

TxSum: User-Centered Ethereum Transaction Understanding with Micro-Level Semantic Grounding

Zifan Peng^{1,2} Jingyi Zheng¹ Yule Liu¹ Huaiyu Jia¹ Qiming Ye¹ Jingyu Liu¹
Xufeng Yang³ Mingchen Li⁴ Qingyuan Gong⁵ Xuechao Wang¹ Xinlei He^{2*}

¹The Hong Kong University of Science and Technology (Guangzhou) ²Wuhan University

³China National Tobacco Corporation Liaoning Province Company

⁴University of North Texas ⁵Fudan University

Abstract

Understanding the economic intent of Ethereum transactions is critical for user safety, yet current tools expose only raw on-chain data or surface-level intent, leading to widespread “blind signing” (approving transactions without understanding them). Through interviews with 16 Web3 users, we find that effective explanations should be structured, risk-aware, and grounded at the token-flow level. Motivated by these findings, we formulate *TxSum*, a new domain-grounded NLP task for DeFi transaction explanation, and construct a dataset of 187 complex Ethereum transactions with 2,375 token-flow annotations and transaction-level summaries.

We further introduce *MATEX*, a grounded multi-agent framework for high-stakes transaction explanation. It selectively retrieves external knowledge under uncertainty and audits explanations against raw traces to improve token-flow-level factual consistency. *MATEX* achieves the strongest overall explanation quality, especially on micro-level factuality and intent quality. It improves user comprehension on complex transactions from 52.9% to 76.5% over the strongest baseline and raises malicious-transaction rejection from 36.0% to 88.0%, while maintaining a low false-rejection rate on benign transactions.

1 Introduction

Understanding the economic intent of an Ethereum transaction is crucial for both proactive risk mitigation and post-incident analysis. It enables users to preview outcomes and avoid irreversible errors before signing, and supports regulatory compliance, anti-money laundering, and forensic diagnosis after exploits—such as the TraceLLM (Wang et al., 2025) case, where on-chain analysis uncovered attack vectors in major DeFi incidents.

However, this understanding is critically lacking in practice. Modern DeFi protocols can coordinate multiple contracts and even multiple chains, as illustrated by bridgeless cross-chain options (Peng et al., 2025b); their resulting transactions are highly compositional, making correct human review difficult and contributing to “blind signing” (Liu et al., 2024; Castillo et al., 2024)—a phenomenon with severe consequences. In early 2025, attackers stole \$1.5B from Bybit by tricking operators into approving a malicious transaction that appeared innocuous in their interface (BlockSec, 2025a). In routine interactions, new users struggle with dual-sign patterns (approve + swap), while experienced users rely on “reject unlimited approvals”, a heuristic that fails across contracts.

These incidents reveal a fundamental gap: current tools expose raw on-chain data but fail to **translate it into human-understandable economic narratives**. While tools like MetaSuites (BlockSec, 2025b), EigenPhi (EigenPhi, 2025), and Tenderly (Tenderly, 2025) expose detailed token flows and contract interactions, they stop at the syntactic level. They expose detailed execution artifacts rather than synthesizing what these low-level events mean economically for users.

As Figure 1 shows, a user’s simple intent (e.g., “swap ETH for DAI”) often spans multiple contracts. Raw traces show *what* happened but not *why*, leaving users unable to distinguish a routine swap from a high-risk operation. Three research questions guide our study:

- **RQ1:** How do Web3 users understand transaction details when signing?
- **RQ2:** What do users expect from transaction explanations in terms of content and format?
- **RQ3:** How to design a framework and how the generated explanations by it affect users’ comprehension and signing behavior?

To address this gap, we conducted semi-

*Corresponding author: Xinlei He (xinlei.he@whu.edu.cn).

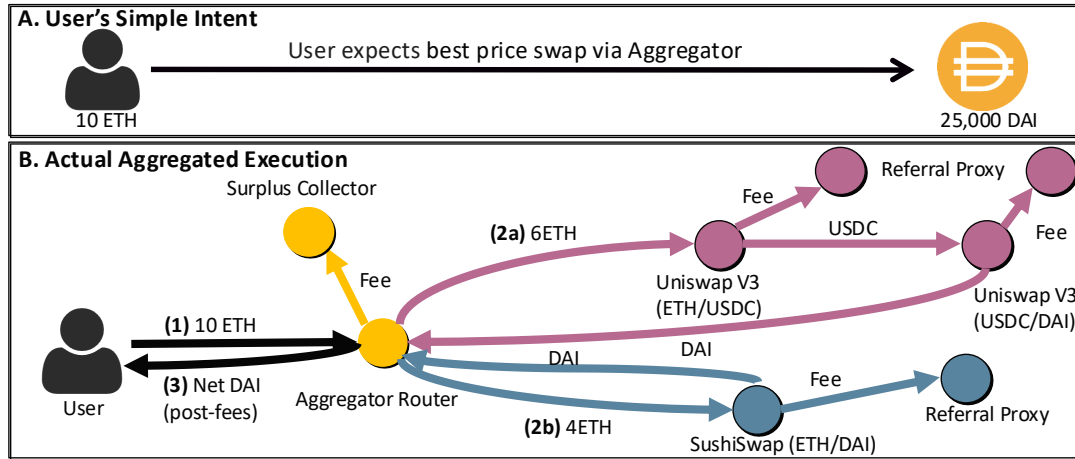


Figure 1: Mismatch between user intent and on-chain execution. A simple swap intent is realized through a complex and opaque transaction trace. The example is simplified for clarity.

structured interviews with 16 Web3 users spanning traders, developers, and casual users. Three findings directly shaped our formulation: (1) blind signing is widespread even among experienced users; (2) existing tools expose syntactic traces but not user-facing economic semantics; and (3) users naturally reason at the token-flow level while still expecting a concise transaction-level summary. Full study details are provided in Appendix A.

Guided by these findings, we formulate the paper around one central thesis: we introduce a new domain-grounded NLP task for DeFi transaction explanation and a grounded method for improving micro-level factual explanation under high-stakes reasoning. As illustrated in Figure 2, our study proceeds from user interviews, to task and dataset construction, to grounded explanation modeling, and finally to multi-faceted evaluation.

Concretely, we propose *TxSum*, a task and schema for transaction understanding at the token-flow level. *TxSum* defines 5 semantic attributes (see Section 2) that ground explanations in step-wise, verifiable actions aligned with user mental models. We then construct a dataset of 187 Ethereum transactions containing 2,375 token-flow annotations. Each transaction includes (1) a natural-language summary and (2) fine-grained flow-level labels following our predefined schema.

We further introduce *MATEX*, a grounded multi-agent framework motivated by how human experts analyze transactions. Rather than using unrestricted tool-augmented generation, *MATEX* implements a dual-process reasoning workflow (Booch et al., 2021): fast pattern recognition (System 1) flags uncertainty and triggers targeted evidence-

seeking investigation (System 2), followed by adversarial factual auditing against raw traces. The design explicitly targets the core NLP challenge in this setting: generating faithful, micro-level explanations for complex DeFi traces without hallucinating unseen protocol behavior.

We evaluate *MATEX* through automatic, expert, and user studies. Results show that *MATEX* consistently outperforms strong baselines—including monolithic prompting, VLM prompting with flow visualization, a single-agent tool user, Few-shot CoT, and a RAG pipeline—in factuality, intent quality, and risk coverage (see Section 4). In summary, we make the following contributions:

- We identify user-centered explanatory needs through semi-structured interviews with 16 diverse Web3 participants and define a structured schema for transaction understanding grounded in real usage.
- We introduce *TxSum*, a new domain-grounded NLP task and dataset for DeFi transaction explanation, pairing natural-language summaries with 2,375 step-wise semantic annotations across 187 complex, real-world transactions.
- We propose *MATEX*¹, a grounded multi-agent framework that combines uncertainty-triggered retrieval, evidence synthesis, and adversarial factual auditing to generate accurate, risk-aware explanations from raw on-chain data, significantly improving user comprehension and reducing “blind signing” across automatic, expert, and user evaluations.

¹<https://github.com/sfofgalaxy/matex>

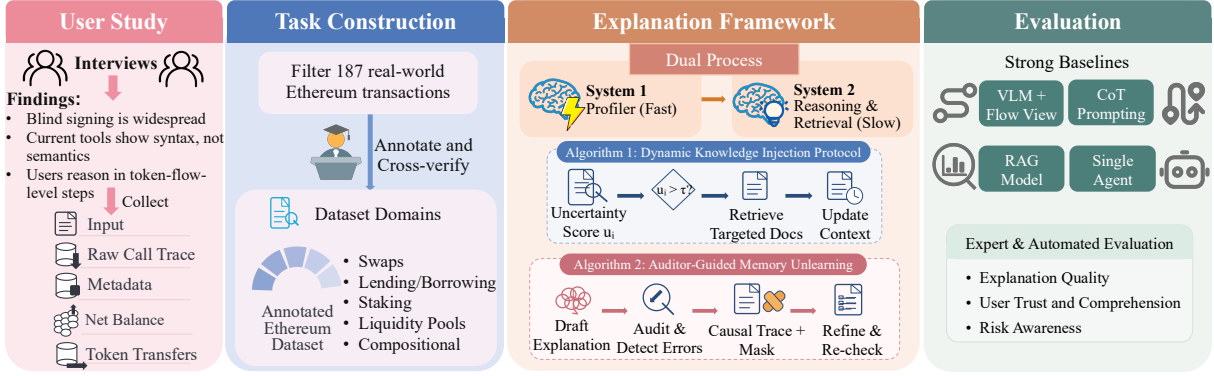


Figure 2: Overview of our study. We first conduct user interviews to identify the explanatory needs behind blind signing, which motivate *TxSum*, a new domain-grounded NLP task and dataset for DeFi transaction explanation. We then introduce *MATEX*, a grounded multi-agent method for this task, and evaluate it against strong baselines using automatic, expert, and user-centered assessments.

2 Understanding Transaction Flow

Our user interviews directly motivated *TxSum*’s schema. While concurrent work (Mao et al., 2025) focuses on transaction-level intent mining over a predefined taxonomy, *TxSum* targets user-centered transaction understanding at two levels: (1) a concise transaction summary that explains the overall intent and outcome, and (2) token-flow annotations that capture step-wise economic actions. This grounds micro-level explanations in user-relevant actions and risks.

Task Definition. *TxSum* is the task of generating a faithful explanation of a DeFi transaction from strictly on-chain observable data. The input consists of four components: (1) transaction metadata (e.g., sender, value, timestamp), (2) the full call trace, (3) a sequence of token transfers (ERC-20/721), and (4) net balance changes for all involved accounts.

The output comprises two parts: (1) A transaction-level summary: a 3-4 sentence natural-language narrative that integrates the economic purpose, execution pathway, and outcomes of the transaction, including critical non-token state changes such as approvals, ownership transfers, or admin control updates. (2) Token-flow annotations: for each token transfer, five semantic attributes—action_type, intent, mechanism, precondition, and result—that ground the summary in step-wise, verifiable actions. The full specification of the five attributes is detailed in Section 2.1. The two output levels are complementary: flow annotations make each economic action checkable, while the summary preserves the concise view requested in our interviews. Requiring both also prevents

a fluent transaction-level narrative from masking unsupported intermediate claims.

The task formulation is broadly applicable to DeFi transaction explanation, but the current schema and implementation assume EVM-style traces and token flows. EVM-compatible chains such as Arbitrum, Optimism, and Base require only limited tooling adaptation, whereas non-EVM chains such as Solana require changes to both the input adapters and parts of the semantic schema.

2.1 Identified Attributes

User interviews reveal that effective explanations must articulate the economic logic behind each token flow. To formalize this, we define five semantic attributes for every flow:

1. action_type, a standardized label for one of 14 financial primitives (e.g., *redeem*, *deposit*) (defined in Table 5 in Appendix);
2. intent, the user’s high-level purpose (e.g., “swap out USDT for USDC”);
3. mechanism, the technical pathway indicating which contract and function were called (e.g., “UniswapV3#exactInput”);
4. precondition, the on-chain conditions required for success (e.g., USDC approved and balance > 10); and
5. result, the factual state change observed after execution (e.g., +1500 USDC balance).

These attributes turn raw transaction traces into explanations that are both easy to follow and explicit about risks, and that are aligned with user mental models. Detailed definitions and examples are provided in Appendix B.

2.2 TxSum Dataset

Data Collection. To build a dataset that reflects real-world DeFi complexity while remaining feasible for expert annotation, we first sampled 500 Ethereum transactions from a full node over 2025. Specifically, we sampled 100 transactions from each of five expert-defined semantic categories: swaps (including multi-hop and aggregator-routed), lending/borrowing cycles, staking and delegation, liquidity pool operations, and compositional or cross-protocol interactions (e.g., flash loans, arbitrage, or custom logic not fitting the other categories). Because DeFi transactions often combine behaviors, we use these categories only for sampling and non-exclusive coverage profiling, not to impose a single label on each transaction.

We then filtered the 500 sampled transactions by token-flow length. We removed transactions with fewer than 5 token flows because they were usually simple transfers or otherwise too easy for step-wise semantic annotation. We also removed transactions with more than 30 token flows because they were too costly to annotate reliably. After filtering, the final *TxSum* dataset contains 187 transactions. For each transaction, we collect transaction metadata and the full call trace from an Ethereum archival node, ordered token transfers and net balance changes from EigenPhi (EigenPhi, 2025), and the token-flow graph from EigenPhi.

Data Annotation. The 187 transactions were annotated by six graduate-level annotators whose research focuses on DeFi. Annotators were organized into three fixed pairs, and each pair independently annotated a disjoint subset of 62 or 63 transactions. For every transaction, the two annotators in the assigned pair independently completed the full annotation, including token-flow labels and the transaction-level summary. Any disagreements were then resolved by a third annotator. Our annotation pipeline mirrors the cognitive process users described in interviews: (1) understand token movement, (2) infer purpose and mechanism, and (3) synthesize a coherent narrative. The process consists of three steps:

Step 1: Transaction Overview. Annotators first review all collected data to understand the transaction’s purpose. They then draft a coarse summary to guide labeling; it is excluded from the dataset.

Step 2: Semantic Labeling. For each token transfer, annotators label five attributes derived from user interviews: `action_type`, `intent`, `mechanism`,

`precondition`, and `result`.

Step 3: Narrative Synthesis. Annotators write a 3-4 sentence natural-language summary that strictly reflects observable evidence, without invoking external knowledge, protocol reputations, or speculative intent. Detailed annotation process and examples are provided in Appendix C.

Data Statistics. The final dataset contains 187 annotated transactions and 2,375 token-flow annotations, with an average of 12.7 token-flow steps per transaction. It focuses on compositionally complex DeFi interactions rather than routine transfers or short single-step executions. The resulting summaries average 3.3 sentences (101.3 tokens). Based on a heuristic of protocol and control-flow complexity, we categorize 43 transactions as simple (single protocol), 90 as medium (2–3 protocols), and 54 as complex (≥ 4 protocols or nested logic). The five sampled scenarios cover swaps, lending/borrowing, staking, liquidity-pool operations, and cross-protocol composition, which align with major DeFi sectors identified by both current ecosystem taxonomies and prior protocol surveys (DeFiLlama, 2026; Shah et al., 2023).

Moreover, 20.2% of all contracts involved in the dataset lack a verified ABI, so the evaluation includes interactions for which protocol semantics cannot be recovered from verified source code alone. Full statistics are shown in Appendix C.2. To assess annotation consistency at the token-flow level, we evaluate the agreement of annotators. For `action_type`, we compute inter-annotator agreement using Cohen’s κ , yielding `action_type` = 0.98.

For the natural language fields `intent`, `mechanism`, `precondition`, and `result`, exact-form agreement would underestimate reliability because annotators may produce semantically equivalent paraphrases. We therefore report pairwise semantic agreement using BERTScore F1, computed between the two independently written annotations for each token flow and averaged over the dataset. Specifically, for each token flow, we compute BERTScore F1 between the two independently written annotations using the official BERTScore (Zhang et al., 2020b) implementation with a roberta-large backbone and baseline rescaling enabled, and then average the resulting scores over the dataset; all four fields score above 0.91 (see details in Table 6 in Appendix). These results indicate high consistency across attributes, supporting the reliability of the dataset.

3 Multi-Agent Framework: *MATEX*

To address **RQ3** (how to explain it), we start from how human experts analyze complex transactions. They do not rely on memorized patterns alone; instead, they follow a clear workflow: (1) quickly scan token flows for familiar patterns, (2) stop when something is unclear, (3) look up documentation or code, (4) synthesize the evidence into an explanation, and (5) check for hidden risks.

Dynamic Knowledge Injection Protocol. The key decision is when to move from System 1 to System 2. Most token flows are routine and can be interpreted from on-chain evidence alone. Only a small number are ambiguous and need extra protocol context. For this reason, *MATEX* uses Dynamic Knowledge Injection Protocol (DKIP). The Profiler computes a semantic uncertainty score u_i for each token flow t_i . If u_i is low, the system keeps a lightweight explanation path. If u_i is above a threshold τ , the Investigator retrieves protocol evidence such as ABI entries or verified documentation. The Profiler also emits a retrieval key when escalation is needed. This design avoids unnecessary retrieval and keeps the context focused on uncertain flows.

MATEX does not require a verified contract or an available ABI. When ABI evidence is missing, it falls back to observable token transfers, event logs, net balance changes, call-trace structure, and call direction; the resulting explanation uses conservative descriptions such as “swap-like routing” and emits an uncertainty warning instead of asserting unverifiable function semantics.

External information can be misleading or unreliable; prior prompt-based curriculum and contrastive methods therefore treat credibility assessment as an explicit prediction problem (Peng et al., 2023, 2025a). Because AI-powered search can also surface malicious or unverified external content (Luo et al., 2025), retrieved context is not accepted as ground truth: the Safety Auditor checks resulting claims against raw traces.

Auditor-Guided Context Pruning. After retrieval, System 2 produces and checks the explanation. The Synthesizer combines on-chain evidence, retrieved protocol context, and flow-level hypotheses to produce a draft summary. The Safety Auditor then checks this draft against the raw trace. Here *MATEX* uses Auditor-Guided Context Pruning (AGCP). The intuition is that unsupported claims can remain in the working context and af-

fect later drafts. Instead of only appending critique, AGCP identifies the error-causing context P_{mask} and removes it before the next refinement step. This reduces repeated hallucinations across iterations.

This design is connected to the “Lost in Conversation” failure mode, in which language models over-rely on earlier incorrect attempts in a multi-turn context (Laban et al., 2026). AGCP acts as a structural mitigation for this form of context contamination because it removes unsupported intermediate content rather than merely appending a correction. The full pseudocode of DKIP and AGCP are provided in Appendix D.1.

- **Profiler** implements System 1. It scans token flows and computes a semantic uncertainty score u_i for each token flow t_i .
- **Investigator** supports System 2 with retrieving live protocol information only when u_i is above the threshold τ , following DKIP.
- **Synthesizer** is the main reasoning module in System 2, which combines on-chain evidence, retrieved protocol context, and flow-level hypotheses to produce a transaction summary.
- **Safety Auditor** checks the draft explanation against the raw trace and applies AGCP when it finds unsupported facts or missing risks.

These components realize the full workflow of *MATEX*: System 1 decides whether a flow can be handled directly or should be escalated, DKIP controls what evidence enters System 2, the Synthesizer turns that evidence into a flow-grounded explanation, and AGCP removes unsupported reasoning before the next draft. This design supports our main goal: grounded micro-level factual explanation rather than only fluent transaction narration. In our implementation, all agents use Qwen3-Max as the backbone LLM and are orchestrated with LangGraph (LangChain, 2025) to support stateful and iterative interactions. The agents share structured tools for on-chain data access and protocol document retrieval. Full implementation details are provided in Appendix D.

4 Evaluations and Results

Our interviews reveal that existing interfaces fail to support safe DeFi interaction: users sign transactions they don’t understand, often missing latent risks. To address **RQ3** (user comprehension and signing behavior), we give 3 evaluation objectives (EOs) that map directly to user-identified needs:

- **EO1: Explanation Quality.** We evaluate the accuracy, intent coverage, outcome correctness, and clarity of generated summaries using automatic metrics and expert annotation.
- **EO2: User Comprehension.** We measure users’ understanding, trust, and clarity via a controlled user study.
- **EO3: Risk Awareness.** We evaluate whether users can identify real-world security threats with the help of *MATEX*.

4.1 Baselines

We compare *MATEX* against five representative baselines: (1) **Monolithic LLM.** An LLM that processes the full input (trace, metadata) in one pass. (2) **Few-shot CoT.** A prompting baseline using the same Qwen3-Max backbone with 2 in-context demonstrations and an explicit chain-of-thought reasoning template. (3) **VLM.** A VLM augmented with a token-flow graph. (4) **RAG.** A retrieval baseline that extracts contract addresses, function signatures, and protocol names from the trace, retrieves external documents with a fixed workflow, and appends the retrieved evidence directly to the model. (5) **Single-Agent.** A ReAct-style agent using Qwen3-Max as its backbone, with access to the same web search and Etherscan tools as *MATEX*.

To ensure a fair comparison, all text-only baselines (*MATEX*, Monolithic LLM, Single-Agent, Few-shot CoT, Pure RAG) use the same Qwen3-Max (Team, 2025) backbone, while the visualization baseline uses Qwen3-VL-Plus (Report, 2025) due to its multimodal input requirement. All retrieval-enabled systems are granted the same retrieval permissions, tool budget, and temporal alignment: the same Etherscan API access, the same web-search scope, the same maximum number of external retrieval calls, and the same transaction-time protocol information window. Single-Agent can call the same web search and Etherscan tools as *MATEX*, while Pure RAG pipeline uses the same retrieval sources but without role specialization, iterative auditing, or uncertainty-triggered control. This setup isolates the contribution of *MATEX*’s selective retrieval and auditing framework rather than confounding it with information asymmetry.

All systems receive identical on-chain inputs and output the result for the whole transaction. We assess EO1 via automatic metrics and expert annotation, EO2 through a controlled user study, and EO3 using a user study on real attack transactions.

Full evaluation details are provided in Appendix E.

4.2 Evaluation Details and Results

4.2.1 EO1: Explanation Quality

We assess transaction summaries using 4 binary (pass/fail) criteria: **T1** (outcome correctness) and **T2** (clarity & conciseness) at the transaction level, and **F1** (faithfulness) and **F2** (intent quality) at the token-flow level; the full rubric is provided in Table 14 in Appendix.

We use two evaluation methods: (1) **LLM-as-judge:** GPT-5.1 (OpenAI, 2025) scores all system outputs with binary decisions for each metric, and we explicitly validate the judge against expert labels; (2) **Expert evaluation:** 3 DeFi Ph.D. researchers rate all transactions, assessing factual correctness, intent alignment, and risk awareness. The resulting pass rates from both evaluation sources are reported in Table 1.

For binary judgments, we additionally report ROUGE-L and BLEU for transaction-level summaries, Exact Match (EM) for flow-level semantic attribute generation, and Accuracy / Macro-F1 for the 14-way action_type classification subtask. As shown in Appendix E.1.3, *MATEX* achieves the best performance on both summary-generation metrics (ROUGE-L, BLEU) and structured prediction metrics (EM, Accuracy, Macro-F1), consistent with the pass-rate results in Table 1.

Judge Reliability and Significance. We validate the GPT-5.1 judge against expert annotations and find strong agreement overall, with per-metric Cohen’s κ and verbosity-bias checks reported in Appendix E.1.4. An additional Llama-3-70B judge yields the same ranking on F1/F2, indicating that the main conclusion is not specific to GPT-5.1. Expert inter-rater reliability is also high (Fleiss’ $\kappa = 0.76$), supporting the reliability of the human evaluation protocol.

To assess robustness, we compare *MATEX* against the strongest baseline using paired significance tests on the same 187 transactions: McNemar tests for binary pass/fail metrics and paired tests for ROUGE-L, EM, and action_type Macro-F1. The results confirm statistically significant gains on the key micro-level factuality and intent metrics.

Results. Results from both LLM-as-judge and expert human evaluation are summarized in Table 1. *MATEX* achieves the highest scores on F1 and F2 across both evaluators, demonstrating its strength in grounded, step-wise reasoning. On T1, *MATEX*

Table 1: EO1 pass rates (%): LLM-as-judge and expert human evaluation on baselines and *MATEX*.

Method	Pass Rate (% , LLM-as-judge)				Pass Rate (% , Human)			
	T1	T2	F1	F2	T1	T2	F1	F2
LLM	79.1	80.7	51.2	64.3	71.2	82.1	53.4	65.7
Few-Shot CoT	74.9	86.1	62.5	63.9	73.9	85.2	62.1	64.8
VLM	77.0	82.9	60.8	61.2	75.8	86.0	61.4	62.8
RAG	78.1	81.8	52.0	64.6	72.1	82.0	53.9	65.8
Single-Agent	66.8	89.8	67.6	67.5	70.1	88.1	69.0	69.2
MATEX	73.3	93.0	73.2	76.6	76.0	94.0	75.1	78.3

performs on par with the best baselines—scoring 76.0% in human evaluation, the highest. Manual inspection shows that *MATEX* sometimes omits minor fees or implicit approvals absent from the call trace, consistent with its rule to report only facts verified on-chain.

The standard NLP metrics reinforce the same pattern: *MATEX* achieves the best ROUGE-L, BLEU, Exact Match, Accuracy, and Macro-F1 under matched retrieval conditions. This indicates that the gains are not confined to judge-based or human pass/fail assessment, but also transfer to standard generation and classification metrics. Cross-backbone experiments with Llama-3-70B and Llama-3-8B show the same pattern: absolute scores fall with smaller open-weight models, but *MATEX* consistently improves over the corresponding Single-Agent baseline on F1 and F2.

The performance patterns align with the architectural limitations of each baseline. Both the Monolithic LLM and the VLM (augmented with a token-flow graph) process the input in a single pass without explicit decomposition or verification, making them prone to hallucination or missed details—especially on minor but critical flows such as small fees. Few-shot CoT improves local reasoning structure but still lacks grounded verification.

The Pure RAG benefits from external evidence, yet its fixed retrieval-and-append workflow cannot selectively investigate ambiguous flows or audit generated claims against the raw trace. The Single-Agent integrates decomposition, retrieval, synthesis, and narration into one cognitive loop, lacking role specialization or adversarial validation, which leads to error propagation in complex transactions.

Efficiency and Cost. *MATEX* incurs higher latency than the Monolithic LLM and RAG baselines because it performs iterative reasoning and auditing, but it remains comparable to Single-Agent. On complex transactions, *MATEX* reduces average la-

tency from 51.2 to 38.9 seconds and retrieval calls from 5.8 to 3.2 relative to Single-Agent; full latency, token, call, and estimated-cost profiles are reported in Table 11 in Appendix.

4.2.2 EO2: User Trust & Comprehension

We conduct a separate controlled user study with a new cohort of 20 active DeFi users, distinct from the 16 interview participants (full details are provided in Appendix E.2). Participants were recruited via Web3 communities and professional networks, and all reported at least three on-chain DeFi transactions in the past six months. The cohort spans six countries/regions and includes 4 practitioners, 5 researchers/developers, 7 students, and 4 casual users. All interfaces, explanations, and questionnaires were presented in English, so participants were required to be able to read English DeFi interfaces and transaction explanations.

EO2 uses a within-subject repeated-measures design where each participant views 9 transactions involving 113 token transfers. For each level (simple, medium, and complex), participants see one transaction under each of the following conditions using the EigenPhi interface: (a) Raw: the default explorer view only; (b) Single-Agent: explorer view augmented with annotations from the strongest baseline in EO1; (c) *MATEX*: explorer view augmented with *MATEX* explanations. Transaction-condition assignment is balanced across participants using a Latin Square design.

Results. Table 2 summarizes the EO2 results. *MATEX* consistently improves both objective comprehension and subjective trust, with the largest gains on complex transactions. In particular, comprehension accuracy on complex transactions rises from 35.3% in the Raw condition and 52.9% in the Single-Agent condition to 76.5% with *MATEX*. Perceived trust and clarity show a similar trend, increasing from 1.8–2.6 in the comparison conditions to 3.7–4.5 with *MATEX*.

The improvements suggest *MATEX* not only improves transaction understanding, but also supports safer signing decisions. The advantage of *MATEX* is especially pronounced on medium and high complexity transactions, where users benefit most from grounded, step-wise explanations. Participant-level bootstrap intervals and effect sizes confirm that the complex-transaction gains remain large.

Table 2: EO2 results ($N = 20$): Comprehension accuracy (%) and perceived trust/clarity (1–5) across transaction complexity (S=Simple, M=Medium, C=Complex).

Cond.	Accuracy (%)			Trust and Clarity		
	S	M	C	S	M	C
Raw	88.2	64.7	35.3	3.2	2.5	1.8
Agent	94.1	76.5	52.9	3.8	3.2	2.6
MATEX	98.0	88.2	76.5	4.5	4.1	3.7

4.2.3 EO3: Risk Awareness

We evaluate whether *MATEX* helps users recognize security threats in real transactions. We select 5 malicious transactions from real-world DeFi exploits (detailed in Appendix E.3) and 5 benign transactions of comparable complexity as controls.

We conduct a between-subjects study using the same 20-participant evaluation cohort as EO2, randomly split into two groups of 10. The first group views all 10 transactions in a standard block explorer (Raw). The second group uses the same interface augmented with *MATEX* explanations.

For each transaction, participants report their signing intention (Yes/No) and describe any perceived risks in an open-ended response. We measure the rejection rate on malicious transactions, the proportion of users who correctly identify the core threat, and the false-rejection rate on benign transactions. Risk recognition is evaluated by two independent researchers using manual semantic coding guided by core threat concepts.

Results. *MATEX* substantially improves users’ ability to identify security risks without increasing false alarms. On malicious transactions, the rejection rate increases from 36% (Raw) to 88% (*MATEX*), and the rate of correctly identifying the underlying threat rises from 28% to 72%. On benign transactions, the false-rejection rate remains low and stable: 12% in the Raw condition versus 8% in the *MATEX* condition.

Qualitative analysis of open-ended responses confirms this shift: users with *MATEX* provide

specific, actionable risk descriptions (e.g., “unlimited approval granted to 0x...789”), while Raw users often offer vague judgments (e.g., “looks sketchy”) or miss threats entirely. This demonstrates that *MATEX* enables non-experts to make more informed, security-aware signing decisions.

Using Fisher’s exact tests, the increase in malicious-transaction rejection is statistically significant ($p < 0.001$), and the increase in correct threat identification is also significant ($p = 0.002$). In contrast, the difference in false-rejection rates on benign transactions is not significant ($p = 0.741$). Bootstrap intervals, odds ratios, and leave-one-participant-out and leave-one-transaction-out checks show that no single participant or transaction drives these conclusions.

4.3 Ablation Study

We then explore which components are responsible for these gains. We perform ablations to assess the contribution of each major *MATEX* component by disabling one component at a time while keeping the rest of the pipeline unchanged, using the same inputs and tool access as the full system. Detailed results are reported in Appendix E.3.1.

The ablation results show that all four components contribute meaningfully, but in different ways. Removing the Profiler causes a modest drop in flow-level factuality (F1: 73.2→70.5), suggesting that uncertainty control mainly helps suppress local hallucinations on ambiguous flows. Removing the Investigator leads to a much larger degradation (F1: 73.2→62.8; action_type Macro-F1: 68.5→60.4), confirming that targeted protocol retrieval is essential for resolving unseen contracts and protocol-specific mechanisms.

Without the Synthesizer, both explanation quality and summary coherence collapse (T2: 93.0→64.2; ROUGE-L: 52.6→41.8), showing that evidence fusion is necessary to turn step-wise observations into coherent user-facing narratives. Finally, removing the Safety Auditor leaves macro-level outcome correctness relatively stable (T1: 73.3→72.2) but sharply reduces micro-level faithfulness (F1: 73.2→62.5), indicating that adversarial trace-based checking is especially important for preserving intermediate factual completeness rather than only the bottom-line outcome.

5 Related Work

LLM-Driven Transaction Analysis and Auditing. LLMs have shown superior capability in multiple domains, including academic writing (Liu et al., 2025b; Zheng et al., 2025b), agentic reasoning (Shao et al., 2024; Liu et al., 2025a), and finance reasoning (Zhang et al., 2025; Chung et al., 2025). Recent benchmarks such as XFinBench (Zhang et al., 2025), polymonitor (Jia et al., 2026), and surveys (Chung et al., 2025) highlight growing interest in applying LLMs to complex financial reasoning (e.g., forecasting).

Early Ethereum analysis relied on static analysis and symbolic execution (Torres et al., 2018; Tsankov et al., 2018). Transaction-level systems such as TXSPECTOR replay historical transactions for attack detection and forensics (Zhang et al., 2020a). Learned representations such as BlockGPT encode EVM traces for anomaly detection, while TLMG4Eth converts numerical records into transaction sentences and combines them with graph structure for fraud detection (Gai et al., 2023; Sun et al., 2025). These methods expose execution structure for detection, not user-facing explanations of realized economic effects.

LLM-Based Smart Contract Auditing. Recent work leverages Large Language Models or agents for smart contract auditing (Li et al., 2025; Ma et al., 2025), vulnerability detection (Wei et al., 2025), and gas optimization (Zheng et al., 2025a). Tx2TXT instead generates security-centric natural-language descriptions from smart-contract bytecode (Pan et al., 2023). Other work applies LLMs to Bitcoin transaction graphs (Lei et al., 2025), which differs from explaining the runtime economic behavior of general DeFi transactions at both transaction and token-flow levels for end users. Attack-centered systems diagnose incidents from traces and contracts (Wang et al., 2025), synthesize executable proofs of concept from attack transactions (Su et al., 2026), or reconstruct evidence-backed postmortems from seed transactions (Wang et al., 2026); these support analyst-facing forensics, not user-facing explanations of benign and malicious DeFi transactions.

Human-Centric Web3 Security and Interfaces. Prior user studies reveal security-relevant misconceptions that wallets and exchanges fail to correct, as well as recurring usability problems in mobile cryptocurrency wallets (Mai et al., 2020; Voskobojnikov et al., 2021). Blind-message attacks further

show that opaque Web3 authentication prompts can induce unsafe signatures, and that targeted wallet warnings can mitigate this risk (Yan et al., 2024).

Clear-signing standards such as ERC-7730 attach formatting metadata and contextual intent descriptions to structured signing data (Castillo et al., 2024), but depend on curated descriptors and target data at signing time. Tools such as PrettSmart and PonziLens+ visualize simulated contract behavior and bytecode actions to support analysis (Wen et al., 2025b,a), but target contract understanding or Ponzi identification.

Recent LLM-based approaches attempt to bridge this gap through transaction-level intent mining (Mao et al., 2025). In particular, the concurrent work of Mao et al. (2025) is closely related to ours, but it addresses a different problem formulation. Their Transaction Intent Mining framework models DeFi understanding primarily as a macro-level intent mining task over a predefined intent taxonomy, with the goal of assigning one or more coarse transaction-level intent labels. By contrast, *TxSum* is a user-centered task grounded in our interviews: it pairs a transaction-level natural-language explanation with micro-level token-flow parsing over five attributes, yielding finer-grained outputs. This difference is important in complex DeFi settings: a system may infer a coarse label such as swap or arbitrage while still failing to explain intermediate fees, approvals, routing, or other risk-bearing state changes relevant to blind-signing decisions.

6 Conclusion

Through formative interviews, we find that safe transaction understanding in DeFi requires explanations that are structured, risk-aware, and grounded at the token-flow level. Motivated by this, we formulate *TxSum*, a domain-grounded task and dataset that pairs transaction-level explanations with micro-level token-flow parsing.

We then introduce *MATEX*, a grounded multi-agent framework that combines uncertainty retrieval, evidence synthesis, and adversarial factual auditing to generate explanations of complex DeFi transactions. Across automatic evaluation, expert assessment, and user studies, *MATEX* consistently outperforms baselines, with especially strong gains on complex multi-protocol transactions and real-world risk recognition. We hope this work provides a useful foundation for future research on high-stakes transaction understanding.

Limitations

Meanwhile, several limitations remain. The *TxSum* dataset covers 187 Ethereum transactions and 2,375 token-flow annotations from 2025 and focuses on single-chain Ethereum mainnet. Our expert annotation protocol operates at the token-flow level and is therefore costly, which limits dataset scale despite the larger number of flow-level observations. The current schema and *MATEX* implementation target EVM-style execution traces; adaptation to EVM-compatible chains should be limited, but non-EVM systems such as Solana and cross-chain executions require additional schema and tooling changes. The current inputs also do not cover payment-channel activity performed off-chain; protocols such as Fair-Relay incur zero on-chain cost in optimistic executions and invoke on-chain dispute handling only when needed (Xue et al., 2026).

While our user study includes participants with diverse backgrounds, it captures signing behavior in a controlled setting rather than real-world high-pressure scenarios. The evaluation also uses a limited set of model families; domain-specific safety benchmarks show that failure behavior can vary across modalities and architectures (Peng et al., 2026), motivating broader backbone evaluation in future work.

Ethical Considerations

This study was reviewed and approved by the Institutional Review Board (IRB) of our institution. All interview participants provided informed consent prior to participation and were informed of their right to withdraw at any time. To protect privacy, all collected data (including interview transcripts, demographic information, and transactions) were anonymized and stored on encrypted, access-controlled servers. No personally identifiable information is included in the published dataset or supplementary materials. *TxSum* contains only public on-chain transactions; no private user data or wallet secrets were collected or exposed.

We also note a dual-use concern. A system that helps users or analysts understand suspicious transactions more clearly could, in principle, also help attackers test whether malicious behavior is less obvious under current interfaces. However, our primary goal is defensive: improving user awareness, supporting security analysis, and reducing blind signing in high-risk DeFi environments. We therefore present *MATEX* as an assistive explana-

tion tool rather than an automated execution or attack-planning system.

Acknowledgments

This work was supported by the National Key Research and Development Program of China (No. 2025YFB3110200), the Guangdong Basic and Applied Basic Research Foundation (No. 2026A1515030046), and the State Key Laboratory of Internet Architecture, Tsinghua University (No. HLW2025ZD14). Xuechao Wang was supported by the Guangdong Basic and Applied Basic Research Foundation (No. 2025A1515110040), the Guangzhou-HKUST(GZ) Joint Funding Program (No. 2025A03J3882), the Guangzhou Municipal Science and Technology Project (No. 2025A04J4168), and a gift from Stellar Development Foundation.

References

- BlockSec. 2025a. [Bybit \\$1.5B Hack: In-Depth Analysis of the Malicious Safe Wallet Upgrade Attack](#).
- BlockSec. 2025b. [MetaSuites, The Swiss Army Knife for Builders](#).
- Grady Booch, Francesco Fabiano, Lior Horesh, Kieran Kate, Jonathan Lenchner, Nick Linck, Andreas Loreggia, Keerthiram Murgesan, Nicholas Mattei, Francesca Rossi, and Biplav Srivastava. 2021. Thinking Fast and Slow in AI. In *AAAI Conference on Artificial Intelligence (AAAI)*, pages 15042–15046.
- Richard E Boyatzis. 1998. *Transforming Qualitative Information: Thematic Analysis and Code Development*. Sage.
- Laurent Castillo, Derek Rein, Pierre Aoun, Arik Galansky, Bartosz Rozwarski, Kaan Uzdogan, Fredrik, and Alex Forshtat. 2024. [ERC-7730: Structured Data Clear Signing Format](#). Ethereum Request for Comments 7730.
- Yoonseo Chung, Jeonghyun Kim, MiYeon Kim, Min-suh Joo, and Hyunsoo Cho. 2025. *Foundations of LLMs and Financial Applications*. Springer Nature Singapore.
- DeFiLlama. 2026. [Protocol Categories: DeFi TVL and Revenue Sectors](#).
- EigenPhi. 2025. [EigenPhi](#).
- Yu Gai, Liyi Zhou, Kaihua Qin, Dawn Song, and Arthur Gervais. 2023. Blockchain Large Language Models. *Cryptology ePrint Archive, Report 2023/592*.
- Huaiyu Jia, Luofeng Zhou, Wentao Zhang, Lin William Cong, Siguang Li, and Shuo Sun. 2026. Unlocking

- the Forecasting Economy: A Suite of Datasets for the Full Lifecycle of Prediction Market: [Experiments & Analysis]. *CoRR abs/2604.20421*.
- Philippe Laban, Hiroaki Hayashi, Yingbo Zhou, and Jennifer Neville. 2026. LLMs Get Lost in Multi-Turn Conversation. In *International Conference on Learning Representations (ICLR)*.
- LangChain. 2025. [LangGraph](#).
- Yuchen Lei, Yuexin Xiang, Qin Wang, Rafael Dowsley, Tsz Hon Yuen, Kim-Kwang Raymond Choo, and Jiangshan Yu. 2025. Large Language Models for Cryptocurrency Transaction Analysis: A Bitcoin Case Study. *CoRR abs/2501.18158*.
- Zongwei Li, Xiaoqi Li, Wenkai Li, and Xin Wang. 2025. SCALM: Detecting Bad Practices in Smart Contracts Through LLMs. In *AAAI Conference on Artificial Intelligence (AAAI)*, pages 470–477.
- Mingyi Liu, Jun Ho Huh, HyungSeok Han, Jaehyuk Lee, Jihae Ahn, Frank Li, Hyoungshick Kim, and Taesoo Kim. 2024. I Experienced More than 10 DeFi Scams: On DeFi Users’ Perception of Security Breaches and Countermeasures. In *USENIX Security Symposium (USENIX Security)*, pages 6039–6055. USENIX.
- Yule Liu, Heyi Zhang, Jinyi Zheng, Zhen Sun, Zifan Peng, Jiaheng Wei, Tianshuo Cong, Yilong Yang, and Xinlei He. 2025a. Auditing Data Membership in Reinforcement Learning with Verifiable Rewards. *CoRR abs/2511.14045*.
- Yule Liu, Zhiyuan Zhong, Yifan Liao, Zhen Sun, Jingyi Zheng, Jiaheng Wei, Qingyuan Gong, Fenghua Tong, Yang Chen, Yang Zhang, and Xinlei He. 2025b. On the Generalization and Adaptation Ability of Machine-Generated Text Detectors in Academic Writing. In *ACM Conference on Knowledge Discovery and Data Mining (KDD)*, page 5674–5685.
- Zeren Luo, Zifan Peng, Yule Liu, Zhen Sun, Mingchen Li, Jingyi Zheng, and Xinlei He. 2025. Unsafe LLM-Based Search: Quantitative Analysis and Mitigation of Safety Risks in AI Web Search. In *USENIX Security Symposium (USENIX Security)*, pages 8055–8074. USENIX.
- Wei Ma, Daoyuan Wu, Yuqiang Sun, Tianwen Wang, Shangqing Liu, Jian Zhang, Yue Xue, and Yang Liu. 2025. Combining Fine-Tuning and LLM-Based Agents for Intuitive Smart Contract Auditing with Justifications. In *IEEE/ACM International Conference on Software Engineering (ICSE)*, pages 1742–1754. IEEE.
- Alexandra Mai, Katharina Pfeffer, Matthias Gusenbauer, Edgar Weippl, and Katharina Krombholz. 2020. User Mental Models of Cryptocurrency Systems - A Grounded Theory Approach. In *USENIX Symposium on Usable Privacy and Security (SOUPS)*, pages 341–358. USENIX.
- Qian’ang Mao, Yuxuan Zhang, Jiaman Chen, Wenjun Zhou, and Jiaqi Yan. 2025. Know Your Intent: An Autonomous Multi-Perspective LLM Agent Framework for DeFi User Transaction Intent Mining. *CoRR abs/2511.15456*.
- OpenAI. 2025. [GPT-5.1: A Smarter, More Conversational ChatGPT](#).
- Yu Pan, Zhichao Xu, Levi Taiji Li, Yunhe Yang, and Mu Zhang. 2023. Automated Generation of Security-Centric Descriptions for Smart Contract Bytecode. In *ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA)*, pages 1244–1256. ACM.
- Zifan Peng, Mingchen Li, Yue Wang, and George T. S. Ho. 2023. Combating the COVID-19 Infodemic Using Prompt-Based Curriculum Learning. *Expert Systems with Applications*.
- Zifan Peng, Mingchen Li, Yue Wang, and Daniel Y. Mo. 2025a. Prompt-Based Contrastive Learning to Combat the COVID-19 Infodemic. *Machine Learning*.
- Zifan Peng, Yule Liu, Zhen Sun, Mingchen Li, Zeren Luo, Jingyi Zheng, Wenhan Dong, Xinlei He, Xuechao Wang, Yingjie Xue, Shengmin Xu, and Xinyi Huang. 2026. JALMBench: Benchmarking Jailbreak Vulnerabilities in Audio Language Models. In *International Conference on Learning Representations (ICLR)*.
- Zifan Peng, Yingjie Xue, and Jingyu Liu. 2025b. Cross-Chain Options: A Bridgeless, Universal, and Efficient Approach. In *IEEE International Conference on Web Services (ICWS)*, pages 902–912. IEEE.
- Bob Price. 2002. Laddered Questions and Qualitative Data Research Interviews. *Journal of Advanced Nursing*.
- Qwen3-VL Technical Report. 2025. Qwen3-VL Technical Report. *CoRR abs/2511.21631*.
- Kaushal Shah, Dhruvil Lathiya, Naimish Lukhi, Keyur Parmar, and Harshal Sanghvi. 2023. A Systematic Review of Decentralized Finance Protocols. *International Journal of Intelligent Networks*.
- Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, and Daya Guo. 2024. DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models. *CoRR abs/2402.03300*.
- Xing Su, Hao Wu, Hanzhong Liang, Yunlin Jiang, Yuxi Cheng, Yating Liu, and Fengyuan Xu. 2026. From Transactions to Exploits: Automated PoC Synthesis for Real-World DeFi Attacks. *CoRR abs/2601.16681*.
- Jianguo Sun, Yifan Jia, Yanbin Wang, Ye Tian, and Sheng Zhang. 2025. Ethereum Fraud Detection via Joint Transaction Language Model and Graph Representation Learning. *Information Fusion*.

- Qwen Team. 2025. Qwen3 Technical Report. *CoRR abs/2505.09388*.
- Tenderly. 2025. [Tenderly](#).
- Christof Ferreira Torres, Julian Schütte, and Radu State. 2018. Osiris: Hunting for Integer Bugs in Ethereum Smart Contracts. In *Annual Computer Security Applications Conference (ACSAC)*, pages 664–676. ACM.
- Petar Tsankov, Andrei Dan, Dana Drachler-Cohen, Arthur Gervais, Florian Buenzli, and Martin Vechev. 2018. Securify: Practical Security Analysis of Smart Contracts. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 67–82.
- Artemij Voskobojnikov, Oliver Wiese, Masoud Mehrabi Koushki, Volker Roth, and Konstantin Beznosov. 2021. The U in Crypto Stands for Usable: An Empirical Study of User Experience with Mobile Cryptocurrency Wallets. In *Annual ACM Conference on Human Factors in Computing Systems (CHI)*. ACM.
- Shuzheng Wang, Yue Huang, Zhuoer Xu, Yuming Huang, and Jing Tang. 2025. TraceLLM: Security Diagnosis Through Traces and Smart Contracts in Ethereum. *CoRR abs/2509.03037*.
- Ziyue Wang, Jiangshan Yu, Kaihua Qin, Dawn Song, Arthur Gervais, and Liyi Zhou. 2026. TxRay: Agentic Postmortem of Live Blockchain Attacks. *CoRR abs/2602.01317*.
- Zhiyuan Wei, Jing Sun, Yuqiang Sun, Ye Liu, Daoyuan Wu, Zijian Zhang, Xianhao Zhang, Meng Li, Yang Liu, Chunmiao Li, Mingchao Wan, Jin Dong, and Liehuang Zhu. 2025. Advanced Smart Contract Vulnerability Detection via LLM-Powered Multi-Agent Systems. *IEEE Transactions on Software Engineering*.
- Xiaolin Wen, Tai D. Nguyen, Shaolun Ruan, Qiaomu Shen, Jun Sun, Feida Zhu, and Yong Wang. 2025a. PonziLens+: Visualizing Bytecode Actions for Smart Ponzi Scheme Identification. *IEEE Transactions on Visualization and Computer Graphics*.
- Xiaolin Wen, Tai D. Nguyen, Lun Zhang, Jun Sun, and Yong Wang. 2025b. PrettySmart: Visual Interpretation of Smart Contracts via Simulation. *IEEE Transactions on Visualization and Computer Graphics*.
- Yingjie Xue, Jingyu Liu, Zifan Peng, Chao Lin, Jianan Hong, and Xinyi Huang. 2026. FairRelay: Fair Off-Chain Incentives for Decentralized Physical Infrastructure Networks. *IEEE Transactions on Dependable and Secure Computing*.
- Kailun Yan, Xiaokuan Zhang, and Wenrui Diao. 2024. Stealing Trust: Unraveling Blind Message Attacks in Web3 Authentication. In *ACM SIGSAC Conference on Computer and Communications Security (CCS)*, pages 555–569. ACM.
- Mengya Zhang, Xiaokuan Zhang, Yinqian Zhang, and Zhiqiang Lin. 2020a. TXSPECTOR: Uncovering Attacks in Ethereum from Transactions. In *USENIX Security Symposium (USENIX Security)*, pages 2775–2792. USENIX.
- Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020b. BERTScore: Evaluating Text Generation with BERT. In *International Conference on Learning Representations (ICLR)*.
- Zhihan Zhang, Yixin Cao, and Lizi Liao. 2025. XFin-Bench: Benchmarking LLMs in Complex Financial Problem Solving and Reasoning. In *Findings of the Association for Computational Linguistics (ACL)*, pages 8715–8758. ACL.
- Jingyi Zheng, Zifan Peng, Yule Liu, Junfeng Wang, Yifan Liao, Wenhan Dong, and Xinlei He. 2025a. GasAgent: A Multi-Agent Framework for Automated Gas Optimization in Smart Contracts. *CoRR abs/2507.15761*.
- Jingyi Zheng, Junfeng Wang, Zhen Sun, Wenhan Dong, Yule Liu, and Xinlei He. 2025b. TH-Bench: Evaluating Evading Attacks via Humanizing AI Text on Machine-Generated Text Detectors. In *ACM Conference on Knowledge Discovery and Data Mining (KDD)*, pages 5948–5959.

APPENDIX

A Interview Protocol Details

A.1 Interview Structure

Each 45–60 minute interview followed a three-phase semi-structured protocol, held via Zoom or in person, recorded with consent, and transcribed verbatim.

Phase 1: Background and Experience (5–10 mins). We began by establishing the participant’s DeFi experience to contextualize their responses. Standardized prompts included:

1. **Usage History:** “How long have you used DeFi? Which protocols do you interact with most frequently (e.g., Uniswap, Aave, Lido)?”
2. **Recent Transaction:** “Describe your most recent on-chain transaction. What were you trying to achieve, and how did you decide whether to sign it?”
3. **Tooling:** “What tools do you use to inspect transactions before signing (e.g., Etherscan, Rabby, MetaMask)? What do you find helpful or frustrating?”

Table 3: Transaction examples used in think-aloud exercises.

ID	Type	Description
TX1	Multi-hop Swap	User swaps 10 ETH for DAI via 1inch aggregator, which routes through Uniswap V3 (ETH→USDC) and SushiSwap (USDC→DAI), with fees collected by intermediary contracts.
TX2	Flash Loan Exploit	Attacker takes Aave flash loan to artificially inflate liquidity on a mock pool, enabling a large swap that drains user funds.
TX3	Revolving Leverage	User borrows USDC from Aave, swaps to ETH on CowSwap, stakes ETH on Lido, then uses stETH as collateral to borrow more USDC—creating cyclic leverage.
TX4	Stake + Approval	User stakes ETH on Lido and grants an unlimited approval to Lido’s stETH contract—standard but often misunderstood pattern.

This phase helps us tailor subsequent questions and identify domain-specific heuristics.

Phase 2: Think-Aloud Task (20–25 mins). Participants analyzed four Ethereum transactions (see Table 3) in a simplified explorer showing the sender, token flows, decoded calls, and net balance changes, but no raw bytecode.

For each transaction, participants were instructed: “Please think aloud as you try to understand this transaction. Explain in your own words: (1) what it does, (2) what risks you see, and (3) whether you would sign it—and why.” Interviewers used non-leading prompts to sustain reflection:

- “Can you walk me through how you reached that conclusion?”
- “What part of the trace supports that interpretation?”
- “Is there anything you’re unsure about?”

Phase 3: Laddered Questioning (15–20 mins). Building on Phase 2, we used laddering (Price, 2002) to surface mental models. Each statement (e.g., “I wouldn’t sign TX4”) prompted the following questions:

1. **Concrete Attribute:** “What specifically about TX4 made you uncomfortable?”
2. **Functional Consequence:** “What could go wrong if you did sign it?”
3. **Personal Value:** “Why is that outcome unacceptable to you? What does it mean for your sense of control or security?”

We repeated this chain until reaching a fundamental value (e.g., “autonomy,” “transparency,” “irreversibility”), then asked what explanations participants wanted:

- “What information would have made TX2 easier to understand?”
- “Should the explanation focus on individual steps or the overall goal?”
- “Would a clear explanation change your signing decision? In what cases?”

These responses directly informed the design of the *TxSum* schema and *MATEX* evaluation (RQ3).

A.2 Interview Participant Demographics

Table 4 summarizes the participants’ self-reported demographics.

Table 4: Demographic summary of interview participants. Role, Experience (years with Web3), and Country (primary residence during Web3 use) are self-reported.

ID	Role	Gender	Country	Experience
1	Trader	M	China (HK)	5
2	Analyst	F	Singapore	4
3	Analyst	M	China	6
4	Trader	M	USA	4
5	Developer	M	USA	5
6	Developer	M	China (HK)	4
7	Developer	M	UK	3
8	Unemployed	M	China	3
9	Designer	F	China	2
10	Student	M	USA	2
11	Student	M	China (HK)	4
12	Student	F	Australia	3
13	Accountant	M	Canada	1
14	Officer	M	China	2
15	Engineer	F	Singapore	3
16	Student	M	South Korea	2

A.3 Interview Analysis and Key Findings

All interviews were transcribed verbatim and analyzed using inductive thematic analysis (Boyatzis, 1998). Two researchers independently coded the

first 20% of transcripts to develop an initial codebook, then iteratively refined it through discussion while coding the remaining data. Inter-coder agreement on the shared subset reached Cohen’s $\kappa = 0.82$, and disagreements were resolved by consensus. Thematic saturation was reached by the 14th interview, with no new major themes emerging in the final two interviews.

Our formative interviews revealed three findings that directly motivated the formulation of *TxSum*.

Finding 1: Blind signing is widespread. Even experienced users often approve transactions they cannot fully interpret. Rather than understanding the complete transaction, they frequently rely on fragile heuristics such as “reject unlimited approvals” or “avoid unknown contracts,” which break down in compositional DeFi transactions spanning multiple contracts and hidden intermediate steps.

Finding 2: Current tools suffer from semantic poverty. Participants repeatedly reported that existing interfaces expose low-level artifacts such as token transfers, calldata, and decoded function names, but fail to explain what those events mean economically. As a result, users often cannot infer the actual purpose, risk, or preconditions of a transaction from syntactic traces alone.

Finding 3: Users reason at the token-flow level but still want concise summaries. Most participants consistently treated token flows as the natural unit of understanding, because each flow corresponds to a small economic action such as swapping, depositing, repaying, or fee deduction. At the same time, they also wanted a concise transaction-level summary that explains the overall intent and highlights critical risk-bearing effects such as approvals, ownership changes, or hidden fees.

B Identified Attributes

We define five semantic attributes for each token flow, grounded in user mental models and derived from our interview study.

- **action_type:** A protocol-agnostic label denoting the financial primitive of a token transfer (e.g., `swap_in`, `deposit`, `fee`). Classification is determined solely by on-chain signals: zero-address interactions, event logs (e.g., `Transfer`, `Mint`), and decoded function names. The taxonomy comprises 14 mutually exclusive types, designed to cover core DeFi operations while enabling cross-protocol consistency. Full definitions are provided in Table 5.

- **intent:** A user-centric statement of the economic purpose behind the step (e.g., “Redeem yield-bearing tokens to unlock liquidity”). This field bridges raw token movements and human-understandable goals, reflecting what the user is trying to achieve—not how the protocol implements it. Intent is inferred from the transaction context, protocol documentation, and the action’s role in the broader sequence.
- **mechanism:** A precise description of the on-chain execution pathway, formatted as “Call `<Contract>.<function>()`” or “`<Contract>#<function>()`”. It specifies which contract and function were invoked (e.g., “Call `PendleRouter.redeem()` to burn SY and withdraw `sUSDe`”), grounding the explanation in decoded calldata and event logs. This enables users to verify behavior against official documentation.
- **precondition:** The minimal set of on-chain conditions required for the action to succeed (e.g., “User holds sufficient SY balance and has approved the router”). Preconditions are derived from protocol documentation, function `require()` statements, and observable state (e.g., prior approvals, pool liquidity). Numerical thresholds are included only when explicitly visible in the trace or documentation; otherwise, the field uses qualitative descriptions.
- **result:** A factual account of the direct state changes caused by the action (e.g., “User receives 53.5 `sUSDe`; SY balance is burned in the `Pendle-SUSDESY` contract”). This field strictly reports what is observable in token transfers, event logs, or state diffs—never inferred profits, market impact, or psychological outcomes. It serves as the ground-truth anchor for the entire explanation.

All attributes are derived using only on-chain data and official protocol documentation. No external knowledge, speculative intent, or subjective interpretation is permitted. Detailed examples are provided in Appendix C.

C Annotation Guidelines, Consistency Rules, and Edge Cases

This section details our annotation protocol, including step-by-step guidelines, attribute definitions, and concrete examples. All annotators were required to have prior experience with DeFi protocols (e.g., Uniswap, Aave, Pendle) and familiarity with

Table 5: Action taxonomy used for structured DeFi token-flow annotation and analysis.

Action Type	Definition
mint	Creation of new tokens issued from the zero address to a recipient (i.e., newly minted assets).
burn	Permanent destruction of tokens by transferring them to the zero address.
redeem	Conversion of derivative or wrapped assets into their underlying assets (e.g., SY $\rightarrow$ sUSDe, LP $\rightarrow$ underlying legs, WETH $\rightarrow$ ETH), where the derivative token is returned to a contract and the underlying asset is released.
issue	Issuance of derivative or receipt assets from underlying assets (e.g., sUSDe $\rightarrow$ SY or single-asset deposits minting LP tokens).
swap_in	Input leg of a token exchange, representing assets sent from the user to a DEX, router, or liquidity pool as part of a swap.
swap_out	Output leg of a token exchange, representing assets sent from a DEX, router, or liquidity pool to the user as the result of a swap.
deposit	Transfer of assets into lending protocols, staking contracts, or vaults to obtain interest-bearing positions or receipt tokens, excluding swaps.
withdraw	Retrieval of assets from lending protocols, staking contracts, or vaults, potentially accompanied by redemption of receipt tokens.
borrow	Borrowing of assets from a lending protocol, resulting in an increase in the user's debt position.
repay	Repayment of borrowed assets to a lending protocol, including repayments made by liquidators on behalf of borrowers.
fee	Fees charged by protocols, routers, bridges, or flash-loan mechanisms within a transaction.
reward	Distribution of protocol incentives or yield (e.g., staking rewards, vault emissions, or airdrops) without a direct input counterpart.
route	Transfer used solely for execution routing or delegation between protocol-internal contracts (e.g., router, settler, adapter, executor), without constituting an independent economic action.
direct transfer	Direct token transfer between two externally owned accounts or non-protocol-related addresses, without triggering swaps, minting, burning, or other specific economic actions.

blockchain explorers (Etherscan, Tenderly, Phalcon). Annotation proceeds in three steps:

Step 1: Transaction Overview. Annotators first examine the transaction's high-level effects in block explorers—focusing on net token flows, called contracts, and state changes—to form a one-sentence coarse summary (e.g., “Extract yield from a Pendle YT position”). The summary guides the remaining labels but is excluded from the dataset.

Step 2: Token Flow Segmentation. Each row in the token transfer log is treated as a single *micro action* representing an atomic economic step.

Step 3: Attribute Labeling. For each micro action, annotators fill five fields using only on-chain evidence and official protocol documentation.

Annotation Consistency Rules. To ensure uniformity, annotators follow these conventions:

- Use “the user” for the transaction initiator, “the protocol” for contract addresses, and “burn address” for zero addresses.
- Report token amounts in native units (e.g., “53.5 sUSDe”), not USD equivalents.
- Avoid speculative intent (e.g., “to short the mar-

ket”); focus only on directly inferable purposes.

- Every intent and result must align with the corresponding token transfer or event log.

Edge Cases and Disambiguation Guidelines. Because real DeFi transactions are highly compositional, the main annotation difficulty is not identifying the final outcome, but disambiguating intermediate steps that are easy to overlook yet critical for faithful explanation and user safety. We therefore define explicit decision rules for common but consequential edge cases:

- **Protocol-internal routing vs. user-facing economic action.** Transfers among routers, adapters, settlers, vault helpers, or executor contracts are labeled as *route* when they merely relay assets for execution. They are labeled with a semantic action (e.g., *swap_out*, *deposit*, *repay*) only if the transfer itself realizes an economically meaningful state transition for the user. This distinction is crucial in aggregator-routed and cross-protocol transactions, where many intermediate hops are operationally necessary but not independently meaningful.

- **Material intermediate effects vs. ignorable plumbing.** Annotators do not collapse away intermediate transfers when they carry economic or security significance. In particular, protocol fees, relay fees, approval side effects, collateral movements, liquidation repayments, or transfers into newly introduced contracts must remain explicit in the annotation even if the final net outcome appears simple. This rule is designed to support blind-signings, where omitted intermediate effects can hide the risk of a transaction.
- **Approval and permission changes.** Pure approval events do not create token-flow labels unless accompanied by an actual token transfer, but they must be reflected in the transaction-level summary whenever they are economically or security relevant. Unlimited approvals, newly granted permissions to unfamiliar contracts, and approval-reset patterns may also be encoded in preconditions when they are necessary for interpreting subsequent token flows.
- **Wrapped, unwrapped, and derivative assets.** Conversions such as $\text{ETH} \leftrightarrow \text{WETH}$ or underlying asset $\leftrightarrow$ receipt/derivative token are annotated by their economically salient primitive rather than surface transfer form. For example, receiving stETH after supplying ETH is not treated as a generic transfer but as issuance of a derivative position; conversely, burning a receipt token to recover the underlying asset is labeled as redeem. This prevents semantically shallow descriptions that miss the user's actual financial position change.
- **Multi-hop aggregation and nested execution.** In aggregator- or meta-router-mediated swaps, each token transfer step is annotated separately at the flow level, including intermediate hops and fee deductions. However, the transaction-level summary may compress contiguous hops into a higher-level operation only when doing so does not obscure economically material fees, hidden approvals, routing side effects, or risk-bearing contract transitions. This preserves readability at the summary level while maintaining full micro-level accountability.
- **Borrow-swap-stake-reborrow loops.** In revolving leverage or recursive collateralization patterns, annotators preserve each borrow, swap, stake, mint, collateralization, and re-borrow step independently rather than collapsing the trace

into a single coarse label such as “leveraged position.” This is important because user-facing risk often lies in the repeated dependency structure itself rather than in the final macro intent.

- **Flash-loan-mediated actions.** Flash-loan transactions are annotated by separating the temporary borrowed leg, intermediate execution steps, fee repayment, and final settlement. Even when the flash loan is fully repaid within the same transaction, the temporary capital injection is retained explicitly because it often explains otherwise unintuitive trade size, liquidation behavior, or exploit mechanics.
- **Control-transfer and upgrade-related transactions.** Transactions that change ownership, implementation, proxy mastercopy, operator privilege, or administrative authority are treated as high-priority cases even when token movement is minimal. In such cases, the summary must explicitly surface the control-transfer risk rather than only describing low-level call mechanics, because these transactions are especially relevant to phishing, wallet compromise, and governance attacks.
- **Direct transfer vs. protocol interaction.** A token movement is labeled as direct transfer only when it does not participate in a broader protocol-mediated economic action. If the transfer is part of a swap, staking flow, repayment path, settlement sequence, or exploit chain, it is labeled according to that semantic role rather than treated as an isolated transfer.
- **Unknown or partially verified contracts.** When verified source code or documentation is unavailable, annotators are instructed to remain conservative: they label only what is directly supported by token movements, calldata, event logs, and observable state changes. This rule prevents speculative intent attribution and aligns the dataset with the paper's broader emphasis on factual grounding under uncertainty.

Example Annotation. We provide an annotated token flow from a cross-DEX swap:

```
{
  "summary": "tx-level summary",
  "token_flow": [
    {
      "action_id": 1,
      "action_type": "swap_out",
      "intent": "Swap out ...",

```

```

    "mechanism": "ContractA#fallback()",
    "precondition": "ETH balance ...",
    "result": "... receives 0.1 ETH."
  },
  {...}, {...}
]
}

```

The final natural-language summary synthesizes all flows into a coherent narrative, concrete examples are provided in Appendix F:

The user executed a cross-DEX swap of 0.1 ETH for 37.1 USDC. To enable the trade, they first granted a one-time approval of 0.1 ETH to the RangoDiamond router. The router then executed the swap via a multi-step aggregation path on underlying AMMs, deducting a small relay fee and a protocol fee from the output. Unused ETH returned to the user.

C.1 Semantic Consistency of Annotations

For the natural language token-flow attributes—intent, mechanism, precondition, and result—we do not report κ as the primary consistency measure, because semantically equivalent annotations may differ lexically. Instead, we perform a supplementary semantic consistency analysis on the independently written annotation pairs for each token flow. Concretely, for each field, we compute BERTScore F1 between the two independently written annotations, and then average scores across all annotated token flows. We use the official BERTScore (Zhang et al., 2020b) implementation with a roberta-large backbone and baseline rescaling enabled.

The resulting field-level semantic consistency scores are all above 0.91, as shown in Table 6. These results indicate that, although annotators often use different surface forms, their free-form annotations remain consistent at the semantic level.

Table 6: Semantic consistency of free-form token-flow annotations, computed as average pairwise BERTScore F1 between independently written annotation pairs.

Field	BERTScore F1
intent	0.962
mechanism	0.913
precondition	0.982
result	0.948

C.2 Dataset Profile

This section summarizes the coverage profile of the 187 annotated transactions retained after filtering. The retained transactions are intentionally skewed toward semantically rich DeFi interactions rather than routine wallet activity. In particular, the dataset contains multi-step swaps, lending loops, staking operations, liquidity management, and cross-protocol compositions that require reasoning across multiple token flows and contracts. These scenarios correspond to prominent DeFi sectors in DeFiLlama’s ecosystem taxonomy and in a systematic survey of DeFi protocol forms (DeFiLlama, 2026; Shah et al., 2023). Table 7 reports non-exclusive scenario coverage, and Table 8 summarizes token-flow complexity, protocol diversity, and contract verification status.

For example, a transaction that borrows assets, swaps them through an aggregator, and then deposits the resulting tokens into a staking protocol contributes to the lending/borrowing, swap, and staking coverage profiles. Because these coverage categories can overlap, their counts need not sum to 187; in particular, 57% of retained transactions involve more than one protocol-level action pattern.

Table 7: Non-exclusive scenario coverage of the 187 annotated transactions in *TxSum* after filtering. Transactions may contribute to more than one category.

Category	Count
Swaps / Aggregator-routed swaps	47
Lending / Borrowing cycles	37
Staking / Delegation	42
Liquidity Pool operations	34
Cross-protocol interactions	29

Table 8: Coverage statistics of *TxSum*.

Coverage Statistic	Value
Total annotated TxS	187
Total token-flow annotations	2,375
Average token-flow steps / Tx	12.7
Average distinct protocols / transaction	2.8
All involved contracts without verified ABI	20.2%
Verified focal contracts	94
Partially verified focal contracts	56
Unverified / closed-source focal contracts	37

Algorithm 1 Dynamic Knowledge Injection

Require: Token flows $T = \{t_1, t_2, \dots, t_n\}$, uncertainty threshold τ , evidence budget k

Ensure: Synthesized context C_{syn}

```

1:  $C_{\text{syn}} \leftarrow \emptyset, B \leftarrow \emptyset$   $\triangleright B$ : evidence cache
2: for all  $t_i \in T$  do
3:    $(u_i, q_i) \leftarrow \text{Profiler}(t_i)$   $\triangleright q_i$ : retrieval key
4:   if  $u_i \leq \tau$  then
5:      $C_{\text{syn}} \leftarrow C_{\text{syn}} \cup \text{ToKnownPattern}(t_i)$ 
6:   else
7:     if  $q_i \notin B$  then
8:        $R_i \leftarrow \text{Investigator}(q_i)$ 
9:        $B[q_i] \leftarrow \text{SelectTopK}(R_i, k)$ 
10:    end if
11:     $C_{\text{syn}} \leftarrow C_{\text{syn}} \cup \text{Format}(t_i, B[q_i])$ 
12:  end if
13: end for
14: return  $C_{\text{syn}}$ 

```

D Agent Design Details

MATEX consists of four specialized agents that collaborate through a shared structured workspace (the “Evidence Board”) and interact through the two core mechanisms described in Appendix D.1. First, we provide the pseudocode for the two core mechanisms used in *MATEX*: Dynamic Knowledge Injection Protocol (DKIP, Algorithm 1) and Auditor-Guided Context Pruning (AGCP, Algorithm 2). Then, we describe each agent’s role, inputs, outputs, and operational behavior.

Table 9 summarizes the inputs, outputs, and pipeline roles of all four agents.

D.1 Algorithmic Details of *MATEX*

Profiler. The Profiler implements the fast, pattern-based stage of *MATEX* (System 1). It ingests the transaction metadata, call trace, token transfers, and balance changes, and produces two kinds of outputs for each token flow: (1) a lightweight semantic hypothesis when the flow matches a known pattern, and (2) an uncertainty assessment used by DKIP. Operationally, the Profiler assigns each flow a semantic uncertainty score u_i and, when escalation is needed, emits a retrieval key q_i that specifies what external protocol evidence should be queried (e.g., a contract-function pair, router name, or unknown address role).

Critically, the Profiler is designed to remain conservative under uncertainty. When it encounters unfamiliar contracts, custom execution paths, or

anomalous state changes, it does not force a semantic interpretation; instead, it raises the uncertainty score and forwards the relevant retrieval key to the Investigator. Ambiguous flows therefore trigger retrieval rather than forced interpretation.

Profiler Prompt

You are an expert Ethereum Transaction Profiler. Your goal is to create a structured investigation plan grounded in [TOKEN TRANSFERS].

CRITICAL: Anchor your analysis to the input [TOKEN TRANSFERS].

- Prefer using TT# ranges (e.g., "TT#1–TT#6") over generic "Step 1-5".
- Each segment should cover a contiguous range of TT# entries.
- For each uncertain flow or segment, assign an uncertainty score and provide a retrieval key for downstream investigation.

Output a JSON object with exactly these four keys:

1. "segments": logical phases of the transaction;
2. "initial_hypothesis": a high-level intuition about the transaction type;
3. "uncertain_flows": a list of uncertain TT# items with uncertainty scores and retrieval keys;
4. "investigation_tickets": concrete verification questions for the Investigator.

JSON FORMAT:

```

{
  "segments": [{"phase_id": 1, "range": "TT#1–TT#5", "summary_label": "..."}],
  "initial_hypothesis": "...",
  "uncertain_flows": [{"transfer_ref": "TT#3", "uncertainty_score": 0.82, "retrieval_key": "..."}],
  "investigation_tickets": [{"target_step": "...", "contract": "...", "reason": "...", "question": "..."}]
}

```

Protocol Investigator. The Investigator is only activated for flows whose uncertainty score exceeds the DKIP threshold. Given a retrieval key q_i from the Profiler, it retrieves protocol evidence from the same external sources used in evaluation: verified contract ABIs, protocol documentation, and temporally aligned web evidence. Its role is not to broadly summarize the protocol, but to return only the evidence necessary to resolve the ambiguous flow under investigation. This selective retrieval

Table 9: Inputs and outputs of the four agents in *MATEX*.

Agent	Input	Output	Role in the pipeline
Profiler	Token transfers, call trace, meta-data, balance changes	Flow-level hypotheses, uncertainty score u_i , retrieval key q_i when needed	Performs fast pattern-based screening and decides whether each flow should remain on the lightweight path or be escalated for retrieval.
Investigator	Retrieval key q_i and supporting trace context	Verified protocol evidence (e.g., ABI entries, documentation snippets, contract-role resolution)	Retrieves protocol context only for uncertain flows under DKIP.
Synthesizer	On-chain evidence, profiler hypotheses, retrieved evidence	Draft transaction explanation and flow-grounded semantic outputs	Fuses evidence into a coherent transaction-level explanation and structured flow-level interpretation.
Safety Auditor	Draft explanation, raw trace, token transfers	APPROVED or critique with error spans / unsupported claims	Checks factual grounding against raw traces and triggers AGCP to prune unsupported revision context before re-synthesis.

Algorithm 2 Auditor-Guided Context Pruning

Require: Fixed evidence context C_{syn} , raw trace Tr , max iterations M

Ensure: Verified explanation E_{final}

```

1:  $D \leftarrow \emptyset$  ▷ Mutable revision context
2: for  $iter = 1$  to  $M$  do
3:    $E_{\text{draft}} \leftarrow \text{Synthesizer}(C_{\text{syn}} \cup D)$ 
4:    $(v, F_{\text{err}}) \leftarrow \text{Auditor}(E_{\text{draft}}, Tr)$ 
5:   if  $v = \text{APPROVED}$  then
6:     return  $E_{\text{draft}}$ 
7:   end if
8:    $P_{\text{mask}} \leftarrow \text{LocateErrorContext}(D, E_{\text{draft}}, F_{\text{err}})$ 
9:    $D \leftarrow (D \setminus P_{\text{mask}}) \cup \text{Critique}(F_{\text{err}})$ 
10: end for
11: return  $E_{\text{draft}}$ 

```

design limits context noise and keeps downstream synthesis focused on uncertain parts of the trace.

If no verified ABI or documentation is available, the Investigator returns the absence of verifiable protocol evidence rather than guessing. The downstream explanation then relies only on observable token transfers, events, balance changes, trace structure, and call direction; it lowers semantic specificity, marks uncertainty, and flags observed risks such as approvals or unlimited allowances.

Investigator Prompt

You are a Researcher. I will provide a question about a contract and raw search results. Answer the question based STRICTLY on the search results.

Verification rule:

- Only output "Verified: ..." if the search results explicitly mention the contract/function AND describe the behavior relevant to the question.
- Otherwise, output: "No verifiable info found."

Keep it concise.

Synthesizer. The Synthesizer serves as the main reasoning module in System 2. It fuses 3 inputs: (1) the raw on-chain trace, (2) the Profiler’s flow-level hypotheses, and (3) the Investigator’s retrieved evidence patches. It maps low-level operations to user-facing actions, producing a transaction summary while grounding each flow in the trace.

To preserve factual grounding, the Synthesizer operates over evidence that remains internally attributed to its source (e.g., raw trace facts, ABI evidence). The final user-facing explanation does not need to expose these provenance tags verbatim, but it is generated from this evidence-grounded internal state. Its output is then passed to the Safety Auditor for adversarial validation. The full Synthesizer prompt is included in the public repository linked in the Introduction.

Safety Auditor. The Safety Auditor acts as an adversarial checker over the Synthesizer’s draft. It checks the explanation against the trace, transfers, and retrieved evidence, catching unsupported claims, omitted effects, action errors, and risks the draft misses.

If contradictions are found, the Auditor does not merely append a critique for the next round. Instead, *MATEX* applies Auditor-Guided Context Pruning (AGCP): the system identifies the unsp-

ported or error-causing portion of the mutable revision context and removes it before re-synthesis. This design reduces repeated hallucinations across iterations and is especially important in long, compositional DeFi transactions where a single unsupported inference can pollute subsequent drafts.

Safety Auditor Prompt

You are a Senior DeFi Security Auditor. Critique the Draft JSON against the Raw Trace.

Hard checks (must fail if violated):

- 1) Hallucinations: tokens/amounts/addresses/protocols/functions not supported by trace.
 - 2) Wrong 'action_type' classification.
 - 3) Missing fees or side effects IF they appear in [TOKEN TRANSFERS].
 - 4) Format errors (JSON schema, intent/mechanism/precondition/result structure).
 - 5) [TOKEN TRANSFERS] 1:1 coverage:
 - Every TT# entry must appear EXACTLY ONCE as token_flows[i].transfer_ref.
 - token_flows length must equal the number of transfers in [TOKEN TRANSFERS].
 - Each token_flows[i].result must EXACTLY match its TT# facts (from/to/token/amount).
- If perfect, output exactly: "APPROVED".
If errors exist, output a concise list of fixes.

E Evaluation Details

E.1 EO1: Explanation Quality

We evaluate explanation quality using both automated and expert-based methods on our full *TxSum* test set (187 complex transactions). The detailed justification is in Table 10.

LLM-as-a-Judge. We employ GPT-5.1 as an automated evaluator, with separate judging protocols for token-flow and transaction levels.

For **transaction-level evaluation (T1, T2)**, the judge receives: (1) the full on-chain input (transaction metadata, decoded call trace, complete token transfer list, and net balance changes), and (2) the generated sentence's natural language summary. It then outputs binary judgments for outcome correctness and clarity.

For **token-flow evaluation (F1, F2)**, the judge receives: (1) the raw on-chain sub-trace corresponding to a single token flow (e.g., a transfer event and its surrounding context), and (2) the predicted semantic attributes for that flow (e.g., intent, mechanism). It then outputs binary (YES/NO) judgments

for faithfulness and intent quality. Both protocols use structured prompts that require the model to ground every decision in the provided evidence. Prompts are validated against expert annotations; the final validation numbers are reported in the main text and Appendix E.1.4.

Transaction Level Judge Prompt

You are a Financial Auditor. Evaluate the "Model Summary" against the "Expert Summary" for the entire transaction.

Task

Evaluate two specific dimensions based ONLY on the summary text.

1. T1: Outcome Correctness (Financial Precision)

- ****Definition****: Does the summary accurately state the ****Final Net Outcome**** (Profit/Loss/Balance Changes)?

- ****PASS (YES)****: Directions (+/-) and amounts match the Expert Summary.

- ****FAIL (NO)****: Wrong direction (gain vs loss), wrong final numbers, or missing significant fees mentioned by Expert.

2. T2: Clarity (Readability)

- ****Definition****: Is the summary concise and easy to understand for a non-expert?

- ****PASS (YES)****: Natural language, logical flow, concise (approx 2-3 sentences).

- ****FAIL (NO)****: Verbose, dumps raw JSON/Hex data, robotic phrasing, or overly long. Output Format (Strict JSON)

{ "reasoning": "Brief analysis...",

"scores": {

"T1_Outcome": "YES",

"T2_Clarity": "YES" } }

Token Flow Level Judge Prompt

You are a DeFi Semantic Expert. Compare the "Model Prediction Flow" against the "Expert Ground Truth Flow" for a specific step in a transaction.

Task

Evaluate the prediction on two specific dimensions.

1. F1: Faithfulness (Semantic Equivalence)

- ****Definition****: Does the model describe the ***same*** factual event? (Amounts, Tokens, Direction).

- ****PASS (YES)****: "User sends 1 ETH" == "Transfer 1 ETH from User". Numbers match

Table 10: Detailed binary scoring rubric for *TxSum*. All judgments must be grounded in the on-chain input.

Metric	PASS (YES)	FAIL (NO)
Token Flow Level		
F1: Faithfulness	All claims are directly verifiable from the on-chain trace. Correct token, amount, direction, and action type (e.g., "swaps 50 USDC for 0.03 ETH via Uniswap V3").	Any hallucination or factual error: wrong token/amount, fabricated action (e.g., claims "stakes 1 ETH" when trace only shows WETH wrap).
F2: Intent Quality	Explains economic purpose using trace evidence (e.g., "deposits USDC into Aave to borrow DAI"). Purpose must be inferable from on-chain steps.	Only describes mechanics without purpose (e.g., "transfers USDC to Aave") or speculates unsupported intent (e.g., "to invest in crypto").
Transaction Level		
T1: Outcome Correctness	Net balance changes, fees, and critical non-token state changes exactly match on-chain data. Includes all fees $\geq 0.1\%$ of principal (e.g., "Net: +240 USDC, -0.05 ETH, fee: 1.2 USDT").	Any error in the final state or omission of material fees (e.g., claims "gained 240 USDC" when net is -10 USDC after fees).
T2: Clarity & Conciseness	Natural-language narrative in 3–4 sentences. No raw addresses, hex, or internal names. Groups related actions (e.g., "multi-hop swap via 1inch, then staked in Curve").	Uses EOA/contract addresses (e.g., "to 0x7a25..."), list format ("Step 1: ... Step 2: ..."), or exceeds 3 sentences.

exactly.

- ****FAIL (NO)****: Wrong amounts, wrong tokens, wrong direction, or factual contradictions.

2. F2: Intent Quality (Economic Purpose)

- ****Definition****: Does the ‘intent’ capture the ****economic purpose**** ("Why"), not just the mechanical action ("What")?

- ****PASS (YES)****: "User deposits ETH ****to collateralize****." / "User sends funds ****to initiate swap****."

- ****FAIL (NO)****: "User transfers ETH." (Mechanical repetition).

Output Format (Strict JSON)

```
{ "reasoning": "Brief analysis...",
  "scores": {
    "F1_Faithfulness": "YES",
    "F2_Intent_Quality": "YES" }}
```

Expert Evaluation. Three DeFi researchers with ≥ 3 years of blockchain security or protocol development experience perform blind evaluations of all 187 transactions (balanced across systems). Experts assess each output on: (1) factual correctness (aligned with F1 and T1), (2) intent alignment (F2), and (3) risk awareness (whether critical risks are obscured or omitted). Each criterion is scored as Pass/Fail. Inter-annotator agreement is measured via Fleiss’ $\kappa = 0.76$.

E.1.1 Efficiency and Cost Profiling

We profile average and P95 end-to-end latency, LLM calls, external retrieval calls, token consumption, and estimated per-transaction cost across transaction complexity levels. Table 11 shows that *MATEX* is slower than single-pass Monolithic LLM and RAG baselines, as expected for an iterative audited system, but remains comparable to Single-Agent. On complex transactions, *MATEX* provides better explanations than Single-Agent while reducing average retrieval calls from 5.8 to 3.2 and latency from 51.2 to 38.9 seconds.

E.1.2 Cross-Backbone Robustness

The main experiments fix Qwen3-Max to isolate architectural differences across systems. To test whether the gains depend on that backbone, we repeat the comparison between *MATEX* and Single-Agent using Llama-3-70B and Llama-3-8B. As shown in Table 12, absolute performance declines with smaller open-weight models, but *MATEX* consistently improves F1 and F2 over the corresponding Single-Agent baseline.

E.1.3 Additional Metrics for EO1

Beyond binary pass/fail evaluation, we report Accuracy and Macro-F1 for the 14-way action_type task, ROUGE-L and BLEU for summaries, and Exact Match (EM) for flow attributes. The results

Table 11: End-to-end inference efficiency and estimated cost by transaction complexity.

Method	Complexity	Avg. latency (s)	P95 latency (s)	Avg. LLM calls	Avg. retrieval calls	Avg. tokens	Est. cost/tx
Monolithic LLM	Simple	5.8	9.7	1.0	0.0	8.4k	\$0.012
Monolithic LLM	Medium	8.9	14.6	1.0	0.0	14.7k	\$0.021
Monolithic LLM	Complex	14.8	23.5	1.0	0.0	24.3k	\$0.035
RAG	Simple	9.6	16.2	1.0	2.0	12.9k	\$0.019
RAG	Medium	15.8	26.7	1.0	4.0	21.8k	\$0.032
RAG	Complex	25.4	41.3	1.0	6.0	34.9k	\$0.052
Single-Agent	Simple	16.9	28.5	3.4	1.8	23.6k	\$0.041
Single-Agent	Medium	30.7	50.8	5.3	3.6	44.2k	\$0.078
Single-Agent	Complex	51.2	83.6	7.0	5.8	73.5k	\$0.131
MATEX	Simple	14.2	24.1	3.1	0.7	21.4k	\$0.037
MATEX	Medium	24.6	39.8	4.6	1.9	39.7k	\$0.070
MATEX	Complex	38.9	64.7	6.3	3.2	65.8k	\$0.116

Table 12: Cross-backbone robustness. Gains are absolute percentage-point improvements over Single-Agent.

Backbone	Method	T1	F1	F2
Qwen3-Max	Single-Agent	66.8	67.6	67.5
Qwen3-Max	MATEX	73.3	73.2	76.6
Llama-3-70B	Single-Agent	61.4	60.8	61.7
Llama-3-70B	MATEX	67.9	67.1	69.8
Llama-3-8B	Single-Agent	45.6	43.9	45.1
Llama-3-8B	MATEX	51.8	50.2	52.4

are reported in Table 13.

Table 13: Standard NLP metrics for EO1. ROUGE-L and BLEU are computed on transaction-level summaries; EM is computed on flow-level semantic attribute generation; Accuracy and Macro-F1 are computed on the 14-way action_type classification task.

Method	ROUGE-L	BLEU	EM	Acc.	Macro-F1
LLM	41.8	19.6	21.4	58.7	54.3
Few-shot CoT	43.6	21.3	24.1	61.5	57.1
VLM	42.9	20.7	22.9	60.6	55.8
Pure RAG	42.1	19.8	22.0	59.1	54.9
Single-Agent	47.9	24.3	28.6	66.8	62.7
MATEX	52.6	27.8	33.9	72.4	68.5

E.1.4 Judge Validation and Statistical Testing

This section provides additional details on judge reliability, cross-judge robustness, verbosity bias, and the statistical significance tests used to compare *MATEX* with the strongest non-*MATEX* baseline.

Judge Validation Setup. For EO1, the GPT-5.1 judge evaluates each system output on the four binary criteria defined in Table 14: **T1** (outcome correctness), **T2** (clarity & conciseness), **F1** (faithfulness), and **F2** (intent quality). To assess the reliability of this automatic evaluator, we compare its judgments against expert annotations produced by three DeFi Ph.D. researchers on the same transaction set.

GPT-5.1 reaches 86.4% overall agreement, with Cohen’s κ values of 0.72 for T1, 0.79 for T2, 0.74 for F1, and 0.76 for F2. As a model-choice robustness check, we repeat the evaluation with Llama-3-70B; it reaches 81.2% agreement and also identifies *MATEX* as the best method on F1/F2. Across the three human experts, inter-rater reliability is also high, with Fleiss’ $\kappa = 0.76$. The cross-judge robustness results are summarized in Table 15.

Judge Error Patterns. To better understand judge failure modes, we inspect its confusion patterns relative to expert labels. Most disagreements arise from borderline cases involving partial factual omissions (e.g., small fees or implicit approvals) rather than purely stylistic differences. This is consistent with the judge prompt, which emphasizes grounding in on-chain evidence and semantic adequacy rather than surface token flow alone.

Verbosity Bias Analysis. A common concern in LLM-as-a-judge evaluation is that longer outputs may receive artificially favorable scores. To test for this effect, we compute the Pearson correlation between output length and binary pass rate across the evaluated summaries. We obtain $r = 0.07$ and $p = 0.31$, indicating no statistically significant verbosity bias or preference for longer outputs.

Statistical Significance Testing. We compare *MATEX* against the strongest non-*MATEX* baseline on the same 187 transactions using paired statistical tests. For the binary pass/fail metrics (T1–F2), we use paired McNemar tests, since all systems are evaluated on the same instances. For continuous automatic metrics, we use paired significance tests appropriate for matched predictions: specifically, we report paired tests for ROUGE-L, Exact Match (EM), and action_type Macro-F1.² The resulting p -values are summarized in Table 16.

²We use paired bootstrap resampling for the continuous metrics.

Table 14: Evaluation for *TxSum*, divided into macro-level summarization and micro-level flow assessment.

Level	Metric	Definition
Transaction (Macro)	T1	Outcome Correctness: The summary accurately describes the final net balance changes, explicit fees, and critical non-token state changes (e.g., approvals, ownership transfers).
	T2	Clarity & Conciseness: The summary is concise (3-4 sentences), avoids technical jargon, and forms a coherent narrative.
Token Flow (Micro)	F1	Faithfulness: The predicted flow is semantically equivalent to the ground truth (i.e., factually correct actions and amounts).
	F2	Intent Quality: The explanation captures the <i>economic purpose</i> (the “why”) rather than just describing the mechanical transfer (the “what”).

Table 15: Robustness of LLM judges against expert majority labels for EO1.

Judge model	Agreement (%)	T1 κ	T2 κ	F1 κ	F2 κ	Best on F1/F2
GPT-5.1	86.4	0.72	0.79	0.74	0.76	MATEX
Llama-3-70B	81.2	0.63	0.70	0.66	0.68	MATEX

Table 16: Key paired significance tests comparing *MATEX* against the strongest baseline on EO1.

Metric	Test	<i>p</i> -value
T1	McNemar	0.118
F1	McNemar	< 0.001
F2	McNemar	0.003
ROUGE-L	Paired bootstrap	0.006
action_type Macro-F1	Paired bootstrap	0.004

Takeaway. These validation and significance results support two conclusions. First, the GPT-5.1 judge is sufficiently aligned with expert judgment to serve as a scalable evaluator for EO1, while expert inter-rater agreement further confirms the reliability of the human protocol. Second, the improvements achieved by *MATEX* on the key factuality- and intent-related metrics are statistically robust rather than artifacts of random variation on a limited test set.

E.2 EO2: User Trust & Comprehension

Participant Cohort and Recruitment. EO2 uses a new cohort of 20 active DeFi users, which is separate from the 16 participants in our formative interview study. The earlier 16 interviewees were used only to identify user needs and inform task design, whereas the EO2 cohort was recruited later for controlled evaluation. No participant took part in both studies.

We recruited the EO2 participants via Web3 com-

munities and professional networks across 6 countries/regions. Inclusion criteria were: (1) at least 3 on-chain DeFi transactions in the past 6 months; (2) familiarity with at least two major protocols (e.g., Uniswap, Aave, or Lido); and (3) age ≥ 18 . Because all interfaces, explanations, and questionnaires were presented in English, participants were also required to be able to read English DeFi interfaces and transaction explanations. Participants include practitioners, researchers/developers, students, and casual users. Their demographic distribution is summarized in Table 17.

Table 17: Demographics of EO2 user study participants ($N = 20$). Practitioners include traders and analysts; R&D includes researchers and developers.

Role	Count	Avg. Exp. (years)
Practitioners	4	4.2
R&D	5	3.8
Students	7	1.8
Casual Users	4	2.2

Study Design. EO2 is a **within-subject controlled study** using the EigenPhi block explorer interface under three conditions: (a) **Raw:** EigenPhi’s default view (call trace + token transfers); (b) **Single-Agent:** Raw explorer view augmented with annotations from the strongest non-*MATEX* baseline in EO1; (c) **MATEX:** Raw explorer view augmented with *MATEX* explanations.

Each participant viewed 9 transactions in total—3 simple, 3 medium, and 3 complex—covering 113 token transfers. Within each complexity level, one transaction was shown under each condition. Condition assignment and presentation order were balanced using a Latin Square design to mitigate order, transaction-identity, and complexity effects. Participants were not placed under a strict time limit, and most completed the study in approximately 40 minutes. For each transaction, participants completed the following questionnaire:

User Study Questionnaire

Comprehension (choose one for each question):

1. What was the main purpose of this transaction?
 A) To swap USDC for ETH and stake the ETH in Lido
 B) To withdraw liquidity from a Uniswap pool
 C) To repay a loan on Aave
 D) Not sure
2. Which of the following actions occurred during this transaction?
 A) Approved unlimited spending for a proxy contract
 B) Swapped USDC for DAI via a multi-hop route
 C) Deposited ETH into a lending protocol
 D) All of the above
3. What is the primary financial risk in this transaction?
 A) High slippage due to low liquidity
 B) Unlimited token approval granted to an unknown contract
 C) No significant risk
 D) Not sure

Perception (rate 1–5, where 1 = Strongly Disagree, 5 = Strongly Agree):

4. I understand what this transaction does.
5. I trust this explanation.

Scoring Procedure. The questionnaire contains two components. **(1) Comprehension Accuracy.** Participants answered three multiple-choice questions covering: (i) the main purpose of the transaction, (ii) a major action or mechanism involved in execution, and (iii) the most salient risk or side effect. Each correct answer received one point, yielding a per-transaction comprehension score from 0 to 3. We report mean accuracy as the percentage

of correctly answered questions aggregated across participants.

(2) Trust and Clarity. Participants rated the following statements on a 5-point Likert scale: (i) “I understand what this transaction does,” and (ii) “I trust this explanation.” We report the average of the two Likert items as the trust/clarity score for each condition and complexity level.

Statistical Analysis. Because each participant completed all three conditions, we used repeated-measures tests to compare comprehension accuracy and Likert-scale trust/clarity ratings across conditions. These analyses support the same overall pattern shown in the main text: *MATEX* yields the strongest gains in comprehension accuracy, with the largest advantage on medium- and high-complexity transactions, and also achieves the highest trust/clarity ratings overall.

The overall condition effect on comprehension accuracy is assessed with Cochran’s Q test, followed by post-hoc paired comparisons between *MATEX* and the two comparison conditions. For trust and clarity ratings, we similarly test for an overall condition effect and then examine pairwise differences. The detailed test statistics and pairwise comparison results are reported below in Table 18.

The statistical results are consistent with the main-text trends: *MATEX* shows a significant overall advantage in both comprehension accuracy and trust/clarity. The largest differences are observed relative to the Raw condition, while the gains over the Single-Agent baseline remain significant but smaller in magnitude. This pattern suggests that grounded, step-wise explanations are especially beneficial when users would otherwise rely on incomplete or weakly structured transaction views.

Interpretation. This design allows us to evaluate whether *MATEX* improves users’ objective understanding of transaction purpose and risk, as well as their subjective trust in the explanation, under matched interface conditions. Because the EO2 cohort is separate from the formative interview participants, the evaluation is not conflated with prior exposure to our study design or task framing.

E.3 EO3: Risk Awareness

Risk Transaction Selection. We select 5 real-world malicious transactions from major DeFi/NFT exploits, each representing a distinct and non-obvious risk pattern:

Table 18: Statistical comparisons for EO2. The pattern is consistent with the descriptive results in the main text: *MATEX* yields the strongest overall performance, with the largest gains over the Raw condition and smaller but still significant improvements over the Single-Agent baseline.

Comparison	Test	Result
Overall comprehension accuracy	Cochran’s Q	$p < 0.001$
<i>MATEX</i> vs. Raw (accuracy)	Paired McNemar	$p < 0.001$
<i>MATEX</i> vs. Single-Agent (accuracy)	Paired McNemar	$p = 0.012$
Overall trust/clarity	Friedman test	$p < 0.001$
<i>MATEX</i> vs. Raw (trust/clarity)	Wilcoxon signed-rank	$p < 0.001$
<i>MATEX</i> vs. Single-Agent (trust/clarity)	Wilcoxon signed-rank	$p = 0.018$

- **BadgerDAO** (Dec 2021): Infinite token approval via compromised frontend.³
- **WazirX** (Jul 2024): Gnosis Safe proxy upgrade via misleading UI.⁴
- **OpenSea** (Feb 2022): Malicious NFT order signature via phishing email.⁵
- **Bybit** (Feb 2025): Gnosis Safe mastercopy upgrade via compromised frontend.⁶
- **BXH Exchange** (Oct 2021): Direct fund theft due to missing multi-sig.⁷

We also include 5 benign transactions of comparable complexity as controls (e.g., standard multi-hop swaps, LP deposits).

User Evaluation. We conduct a dedicated between-subjects study with 20 participants (the same 20 participants in EO2), split into two groups of 10. (1) The **Raw group** views all 10 transactions (5 malicious, 5 benign) in a standard block explorer interface (EigenPhi). (2) The ***MATEX* group** views the same transactions with *MATEX* explanations overlaid on the same interface.

For every transaction, we simulate the pre-signing confirmation screen as it appeared in the original attack (e.g., in MetaMask or EigenPhi), showing the identical calldata and token flows that victims encountered. Participants see either (a) the raw transaction view or (b) the same view enhanced with *MATEX* annotations, with condition assignment balanced across users. Participants pro-

vide (1) their signing intention (Yes/No) and (2) an open-ended description of any risks they notice.

Two researchers independently code responses for risk detection using manual semantic coding guided by core threat concepts, rather than rigid keyword matching. For example, responses such as “this contract can keep spending my tokens forever” are credited as recognizing an unlimited approval even if the exact phrase is absent. The coders achieve Cohen’s $\kappa = 0.81$.

Scoring Procedure and Real Cases. For malicious transactions, we compute (i) rejection rate and (ii) correct threat-identification rate. For benign transactions, we compute the false-rejection rate. Threat identification is counted when the participant correctly recognizes the core exploit mechanism (e.g., unlimited approval, malicious proxy upgrade, hidden ownership change, phishing signature abuse), regardless of exact wording. Appendix F provides real examples showing how unconventional phrasing is still credited under this rubric.

User-Study Robustness. To quantify uncertainty under the existing participant sample, we report participant-level bootstrap confidence intervals, effect sizes, and odds ratios for the main EO2/EO3 comparisons in Table 19. Leave-one-participant-out and leave-one-transaction-out checks for EO3 preserve the qualitative conclusions, indicating that no single participant or transaction drives the gains.

E.3.1 Ablation Results

To better understand which architectural components drive the gains of *MATEX*, we perform ablations in which we disable one major component at a time while keeping the rest of the pipeline unchanged. We report EO1 judge-based metrics together with representative standard NLP metrics. The results are shown in Table 20.

³<https://www.trmlabs.com/resources/blog/trm-investigates-badgerdao-defi-protocol-hacked>

⁴<https://www.cloudsek.com/blog/wazirx-incident-explained>

⁵<https://www.certik.com/resources/blog/opensea-phishing-incident-analysis>

⁶<https://www.certik.com/resources/blog/bybit-incident-technical-analysis>

⁷<https://www.certik.com/resources/blog/boy-x-highspeed-incident-analysis>

Table 19: Participant-level uncertainty and effect-size analyses for EO2 and EO3.

Evaluation	Metric	Comparison	Estimate	Effect size	95% CI	Test
EO2	Complex comprehension	MATEX vs. Single-Agent	76.5% 52.9%	vs. $\Delta = +23.6$ pp	[9.4, 38.1]	Participant bootstrap
EO2	Complex comprehension	MATEX vs. Raw	76.5% 35.3%	vs. $\Delta = +41.2$ pp	[24.7, 56.8]	Participant bootstrap
EO2	Complex trust/clarity	MATEX vs. Single-Agent	3.7 vs. 2.6	Cohen’s $d_z = 1.08$	[0.52, 1.72]	Wilcoxon/bootstrap
EO3	Malicious rejection	MATEX vs. Raw	88.0% 36.0%	vs. OR = 13.04	[4.65, 36.52]	Fisher, $p < 0.001$
EO3	Threat identification	MATEX vs. Raw	72.0% 28.0%	vs. OR = 6.61	[2.76, 15.83]	Fisher, $p = 0.002$
EO3	Benign false rejection	MATEX vs. Raw	8.0% vs. 12.0%	OR = 0.64	[0.17, 2.41]	Fisher, $p = 0.741$

Table 20: Ablation results on EO1 and representative standard NLP metrics. Each variant disables one component while keeping the remaining pipeline unchanged. T1–F2 are LLM-as-judge pass rates (%), while ROUGE-L and action_type Macro-F1 evaluate transaction-level summary quality and structured semantic parsing, respectively. Arrows indicate absolute drops relative to full MATEX.

Variant	T1	T2	F1	F2	ROUGE-L	Action Macro-F1
w/o Profiler	68.4 _{↓4.9}	88.2 _{↓4.8}	70.5 _{↓2.7}	74.2 _{↓2.4}	50.2 _{↓2.4}	66.0 _{↓2.5}
w/o Investigator	57.2 _{↓16.1}	74.9 _{↓18.1}	62.8 _{↓10.4}	70.3 _{↓6.3}	46.3 _{↓6.3}	60.4 _{↓8.1}
w/o Synthesizer	47.1 _{↓26.2}	64.2 _{↓28.8}	57.9 _{↓15.3}	68.5 _{↓8.1}	41.8 _{↓10.8}	56.5 _{↓12.0}
w/o Auditor	72.2 _{↓1.1}	91.4 _{↓1.6}	62.5 _{↓10.7}	74.9 _{↓1.7}	49.9 _{↓2.7}	62.0 _{↓6.5}
Full MATEX	73.3	93.0	73.2	76.6	52.6	68.5

F Case Studies

We present three cases showing why *TxSum* matters: coarse intent labels and final balances do not give users enough information to act safely.

Case 1: Aggregator-Routed Swap with Hidden Intermediate Fees. A user swaps ETH for DAI through 1inch. The transaction crosses multiple contracts: ETH moves to the 1inch router, which swaps ETH→USDC on Uniswap V3, routes USDC through an executor, and completes USDC→DAI on SushiSwap. The trace includes a small ETH refund, a relayer fee, and a routing-related deduction.

Why it matters. This case shows why user-facing transaction understanding cannot stop at a coarse label such as “swap.” For signing safety, the user also needs to know whether the transaction introduces unfamiliar intermediary contracts, applies hidden deductions, or contains additional state changes beyond the intended exchange.

Why it is hard. The overall intent is relatively easy to identify, but a faithful explanation is more demanding. A model can correctly infer that the transaction is a swap while still omitting the intermediate fee transfer, misclassifying executor hops, or failing to explain why multiple protocols appear in what seems to be a single operation. This gap between macro intent and micro-level factual

completeness is central to our task.

Annotation and evaluation implication. Intermediate executor hops may be labeled as route, but fee-bearing transfers must remain explicit. A summary that only states “the user swapped ETH for DAI” is incomplete if it omits economically material fees or intermediate deductions.

Case 2: Recursive Leverage via Aave, CowSwap, and Lido. A user borrows USDC from Aave, swaps the USDC into ETH through CowSwap, stakes the ETH into stETH through Lido, supplies the stETH back to Aave as collateral, and then borrows again. The position is a leveraged staking strategy assembled through borrowing, swapping, staking, derivative issuance, collateral supply, and re-borrowing.

Why it matters. This case reflects a realistic DeFi workflow in which the main user risk does not lie in a single transfer, but in the dependency structure created across multiple protocols. For transaction explanation, it is not sufficient to say that the user “opened a leveraged position”; the explanation should reveal how the leverage is constructed and where the exposure comes from.

Why it is hard. A coarse transaction label can be correct while still being inadequate. The trace mixes economically distinct actions that are easy to

compress into one high-level description, yet such compression hides the repeated collateralization pattern that determines liquidation sensitivity and protocol dependence. The difficulty therefore lies in preserving enough step-wise structure to remain faithful without losing readability.

Annotation and evaluation implication. Each borrow, swap, staking, collateral deposit, and re-borrow step is annotated independently. This case shows why *TxSum* needs micro-level semantic parsing, not just top-level intent prediction.

Case 3: Safe Wallet Mastercopy Upgrade with Minimal Token Movement. A transaction presented in the wallet interface appears to be a routine administrative confirmation, but in fact changes the mastercopy or implementation address of a Safe-style wallet. At signing time, there may be little or no visible token movement, even though control over future wallet behavior is effectively altered.

Why it matters. This is representative of the transactions that are most dangerous for blind-signing: the immediate balance outcome looks harmless, but the security consequence is severe. A user-facing explanation must therefore surface non-token state changes when they alter ownership, upgrade authority, or execution control.

Why it is hard. Purely token-centric reasoning is insufficient here because the critical event is not a transfer but a control-plane change. A system that focuses only on token flows or final balances may produce an explanation that is mechanically accurate yet fails to communicate the actual security implication. This makes the case especially important for high-stakes explanation settings.

Annotation and evaluation implication. These cases motivate why *TxSum* includes transaction-level summaries that must mention critical non-token state changes. An explanation that only paraphrases the low-level call mechanics, but does not surface the control-transfer risk, is considered inadequate for user-facing safety.