BGENT: A MULTI-TASK BLOCKCHAIN INVESTIGATION AGENT WITH EXECUTABLE CLAIM CHECKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Blockchain investigations connect transaction execution, account behavior, and contract state to explain on-chain activity. Task-specific prediction systems assist investigators, while evidence collection and interpretation still require substantial expertise. We present Bgent, a multi-task blockchain investigation agent that combines tutorial-derived guidance with executable checks of factual claims. Offline distillation extracts tool operations and investigation procedures, producing a tool taxonomy and a reusable skill. At runtime, the investigator retrieves evidence and submits an answer with explicit factual claims. A planner selects checkable claims, and independent validators execute Python predicates over saved tool responses. Failed checks trigger revision within the same investigation session. Experiments on Ethereum cover account classification, attack-transaction detection, and smart-contract fault localization. Bgent achieves 96.45% mean F1 on valid attack-detection decisions and 56.86% Recall@1 in fault localization. Removing distilled guidance reduces account-classification F1 from 80.40% to 38.14%. Executable checking corrects specific factual errors in explanations while leaving the reported classification scores unchanged.

1 INTRODUCTION

Public blockchains maintain shared ledgers of transactions, enabling participants to transfer assets and interact with applications without a central ledger operator (Yaga et al., 2018). On account-based platforms, addresses identify accounts that hold balances; contract accounts also store executable code and persistent state. Transactions can transfer assets or invoke smart contracts—programs whose execution updates blockchain state. A single transaction may trigger multiple contract calls and asset movements, linking the behavior of accounts to the execution of application code. These records support investigations at several levels: account classification assigns a role or risk category to an address, attack detection identifies malicious transactions, and fault localization identifies functions implicated in a contract failure or exploit. Such analysis supports both financial investigations (Financial Action Task Force, 2020) and the diagnosis of faults that can directly cause asset losses (Wu et al., 2024).

Public access to records does not make their interpretation automatic. Investigators must connect transaction outcomes, contract execution, and address histories to reconstruct what happened. Existing tools assist this process, but collecting relevant evidence and interpreting complex interactions still require substantial expertise. For smart-contract incidents, Sun et al. (2025) cite an estimate of 16.7 analyst hours on average for manual fault localization and describe the continued use of manual transaction-trace inspection by security teams. This workload motivates agents that can organize evidence collection, reuse investigative procedures, and check the factual statements in their explanations.

Research has advanced each of these tasks. Transaction reconstruction supports detection of new attack types (Wu et al., 2025); sequential and graph-based representations improve account-level detection and de-anonymization (Hu et al., 2023; Jia et al., 2025); and learning-based or agent-based systems localize faults in smart contracts (Wu et al., 2024; Sun et al., 2025). Some account encoders support several account labels, and recent security agents produce explanatory reports. Their evaluation targets nevertheless remain different objects and outputs, such as a transaction

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

score, an address label, or a ranked function list. Reusing investigative steps across these task boundaries remains an open design problem.

Explanations introduce a second gap. Systems that interpret traces and contract code can produce detailed narratives (Wang et al., 2025a), while a DeFi intent agent combines multiple analytical perspectives with an evaluator (Mao et al., 2025). A plausible narrative may still include an incorrect event, transfer, or contract relation, and a collection of correct facts may still fail to justify the final label. Existing work has begun to address verifiability, for example, TracePilot tests proposed patches as executable evidence for fault localization (Zhu et al., 2026).

Challenge 1: Reusing investigative procedures across tasks. Different blockchain data investigation tasks may share operations, such as transaction reconstruction, counterparty inspection, and consideration of alternative explanations, while retaining distinct targets and output schemas. A common AI agent must select the relevant evidence and answer format for each task. The same procedure should also admit a platform-specific evidence adapter, since blockchains expose different transaction, event, and state interfaces.

Challenge 2: Executable checks for factual statements. An AI agent may describe a plausible explanation while misstating an event, amount, or address relation. However, the generated explanation does not completely follow the facts. Even if the decision of agent is correct, there may still be errors in explanation. And these explanations are usually in natural language, which is difficult to verify directly.

To address the aforementioned challenges, we proposed Bgent, which includes two core modules, i.e., tutorial distillation and executable claim checking. Specifically, separate sessions extract tool operations and standard operating procedures (SOPs) from each tutorial. Cross-source synthesis organizes recurring principles into tool families and a shared investigation skill. At runtime, the skill guides evidence collection and reasoning for the requested task. The investigator saves tool responses and submits an answer with explicit claims. A planner selects checkable claims; independent validators test them against the saved evidence and return feedback for revision. Our contributions are:

- We construct a reusable tool taxonomy and investigation skill from separate, source-specific tool and SOP trajectories, then instantiate one agent workflow across three on-chain tasks with distinct outputs.
- We implement a claim-selection and revision loop that checks selected factual statements against saved on-chain evidence and records the selection, scripts, and verdicts.
- We evaluate account classification, attack detection, and fault localization on Ethereum and analyze valid answers separately from end-to-end success. An account-classification ablation measures the effect of distillation, while case analyses document factual corrections made by executable checking.

2 Related Work

2.1 Blockchain analysis tools and predictive models

Analysis infrastructure. Blockchain investigation begins with access to transaction histories and relationships between addresses. BlockSci provides an in-memory analysis platform supporting multiple blockchains and analytical tasks (Kalodner et al., 2020). Practical tracing guides describe how to follow transfers, examine counterparties, and interpret public attribution (Smith, 2019; BlockSec, 2023). Tools expose records and relationships; tutorials explain their investigative use. Our separate tool-Principle and SOP-Principle tracks organize these operations and their conditions of application into agent guidance.

Attack-transaction detection. On-chain attack detectors often learn a representation of a transaction and predict whether it is malicious. Wu et al. (2025) pre-train by reconstructing transaction information and study detection with few or no known examples of a new attack. The resulting

scores identify suspicious transactions but do not provide executable checks of individual explanatory claims. Wang et al. (2025a) combines execution traces and decompiled contract code to diagnose incidents and generate reports, moving beyond classification alone. Its factual-precision evaluation highlights the need to assess whether reports accurately describe the retrieved evidence. Our method focuses on the evidence trail for individual answers across task formats.

Account representation and classification. Hu et al. (2023) model transaction sequences with a pre-trained Transformer for phishing detection and account de-anonymization. Huang et al. (2025) combine local and global graph information for Ethereum account de-anonymization. Jia et al. (2025) integrate transaction semantics and knowledge-graph representations for multiple account-focused fraud tasks. Their outputs are account predictions or embeddings. We examine which retrieved observations support an individual prediction and how the same investigation procedure handles transaction and function-level questions.

2.2 Agentic interpretation and contract fault localization

Fault localization requires linking observed execution and fund flows to candidate source-level causes. Wu et al. (2024) combine reverse engineering with cryptocurrency-flow analysis to rank faulty functions. Sun et al. (2025) coordinate transaction forensics and specialist agents to reason across contracts and provide explanatory evidence. Zhu et al. (2026) extend the setting across transactions and test whether candidate patches block attacks, using executability as evidence for a diagnosis. Function rankings, explanatory reports, and patch tests provide complementary evidence for a fault diagnosis. Our claim-level checks target reported on-chain observations across three task families.

LLM agents have also been used to infer DeFi transaction intent. Mao et al. (2025) coordinate several perspectives and include a cognitive evaluator to refine the interpretation of transaction activity. Our study applies one evidence-first procedure to transaction, account, and function outputs, then tests checkable statements against saved records.

2.3 Tool-using agents and reusable procedural knowledge

Yao et al. (2023) interleave reasoning and actions, allowing an agent to update its plan as tools return observations. Reusable skills extend this interaction loop by retaining procedures across tasks. Voyager builds a library of executable skills through environment interaction and refines programs using execution feedback (Wang et al., 2024). Agent Workflow Memory induces recurring routines from examples or prior agent experience and supplies them to guide later actions (Wang et al., 2025b). Both acquire reusable procedures from task examples or interaction trajectories. Bgent instead uses public investigation tutorials as the source of procedural knowledge. Cross-source synthesis organizes extracted operations and SOPs into tool families and a shared skill for account-, transaction-, and function-level questions.

2.4 Factual verification and executable feedback

Verification methods differ in both the unit they check and the evidence used for revision. RARR retrieves supporting material and edits unsupported passages (Gao et al., 2023). Chain-of-Verification generates verification questions and answers them independently before revising the original response (Dhuliawala et al., 2024). FacTool extracts claims and uses task-relevant tools to assess their factuality across several domains (Chern et al., 2023). CRITIC obtains external feedback through tool interaction and uses it to correct model outputs (Gou et al., 2024). These methods connect revision to additional questions, retrieved evidence, or tool results.

Blockchain explanations contain aggregates, address attributions, and causal interpretations that call for different checks. Bgent selects factual claims and executes per-claim predicates over saved tool responses, retaining the evidence, programs, verdicts, and revisions for inspection. TracePilot tests candidate patches to support a fault diagnosis (Zhu et al., 2026); our predicates check reported observations, such as transfer totals or counterparty labels.

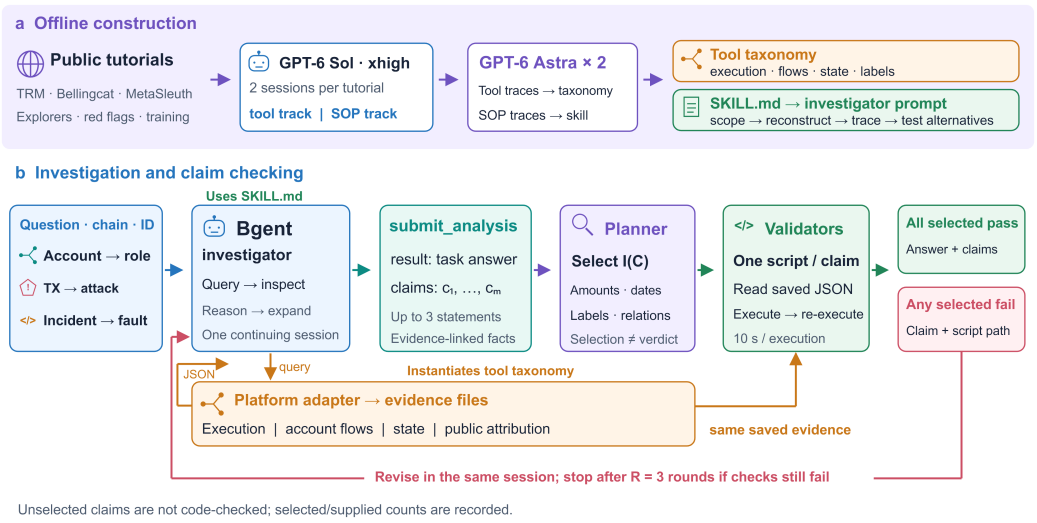


Figure 1: Bgent overview. (a) Separate extraction and synthesis tracks produce a tool taxonomy and a shared skill. (b) The investigator uses these assets to answer three task types. Amber links carry saved evidence to the investigator and validators; red links return failed checks for revision. Passing checks apply only to selected claims.

3 METHOD: BGENT

Bgent combines offline knowledge distillation with runtime investigation and executable claim checking (Figure 1). Offline, separate extraction and synthesis tracks turn public tutorials into a tool taxonomy and a reusable investigation skill (Section 3.1). At runtime, the skill guides an investigator in selecting tools, gathering evidence, and producing an answer with explicit factual claims. A planner selects claims for verification, and independent validators execute programs against the saved evidence. Failed checks return feedback to the investigator for revision (Section 3.2). This workflow serves account classification, attack detection, and fault localization through task-specific questions and output formats; platform adapters supply the underlying evidence.

3.1 TOOL AND SKILL DISTILLATION

Tutorial discovery and source-specific extraction. A GPT-6 Sol agent with xhigh reasoning searches the public web for blockchain transaction-analysis tutorials. The collected guidance includes the TRM Labs *Blockchain Investigator's Flip Book*, Bellingcat's blockchain-tracing walkthrough, the *Cryptocurrency Transaction Investigation Guide*, BlockSec MetaSleuth tutorials, the Etherscan Information Center, FATF and FINTRAC red-flag guidance, and UNODC cryptocurrency-investigation training material (TRM Labs, 2024; Smith, 2019; Canadian Blockchain Consortium, 2025; BlockSec, 2023; Amir, 2020; Financial Action Task Force, 2020; Financial Transactions and Reports Analysis Centre of Canada, 2021; United Nations Office on Drugs and Crime, n.d.). For each tutorial, the agent opens two independent GPT-6 Sol sessions. The first prompt asks for *tool Principles*: progressive, compact operation summaries in the form *chain* → *object* → *operation*, such as *chain* → *transaction* → *inspect event records*. The second prompt asks for *SOP Principles*: concise rules for the investigative sequence and decisions, such as reconstructing a transfer before attributing its sender. Each session retains its analysis and extracted principles as a separate source-specific trajectory.

Cross-source synthesis. One GPT-6 Astra session reads all tool-Principle trajectories, extracts recurring principles, and groups them into a tool taxonomy. A separate GPT-6 Astra session reads all SOP-Principle trajectories, extracts their common rules, and organizes them into the task-agnostic natural-language `SKILL.md`. Figure 2 shows the two branches and their distinct products. The tool taxonomy organizes evidence operations, while the runtime loads the skill text into the investigator's system prompt.

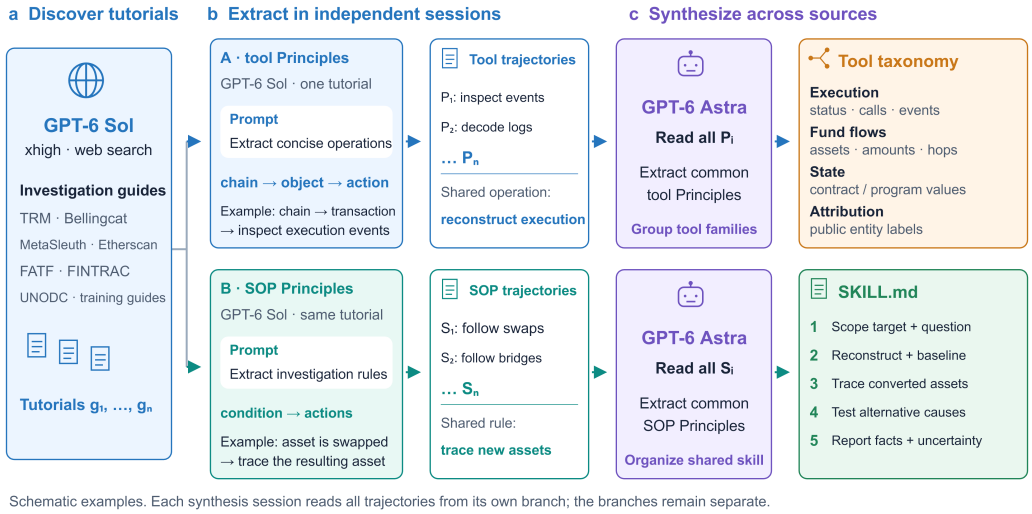


Figure 2: Tool and skill distillation. (a) GPT-6 Sol (xhigh) discovers tutorials. (b) Independent sessions extract tool and SOP Principles from each tutorial, retaining separate trajectories P_i and S_i. (c) Two GPT-6 Astra sessions synthesize the respective trajectory sets into tool families and `SKILL.md`. The intermediate examples illustrate how different operations or rules share a common principle.

Reusable investigation procedure. The skill directs the investigator to define the target and question, reconstruct on-chain events, establish account context, trace funds, and challenge its working hypothesis. The final conclusion identifies supporting observations, alternative explanations, and unresolved uncertainty. The task prompt determines which steps are relevant and supplies the output schema.

Evidence tools. The tool taxonomy distinguishes transaction reconstruction, account and fund-flow history, contract or program state, and public attribution. For a task targeting transaction or account identifier o on platform k, the adapter collects evidence $E(o, k)$ through tool calls. The platform adapter supplies read-only operations for the evidence types available on that chain. The investigator selects operations according to the question and expands the investigation as new addresses or transactions become relevant. Each collection call saves its full JSON response under `tool_data/` and gives the agent a path to that file. A local Python or Node.js tool prints compact aggregates or relevant records. The saved responses also provide the validator's evidence base.

3.2 Claim Selection and Executable Validation

Structured submission and selection. The investigator must call `submit_analysis` with exactly a `result` string, formatted according to the task's output schema, and a `claims` array $C = (c_1, \ldots, c_m)$ with at most $m_{\max} = 3$ entries. The prompt asks for one-sentence, quantitatively checkable claims that directly support the answer. A separate planner receives the claims and returns the one-based indexes $I(C)$ requiring code-based factual verification. It selects externally checkable facts about transactions, addresses, events, amounts, counts, labels, and relations. Definitions and ordinary common knowledge are excluded. The planner determines which claims require verification; each selected claim receives an independent check. Figure 3 traces one illustrative submission through selection, independent scripts, and possible revision.

Per-claim programs and revision. For each $i \in I(C)$, an independent validator writes a Python program p_i for the saved JSON responses and emits a Boolean verdict:

$$v_i = p_i(E(o,k)) \in \{\text{pass}, \text{fail}\}, \qquad \text{accept}(C) = (|C| > 0) \wedge \bigwedge_{i \in I(C)} v_i, \tag{1}$$

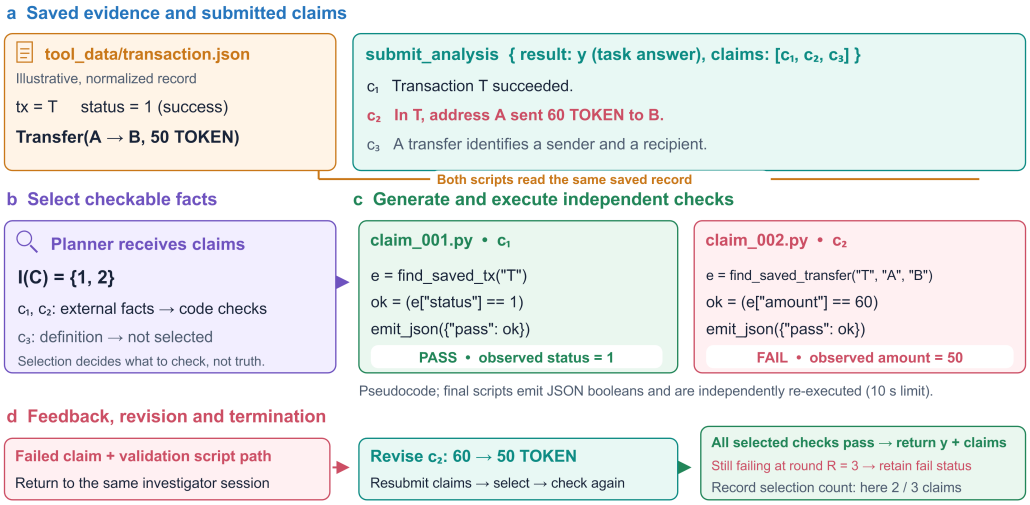


Figure 3: Executable checking of an illustrative submission. (a) Saved evidence records a successful transfer of 50 tokens, while claim c_2 reports 60. (b) The planner selects factual claims c_1 and c_2, excluding the definition c_3. (c) Independent validators check the selected claims against the record. (d) Failure prompts correction and resubmission. Code and record fields are schematic; production scripts emit JSON verdicts and are independently re-executed.

where an empty conjunction is true. Validators identify relevant records by transaction or address identifiers, including when several files share the same tool prefix. They are instructed to use saved evidence without network access or evidence modification. The implementation enforces a 10-second execution timeout and independently re-executes each submitted script. Scripts, per-claim verdicts, and the selected-claim count are retained with the trajectory. If any selected claim fails, its script path and failure details return to the same investigator session. The loop stops when all selected checks pass or after $R = 3$ rounds by default. Unresolved checks retain a failure status at the round limit. An empty $I(C)$ permits acceptance of a nonempty claim set; the recorded selected and supplied counts distinguish this outcome from executed checks.

Each passing verdict records that the generated predicate returned true on the saved evidence for a selected claim. The retained program exposes the records and conditions used in that decision, enabling inspection of how it represents the claim. Section 4.2 examines factual repairs produced by this feedback loop.

4 Experiments

We organize the evaluation around four research questions.

RQ1: Account classification. Can it distinguish account roles from address histories?

RQ2: Attack-transaction detection. Can it identify five attack types from completed transactions?

RQ3: Fault localization. Can it identify implicated contract functions in an incident?

RQ4: Component ablation. How do distilled tools and skills, and executable claim checking, affect task outcomes and factual claims?

4.1 Experimental Setup

Investigator configuration. The saved runs use `deepseek-v4.1-flash` with high reasoning effort as the investigator backbone. The investigator is prompted with a budget of 100 tool-use turns and submits at most three claims per round. The validation loop allows up to three rounds, with each validation script execution limited to 10 seconds.

Platform and evidence adapter. We evaluate on Ethereum, an EVM-compatible chain with abundant public transaction histories, substantial user activity, decoded execution records, and established benchmarks for all three tasks. This setting permits transaction-, address-, and function-level evaluation with one evidence adapter. The adapter exposes `get_evm_transaction` for traces and logs, `get_evm_account_transfers` for external, internal, ERC-20, and ERC-721 activity, `get_contract_params` for available zero-argument getters, and `get_evm_label` for public labels. Account-transfer queries return at most 10,000 records per category; contract getters read the latest state. The account protocol disables direct label lookup for the queried address.

Tasks and datasets. Account classification asks four balanced one-versus-rest questions: ICO wallet, mining, exchange, and phishing/hacking. Seeds 42–44 yield 120 role-address attempts over 109 unique addresses. This is a retrospective investigation setting using present-day histories and state. For post-execution attack detection, the five categories are reentrancy, honeypot, flash-loan attack, integer overflow, and call injection, as in Wu et al. (2025). Each test object is a completed transaction with a binary label for one category. Three balanced seeded draws yield 150 category-level attempts over 88 distinct transaction hashes. Fault localization takes an incident transaction and a supplied fault-cause description, then returns ranked contract-address/function pairs. The local run contains 102 Ethereum cases, whereas the Sun et al. (2025) reference contains 115. The saved experiment plans specify the evaluated identifiers and their labels.

Comparators and metrics. Tables 1, 2, and 3 include published task-specific reference scores. Published scores provide task-specific reference results; the local runs use the sampled objects described above. For local account and attack decisions, we report precision, recall, and F1 on valid Yes/No outputs, plus correct-and-valid answers over all attempts. The account table reports mean scores across three seeds for each role. We report invalid and failed outputs as a proportion of all attempts. For attack detection, we also report the area under the receiver operating characteristic curve (AUC). Fault localization uses Recall@1, the proportion of cases in which the top-ranked function matches a labeled faulty function. Invalid outputs count as failures in end-to-end success.

Ablation and claim-audit protocol. RQ4 compares full Bgent with (i) a version that replaces the distilled taxonomy and `SKILL.md` with generic tool descriptions while retaining the same read-only data access, and (ii) a version that omits claim selection, Python validators, and feedback revision. The backbone, test objects, agent tool permissions, and maximum agent budget remain fixed. We inspect rejected claims against the saved tool records and their revised versions to identify factual errors corrected by validator feedback.

4.2 Results

RQ1: Account classification. Account classification showed substantial variation across roles (Table 1): mining reached 93.94% F1, whereas ICO wallets reached 72.28%, a 21.66-point gap. The shared procedure therefore achieved different levels of accuracy across account categories. Pooling roles within each seed yielded 91.67% recall and 72.28% precision. This asymmetry indicates that recovering positive accounts was easier than excluding accounts outside the queried role. The result motivates more discriminative checks of alternative account roles before committing to a positive label.

Answer completion introduced a separate source of error. Valid decisions accounted for 95.0% of attempts, while correct-and-valid answers accounted for 74.2%. Incorrect valid decisions therefore contributed 20.8% of attempts, compared with 5.0% for failed or invalid outputs. Role discrimination thus accounted for a larger share of errors than output completion.

RQ2: Attack-transaction detection. In attack detection tasks, Bgent surpasses all comparison methods, even fully supervised techniques. Mean F1 on valid decisions was 96.45%, with precision of 1.00 in every category except honeypots (Table 2). Flash-loan attacks combined 1.00 precision with 0.78 recall, indicating a conservative decision boundary for this category. A flash loan can participate in both legitimate and malicious execution; identifying borrowing alone does not establish an exploit. The recall gap motivates checking how borrowed funds interact with contract state and subsequent repayment when the initial evidence is inconclusive. Honeypots showed the opposite trade-off, with 1.00 recall and 0.92 precision, motivating stronger exclusion of benign behavior.

Table 1: Account classification. Top: published micro-F1 (%) under Amount sampling from Huang et al. (2025). Bottom: local one-versus-rest results from three seeded runs, reported as mean F1 (%) on valid outputs. Local results use different sampled addresses and a different metric protocol.

Method	Eth-ICO	Eth-Mining	Eth-Exchange	Eth-Phish&Hack
Trans2Vec + LR	73.77	71.41	57.52	68.67
GCN	87.57	75.25	90.23	96.49
GAT	88.44	80.28	91.01	95.87
Ethident	94.05	86.38	93.16	97.93
ETGNN	94.50	86.89	92.74	95.18
LGA	96.95	89.67	93.51	98.18
Bgent	72.28	93.94	79.05	85.64

Table 2: Attack-transaction detection. Each attack cell is precision/recall; the final column reports AUC. Literature results are from Wu et al. (2025). Local Bgent values are means over three runs on valid decisions.

Supervised	Method	Reentrancy	Honeypot	Flash loan	Integer overflow	Call injection	AUC
Full	DeFiScanner	1.00/0.83	0.00/0.00	0.85/0.32	1.00/0.65	0.67/1.00	0.78
Full	DEFIER	0.00/0.00	0.00/0.00	1.00/0.03	0.36/0.45	1.00/0.05	0.54
Full	MoTS	0.99/0.70	0.00/0.00	0.50/0.40	0.83/0.11	0.00/0.00	0.64
Full	Wu et al.	1.00/0.95	0.74/0.35	0.85/0.36	0.98/0.70	0.95/0.70	0.94
Few-shot	Wu et al. (10-shot)	1.00/0.66	0.79/0.07	0.51/0.16	0.99/0.29	0.99/0.37	0.74
Few-shot	Wu et al. (30-shot)	1.00/0.84	0.73/0.38	0.39/0.25	0.99/0.30	0.95/0.85	0.91
Zero-shot	ChatGPT (MetaSuites)	0.00/0.00	0.00/0.00	1.00/0.15	0.00/0.00	0.00/0.00	0.515
Zero-shot	Bgent	1.00/1.00	0.92/1.00	1.00/0.78	1.00/1.00	1.00/0.93	0.96

Table 3: Fault localization Recall@1 (%) by complexity. DAppFL and FaultSeeker scores are from Sun et al. (2025); Bgent scores are from the local evaluation. Δ denotes the reported Recall@1 difference between Bgent and FaultSeeker.

Complexity of Fault	DAppFL	FaultSeeker	Bgent	ΔScore
Moderate	0.00	71.43	64.28	-7.15
Complex	13.73	58.82	59.57	0.75
Extreme	0.00	36.00	55.56	19.56
Overall	6.09	50.43	56.86	6.43

Correct-and-valid answers accounted for 91.3% of attempts. Failed or invalid outputs accounted for 5.3%, exceeding the 3.3% with incorrect valid decisions. Unlike account classification, attack detection thus had a larger completion bottleneck than a decision-error bottleneck. Recovery from tool or output failures could complement improvements in category-specific reasoning.

RQ3: Fault localization. Bgent achieved 56.86% Recall@1, with valid outputs for all cases (Table 3). Recall@1 decreased from 64.28% for Moderate faults to 59.57% for Complex and 55.56% for Extreme faults, an 8.72-point span. Relative to the published FaultSeeker results, the reported Recall@1 differences were 19.56 points for Extreme faults, 0.75 for Complex faults, and -7.15 for Moderate faults. In the Moderate category of 14 samples, the latter difference corresponds to one additional incorrectly localized fault. Overall, the local Recall@1 was 6.43 points above the published reference score; the evaluated case sets are specified in the dataset description.

RQ4: Component ablation. In account classification, removing distillation reduces mean F1 from 80.40% to 38.14% (Table 4), a 42.26-point drop, driven primarily by a 64.32-point loss in recall, compared with a 5.61-point loss in precision. With the same underlying data access, distilled guidance primarily improved recognition of positive accounts. The observed gain reflects the combined guidance of the tool taxonomy and investigation skill.

Table 4: Account-classification ablation. Precision, recall, and F1 are mean percentages on valid Yes/No decisions, pooled over the four roles within each run.

Variant	Precision	Recall	F1
Full Bgent	72.28	91.67	80.40
Without distillation	66.67 (-5.61)	27.35 (-64.32)	38.14 (-42.26)
Without code checking	72.28 (-0.00)	91.67 (-0.00)	80.40 (-0.00)

Evidence	Initial claim ×	After validation ✓
Amount scope	1–10 ETH integer outflows: **1,030 tx** / **34,416 ETH**	All integer ETH: 1,030 tx / 34,416 ETH. 1–10 ETH: **715 tx** / **2,693 ETH**
Labels	`0xbfc39...`: **Poloniex 2** `0x209c47...`: **Poloniex 5**	`0xbfc39...`: **Poloniex 5** `0x209c47...`: **Poloniex 2**
Transfer date	50 ETH sent on **2015-08-08**	50 ETH sent on **2015-08-07** at 03:52:51 UTC
Flow totals	Four incoming transfers: **3.510021 ETH** Outgoing: 3.510021 ETH	Four incoming transfers: **3.510042 ETH** Outgoing: 3.510021 ETH

Figure 4: Four factual claim corrections in account investigations. Colored highlights mark the incorrect value or label in the submitted claim and its correction after executable checking of saved transfer or label records.

Removing code checking left all three classification metrics unchanged, yet the before-and-after claims expose a benefit that label metrics do not capture (Figure 4). Checking corrected an aggregation over the wrong amount range, swapped counterparty labels, an incorrect date, and conflated incoming and outgoing totals. These errors concern the factual basis of an explanation even when its final account label is unchanged. Distillation and checking therefore served complementary roles in these experiments: the former improved task decisions, while the latter repaired specific factual statements used to communicate those decisions.

5 Conclusion

We presented Bgent, a blockchain investigation agent that combines tutorial-derived tools and skills with executable claim checks. Experiments on Ethereum demonstrate its applicability to account classification, attack-transaction detection, and fault localization. Distilled guidance improves account-classification F1 by 42.26 percentage points, primarily through higher recall. Code checking preserves classification scores while correcting factual errors in explanations. The account ablation supports the combined distilled guidance, while the correction cases demonstrate how executable feedback repairs individual factual claims. Future work will extend the evidence adapters to additional blockchains and evaluate claim-checking coverage and accuracy across tasks.

Reproducibility Statement

Code is available at `https://anonymous.4open.science/r/bgent-DE15`. Experiment plans record seeds and evaluation objects; saved tool responses, validation scripts, and trajectories support inspection and replay of claim checks.

REFERENCES

Raja Amir. Understanding an Ethereum transaction. Etherscan Information Center, 2020. URL https://info.etherscan.com/understanding-an-ethereum-transaction/.

BlockSec. How to track and analyze stolen funds with MetaSleuth: A comprehensive guide for crypto investigations, 2023. URL https://blocksec.com/blog/crypto-tracking-starting-with-a-transaction-1.

Canadian Blockchain Consortium. Cryptocurrency transaction investigation guide, 2025. URL https://www.canadablockchain.ca/wp-content/uploads/2025/02/Cryptocurrency-Transaction-Investigation-Guide-Final.pdf.

I-Chun Chern, Steffi Chern, Shiqi Chen, Weizhe Yuan, Kehua Feng, Chunting Zhou, Junxian He, Graham Neubig, and Pengfei Liu. FacTool: Factuality detection in generative AI—a tool augmented framework for multi-task and multi-domain scenarios, 2023. URL https://arxiv.org/abs/2307.13528.

Shehzaad Dhuliawala, Mojtaba Komeili, Jing Xu, Roberta Raileanu, Xian Li, Asli Celikyilmaz, and Jason Weston. Chain-of-Verification reduces hallucination in large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 3563–3578, 2024. doi: 10.18653/v1/2024.findings-acl.212. URL https://aclanthology.org/2024.findings-acl.212/.

Financial Action Task Force. Virtual assets red flag indicators of money laundering and terrorist financing, 2020. URL https://www.fatf-gafi.org/en/publications/Methodsandtrends/Virtual-assets-red-flag-indicators.html.

Financial Transactions and Reports Analysis Centre of Canada. Money laundering and terrorist financing indicators: Virtual currency transactions, 2021. URL https://fintrac-canafe.canada.ca/guidance-directives/transaction-operation/indicators-indicateurs/vc_mltf-eng.

Luyu Gao, Zhuyun Dai, Panupong Pasupat, Anthony Chen, Arun Tejasvi Chaganty, Yicheng Fan, Vincent Zhao, Ni Lao, Hongrae Lee, Da-Cheng Juan, and Kelvin Guu. RARR: Researching and revising what language models say, using language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 16477–16508, 2023. doi: 10.18653/v1/2023.acl-long.910.

Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Nan Duan, and Weizhu Chen. CRITIC: Large language models can self-correct with tool-interactive critiquing. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=WSrRF5Wy6v.

Sihao Hu, Zhen Zhang, Bingqiao Luo, Shengliang Lu, Bingsheng He, and Ling Liu. BERT4ETH: A pre-trained transformer for Ethereum fraud detection. In *Proceedings of the ACM Web Conference 2023*, pp. 2189–2197, 2023. doi: 10.1145/3543507.3583345. URL https://arxiv.org/abs/2303.18138.

Jiahui Huang, Teng Huang, Changyu Dong, Sisi Duan, and Yan Pang. Hierarchical network with local-global awareness for Ethereum account de-anonymization. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 55(9):5839–5852, 2025. doi: 10.1109/TSMC.2025.3571795.

Yifan Jia, Ye Tian, Liguo Zhang, Yanbin Wang, Jianguo Sun, and Liangliang Song. KGBERT4Eth: A feature-complete transformer powered by knowledge graph for multi-task Ethereum fraud detection, 2025. URL https://arxiv.org/abs/2509.03860.

Harry Kalodner, Malte Möser, Kevin Lee, Steven Goldfeder, Martin Plattner, Alishah Chator, and Arvind Narayanan. BlockSci: Design and applications of a blockchain analysis platform. In *29th USENIX Security Symposium*, pp. 2721–2738, 2020. URL https://www.usenix.org/conference/usenixsecurity20/presentation/kalodner.

Qian'ang Mao, Yuxuan Zhang, Jiaman Chen, Wenjun Zhou, and Jiaqi Yan. Know your intent: An autonomous multi-perspective LLM agent framework for DeFi user transaction intent mining, 2025. URL https://arxiv.org/abs/2511.15456.

Brenna Smith. Tracking illicit transactions with blockchain: A guide, featuring mueller, 2019. URL https://www.bellingcat.com/resources/how-tos/2019/02/01/tracking-illicit-transactions-with-blockchain-a-guide-featuring-mueller/.

Kairan Sun, Zhengzi Xu, Kaixuan Li, Lyuye Zhang, Yuqiang Sun, Liwei Tan, and Yang Liu. FaultSeeker: LLM-empowered framework for blockchain transaction fault localization. In *Proceedings of the 40th IEEE/ACM International Conference on Automated Software Engineering*, pp. 104–116, 2025. doi: 10.1109/ASE63991.2025.00017.

TRM Labs. The blockchain investigator's flip book: A guide for law enforcement, 2024. URL https://www.trmlabs.com/guides/the-blockchain-investigators-flip-book-a-guide-for-law-enforcement.

United Nations Office on Drugs and Crime. Cryptocurrency training: Residential courses, n.d. URL https://syntheticdrugs.unodc.org/uploads/syntheticdrugs/res/library/cybercrime_html/Cryptocurrency_training_-_Residential_Courses.pdf.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *Transactions on Machine Learning Research*, 2024. URL https://openreview.net/forum?id=ehfRiF0R3a.

Shuzheng Wang, Yue Huang, Zhuoer Xu, Yuming Huang, and Jing Tang. TraceLLM: Security diagnosis through traces and smart contracts in Ethereum, 2025a. URL https://arxiv.org/abs/2509.03037.

Zora Zhiruo Wang, Jiayuan Mao, Daniel Fried, and Graham Neubig. Agent workflow memory. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 63897–63911. PMLR, 2025b. URL https://proceedings.mlr.press/v267/wang25bx.html.

Zhiying Wu, Jiajing Wu, Hui Zhang, Ziwei Li, Jiachi Chen, Zibin Zheng, Qing Xia, Gang Fan, and Yi Zhen. DAppFL: Just-in-time fault localization for decentralized applications in Web3. In *Proceedings of the 33rd ACM SIGSOFT International Symposium on Software Testing and Analysis*, pp. 137–148, 2024. doi: 10.1145/3650212.3652116.

Zhiying Wu, Jiajing Wu, Hui Zhang, Zibin Zheng, and Weiqiang Wang. Hunting in the dark forest: A pre-trained model for on-chain attack transaction detection in Web3. In *Proceedings of the ACM on Web Conference 2025*, pp. 4519–4530, 2025. doi: 10.1145/3696410.3714928.

Dylan Yaga, Peter Mell, Nik Roby, and Karen Scarfone. Blockchain technology overview. Technical Report NISTIR 8202, National Institute of Standards and Technology, 2018. URL https://doi.org/10.6028/NIST.IR.8202.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

Xuanyu Zhu, Zhiying Wu, Tao Wang, Ying Yan, Wei Zhou, Jiajing Wu, Zigui Jiang, and Zibin Zheng. TracePilot: Self-verifiable framework for decentralized applications fault localization across transactions. In *Proceedings of the 35th ACM SIGSOFT International Symposium on Software Testing and Analysis*, 2026. URL https://conf.researchr.org/details/issta-2026/issta-2026-research-papers/201/TracePilot-Self-verifiable-Framework-for-Decentralized-Applications-Fault-Localizati. Accepted at ISSTA 2026.