle F1 ↑	Path P ↑	Path R ↑
Poison	0.199	0.167	0.295	0.180
Haircut	0.199	0.167	0.295	0.180
FIFO	0.219	0.228	0.318	0.190
LIFO	0.217	0.136	0.379	0.164
TIHO	0.237	0.228	0.386	0.202
TRacer	0.109	0.137	0.220	0.075
DenseFlow+	0.087	0.126	0.167	0.063
LR	0.303	0.565	0.473	0.283
RF	0.309	0.588	0.553	0.252
MLP	0.196	0.386	0.394	0.145
GCN	0.199	0.423	0.295	0.180
I ² BGNN	0.176	0.350	0.265	0.161
PEAE-GNN	0.166	0.383	0.250	0.153
TokenScout	0.187	0.400	0.280	0.170
XGB	0.218	0.391	0.485	0.148
DS-V4-Flash *	0.094	0.094	0.081	0.127
DS-V4-Pro *	0.213	0.204	0.166	0.381
Qwen3.6-27B *	0.375	0.306	0.314	0.568
Qwen3.6-27B *+	0.182	0.271	0.223	0.183

Table 5: Execution reliability of ChainSleuth on path completion. All the methods are ChainSleuth and + indicates detector tool enabled.

Method	Structure-valid rate ↑	Tool-protocol pass rate ↑
DS-V4-Flash	0.318	0.099
DS-V4-Pro	0.780	0.292
Qwen3.6-27B +	0.802	0.766
Qwen3.6-27B	1.000	0.979

achieves a complete-path rate of 0.665 and a Path F1 of 0.734, substantially outperforming the strongest conventional baselines. It also reduces over-expansion to 0.116, compared with 0.158 for the strongest baseline, indicating that ChainSleuth can revise local hypotheses and terminate irrelevant branches before their errors accumulate along the trajectory. A similar pattern appears on the *Subgraph*, where Qwen3.6-27B obtains the highest predicted-seed F1 of 0.375 and the highest path recall of 0.568. RF performs better when an oracle seed is supplied, achieving an oracle-seed F1 of 0.588 and a path precision of 0.553, but its predicted-seed F1 decreases to 0.309. These results indicate that specialized classifiers remain strong when the starting point is known, whereas ChainSleuth is more effective in the end-to-end setting that jointly requires seed discovery, evidence acquisition, and subsequent path expansion.

Effect of LLM Backends The performance gap between Qwen3.6-27B and the DeepSeek-V4-Flash/Pro backends is closely associated with execution reliability rather than being attributable solely to their underlying reasoning capability. As shown in Table 5, on the *Path* task, Qwen3.6-27B achieves a structured-valid rate of 1.000 and a tool-protocol pass rate of 0.979, whereas DeepSeek-V4-Flash reaches only 0.318 and 0.099, respectively. DeepSeek-V4-Pro improves structured validity to 0.780 and tool-protocol compliance to 0.292, which is accompanied by substantially better tracing performance. The same ordering is observed on *Subgraph*,

Table 6: Case-disjoint comparison of candidate seed detectors under different illicit-node ratios in the training set.

Train rate	Method	AP ↑	Hit@1 ↑	Hit@5 ↑	MRR ↑
1%	LR	0.651	0.546	0.955	0.746
	RF	0.881	0.955	1.000	0.970
	XGB	0.883	0.955	1.000	0.970
5%	LR	0.659	0.546	0.955	0.735
	RF	0.862	0.909	1.000	0.947
	XGB	0.877	0.864	1.000	0.921
10%	LR	0.688	0.500	1.000	0.750
	RF	0.827	0.864	1.000	0.921
	XGB	0.829	0.773	1.000	0.886

where the predicted-seed F1 rises from 0.094 for DeepSeek-V4-Flash to 0.213 for DeepSeek-V4-Pro, but remains below the 0.375 achieved by Qwen3.6-27B. The consistent relationship between protocol reliability and downstream tracing performance suggests that execution reliability is an important contributor to the observed backend gap.

Detector as Tool We also evaluate the performance of ChainSleuth with detector as tool, as show in Table 3 and Table 4. In detail, we utilize LR, RF, and XGBoost as tools, as those methods can perform a decent result in the *Action*, *Path*, and *Subgraph* tasks, comparing in baselines. Adding the detector tool increases *Path* completeness from 0.665 to 0.760, but decreases Path F1 from 0.734 to 0.628 and increases over-expansion from 0.116 to 0.221. On the *Subgraph* task, it degrades all reported metrics, suggesting that detector scores may bias the agent toward locally plausible but globally inconsistent expansions.

Seed-Detector Sensitivity

To ensure a fair comparison in *Subgraph*, all baselines share the same case-disjoint seed detector under identical public features and target-case splits. As shown in Table 6, RF and XGBoost achieve comparable ranking performance and both outperform LR, indicating that the selected detector is not an unusually favorable choice. We therefore use XGBoost for predicted-seed evaluation and report oracle-seed results separately to isolate propagation performance.

Conclusion

We introduce a case-grounded benchmark for Ethereum fund-flow tracing that unifies transaction verification, local decisions, multi-step path completion, and seedless subgraph filtering under strict public-oracle separation. We further develop ChainSleuth that retrieves auditable on-chain evidence and makes structured tracing decisions without requiring the full transaction graph in context. Experiments show that conventional methods remain competitive on single-step decisions, whereas ChainSleuth performs better on sequential and seed-uncertain tracing. These results highlight the potential of evidence-grounded agents for practical and reproducible cryptocurrency investigations.

Social Impact. Cryptocurrency-enabled crime causes substantial financial harm, yet practical fund tracing remains difficult due to pseudonymous accounts and complex

multi-hop transactions. We provide a new AI perspective by modeling tracing as a case-grounded, sequential investigation problem rather than static illicit-activity detection. CryptoFlowBench supports reproducible evaluation across factual verification, local decisions, path reconstruction, and seedless discovery, while ChainSleuth illustrates how tool-augmented agents can acquire evidence, revise earlier decisions, and limit irrelevant expansion. This work provides a clear foundation for follow-up research and future decision-support tools for cryptocurrency investigators.

References

- Almeida, H.; Pinto, P.; and Vilas, A. F. 2023. A review on cryptocurrency transaction methods for money laundering. *arXiv preprint arXiv:2311.17203*.
- Bartoletti, M.; Lande, S.; Loddo, A.; Pompianu, L.; and Serusi, S. 2021. Cryptocurrency scams: Analysis and perspectives. *Ieee Access*, 9: 148353–148373.
- Chen, N.; Liu, Z.; Hooi, B.; He, B.; Fathony, R.; Hu, J.; and Chen, J. 2024. Consistency training with learnable data augmentation for graph anomaly detection with limited supervision. In *The twelfth international conference on learning representations*.
- Cunjak Mataković, I. 2022. Crypto-assets illicit activities: theoretical approach with empirical review. *International e-journal of criminal sciences*, 17.
- DeepSeek-AI. 2026. DeepSeek-V4: Towards Highly Efficient Million-Token Context Intelligence.
- Elmougy, Y.; and Liu, L. 2023. Demystifying fraudulent transactions and illicit nodes in the bitcoin network for financial forensics. In *Proceedings of the 29th ACM SIGKDD conference on knowledge discovery and data mining*, 3979–3990.
- Federal Bureau of Investigation. 2026. 2025 Internet Crime Report. Technical report, Internet Crime Complaint Center. Accessed: 2026-07-22.
- Fouzy, N.; Elhadeif, M.; Zarouk, H.; Jawish, A.; and Modugu, K. 2026. Cryptocurrency Money Laundering Techniques and AI-Driven Detection Approaches. In *2026 International Conference on Integrated Intelligence and Cognitive Engineering (ICIICE)*, 1–6. IEEE.
- He, R.; Zheng, T.; Peng, H.; Wan, Y.; Hu, B.; Chen, J.; Jin, C.; Song, M.; and Wang, C. 2026. Temporal Motif-aware Graph Test-time Adaptation for OOD Blockchain Anomaly Detection. *arXiv preprint arXiv:2605.29526*.
- He, Z.; Li, Z.; Yang, S.; Ye, H.; Qiao, A.; Zhang, X.; Luo, X.; and Chen, T. 2024. Large language models for blockchain security: A systematic literature review. *arXiv preprint arXiv:2403.14280*.
- Huang, H.; Zhang, X.; Wang, J.; Gao, C.; Li, X.; Zhu, R.; and Ma, Q. 2024. PEA-E-GNN: Phishing detection on Ethereum via augmentation ego-graph based on graph neural network. *IEEE Transactions on Computational Social Systems*, 11(3): 4326–4339.
- Jin, B.; Xie, C.; Zhang, J.; Roy, K. K.; Zhang, Y.; Li, Z.; Li, R.; Tang, X.; Wang, S.; Meng, Y.; et al. 2024. Graph chain-of-thought: Augmenting large language models by reasoning on graphs. In *Findings of the Association for Computational Linguistics: ACL 2024*, 163–184.
- Lin, D.; Wu, J.; Yu, Y.; Fu, Q.; Zheng, Z.; and Yang, C. 2024. Denseflow: Spotting cryptocurrency money laundering in ethereum transaction graphs. In *Proceedings of the ACM Web Conference 2024*, 4429–4438.
- Liu, X. F.; Jiang, X.-J.; Liu, S.-H.; and Tse, C. K. 2021. Knowledge discovery in cryptocurrency transactions: A survey. *Ieee access*, 9: 37229–37254.
- Liu, Y.; Li, S.; Zheng, Y.; Chen, Q.; Zhang, C.; and Pan, S. 2024. Arc: A generalist graph anomaly detector with in-context learning. *Advances in Neural Information Processing Systems*, 37: 50772–50804.
- Lu, R.; Hou, Z.; Wang, Z.; et al. 2025. Deepdive: Advancing deep search agents with knowledge graphs and multi-turn rl. *arXiv preprint arXiv:2509.10446*.
- Möser, M.; Böhme, R.; and Breuker, D. 2014. Towards risk scoring of Bitcoin transactions. In *International conference on financial cryptography and data security*, 16–32. Springer.
- Niu, C.; Qiao, H.; Chen, C.; Chen, L.; and Pang, G. 2024. Zero-shot generalist graph anomaly detection with unified neighborhood prompts. *arXiv preprint arXiv:2410.14886*.
- Paranjape, B.; Lundberg, S.; Singh, S.; Hajishirzi, H.; Zettlemoyer, L.; and Ribeiro, M. T. 2023. Art: Automatic multi-step reasoning and tool-use for large language models. *arXiv preprint arXiv:2303.09014*.
- Qi, Y.; Wu, J.; Xu, H.; and Guizani, M. 2023. Blockchain data mining with graph learning: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 46(2): 729–748.
- Qiao, H.; Niu, C.; Chen, L.; and Pang, G. 2025. Anomalygfm: Graph foundation model for zero/few-shot anomaly detection. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*, 2326–2337.
- Qwen Team. 2026. Qwen3.6-27B: Flagship-Level Coding in a 27B Dense Model.
- Shen, J.; Zhou, J.; Xie, Y.; Yu, S.; and Xuan, Q. 2021. Identity inference on blockchain using graph neural network. In *International conference on blockchain and trustworthy systems*, 3–17. Springer.
- Tironsakkul, T.; Maarek, M.; Eross, A.; and Just, M. 2019. Probing the mystery of cryptocurrency theft: an investigation into methods for taint analysis. *arXiv preprint arXiv:1906.05754*.
- Turner, A. B.; McCombie, S.; and Uhlmann, A. J. 2020. Analysis techniques for illicit bitcoin transactions. *Frontiers in Computer Science*, 2: 600596.
- Weber, M.; Domeniconi, G.; Chen, J.; Weidele, D. K. I.; Bellei, C.; Robinson, T.; and Leiserson, C. E. 2019. Anti-money laundering in bitcoin: Experimenting with graph convolutional networks for financial forensics. *arXiv preprint arXiv:1908.02591*.
- Wei, Z.; Sun, J.; Zhang, Z.; Zhang, X.; Li, M.; and Hou, Z. 2024. Llm-smartaudit: Advanced smart contract vulnerability detection. *arXiv preprint arXiv:2410.09381*.

Wu, C.; Chen, J.; Zhao, Z.; He, K.; Xu, G.; Wu, Y.; Wang, H.; Li, H.; Liu, Y.; and Xiang, Y. 2024a. Tokenscout: Early detection of ethereum scam tokens via temporal graph learning. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, 956–970.

Wu, J.; Yu, Y.; Lin, D.; Chen, Z.; and Jiang, Z. 2024b. From Shadows to Light: Uncovering Money Laundering in Ethereum Transaction Graphs. *Available at SSRN 5076472*.

Wu, Z.; Liu, J.; Wu, J.; Zheng, Z.; and Chen, T. 2023. TRacer: Scalable graph-based transaction tracing for account-based blockchain trading systems. *IEEE Transactions on Information Forensics and Security*, 18: 2609–2621.

Xie, T.; Zhang, D.; Chen, J.; Li, X.; Zhao, S.; Cao, R.; Hua, T. J.; Cheng, Z.; Shin, D.; Lei, F.; et al. 2024. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments. *Advances in Neural Information Processing Systems*, 37: 52040–52094.

Yao, S.; Zhao, J.; Yu, D.; Du, N.; Shafran, I.; Narasimhan, K.; and Cao, Y. 2022. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*.

Ying, Y.; Jiang, W.; Zheng, T.; Wang, Y.; Liu, S.; Chen, K.; and Song, M. 2026. GraphScout: Empowering Large Language Models with Intrinsic Exploration Ability for Agentic Graph Reasoning. *arXiv preprint arXiv:2603.01410*.

Zhai, Y.; Tao, S.; Chen, C.; Zou, A.; Chen, Z.; Fu, Q.; Mai, S.; Yu, L.; Deng, J.; Cao, Z.; et al. 2025. Agentevolver: Towards efficient self-evolving agent system. *arXiv preprint arXiv:2511.10395*.

Zhou, S.; Xu, F. F.; Zhu, H.; Zhou, X.; Lo, R.; Sridhar, A.; Cheng, X.; Ou, T.; Bisk, Y.; Fried, D.; et al. 2024. Webarena: A realistic web environment for building autonomous agents. In *International Conference on Learning Representations*, volume 2024, 15585–15606.

Zhu, J.; Yan, Y.; Zhao, L.; Heimann, M.; Akoglu, L.; and Koutra, D. 2020. Beyond homophily in graph neural networks: Current limitations and effective designs. *Advances in neural information processing systems*, 33: 7793–7804.

Zhuo, W.; Liu, Z.; Hooi, B.; He, B.; Tan, G.; Fathony, R.; and Chen, J. 2024. Partitioning message passing for graph fraud detection. *arXiv preprint arXiv:2412.00020*.