Cross-Referencing Whitepaper Claims and Smart Contract Code for Automated ICO Auditing

Chenglong Fu*, Chi Zhang†, Mengtian Li‡
*University of North Carolina at Charlotte, USA, chenglong.fu@charlotte.edu
†Trine University, USA, czhang244@my.trine.edu
‡University of North Carolina at Greensboro, USA, m_li10@uncg.edu

***Abstract*—Initial Coin Offering (ICO) whitepapers serve as informal specifications for token-governing smart contracts, yet no existing tool verifies whether deployed code matches the claims in these documents. NLP-based fraud detection analyzes whitepaper text but ignores the contract. Smart contract analysis tools examine code but have no knowledge of what developers promised. We built ICOAudit, a pipeline that bridges this gap. ICOAudit extracts structured claims from ICO whitepapers using LLM-based information extraction across nine token-governance categories, analyzes the corresponding Ethereum smart contracts via Slither static analysis, and checks each claim against code-level features using a hybrid rule-based and LLM-based consistency scorer. We evaluate ICOAudit on a novel paired dataset of 60 real Ethereum ICOs with whitepaper text, verified contract source, and fraud labels. The pipeline detected 86 specification-implementation inconsistencies across 21 of 54 analyzed projects. Fraud-labeled projects exhibited 3.0× higher combined inconsistency signals than legitimate ones (p=0.075, suggestive but below the 0.05 significance threshold), while text-only analysis showed no discrimination (p=0.569). The five highest-severity inconsistencies, where entire promised subsystems were absent from deployed code, all involved confirmed fraud cases. Claim extraction validated against hand-labeled ground truth from Cohney et al. achieved F1=0.72 across 19 ICOs.**

***Index Terms*—blockchain, smart contracts, ICO fraud detection, specification verification, NLP, static analysis**

1. Introduction

An ICO whitepaper promises a hard cap of 100 million tokens, but the deployed smart contract contains an unrestricted `mint()` function callable by the contract owner. No automated tool flags the discrepancy. Between 2014 and 2019, ICOs raised over $12.9 billion from investors [1], while cumulative losses to cryptocurrency fraud now exceed $16.7 billion [2]. Smart contracts are immutable once deployed and govern real investor funds. The whitepaper functions as the closest analog to a securities prospectus, but unlike prospectuses, no regulatory or technical mechanism verifies that the code matches the claims. This is a specification-to-implementation gap: the whitepaper is the informal specification, and the contract is the deployed system.

Two research communities have studied parts of this problem in isolation. NLP researchers analyze whitepaper text to predict fraud [3], [4], [5], achieving up to F1=0.80 [3] or AUC=0.94 [5]. Yet Hornuf et al. [6] found that fraud is "extremely difficult to predict" from pre-issuance text alone. Smart contract security researchers build tools that analyze code for vulnerabilities [7], [8], [9] but never reference the whitepaper. They find bugs but cannot distinguish intentional fraud mechanisms from honest mistakes without knowing what the developers promised. Luo et al. [2], the most comprehensive AI fraud detection survey, explicitly identify NLP-based and smart-contract-based analysis as two separate tracks with no intersection.

We built ICOAudit, to our knowledge the first pipeline that cross-references off-chain whitepaper claims against on-chain smart contract implementations. Three contributions:

1) A **claim-to-code verification pipeline** that extracts structured claims from whitepapers using LLM-based information extraction and checks them against contract features from Slither static analysis.
2) A **novel paired dataset** of 60 Ethereum ICO projects with whitepaper text, smart contract source code, and fraud labels (12 fraud-labeled, 6 with SEC/DOJ enforcement actions).[1] To our knowledge, no existing dataset pairs whitepaper text with contract source and fraud labels.
3) **Empirical findings** from running the pipeline on real projects: 86 specification-implementation inconsistencies detected across 21 of 54 analyzed ICOs, with fraud projects showing 3.0× higher combined inconsistency signals than legitimate ones (p=0.075). The five highest-severity inconsistencies, where entire promised subsystems were absent, all come from confirmed fraud cases.

We validated claim extraction against the Cohney et al. [10] hand-labeled ground truth, achieving F1=0.72 across 19 ICOs with four governance dimensions.

1. Fraud labels from SEC enforcement actions (KIN, PRG, CMCT, GLA), DOJ indictments (BCC, CTR), exchange delisting, and community fraud databases.

The ICOAudit pipeline performs specification-implementation consistency verification: checking whether a deployed financial system (the contract) conforms to its specification (the whitepaper). This is analogous in spirit to specification conformance checking in safety-critical systems and aligns with DSN's dependability mission. Section 2 provides background. Section 3 describes the pipeline. Section 4 presents evaluation results. Section 5 discusses lessons learned. Section 6 covers related work and conclusions.

2. Background and Motivation

ICOs operate without the disclosure standards of traditional securities. Whitepapers are unregulated, unstandardized, and vary wildly in quality. Bourveau et al. [11] documented substantial variation across 2,113 ICOs, finding that disclosure quality correlates with capital-raising ability. Morin et al. [12] found 19% of whitepapers are near-copies of existing ones and 67% show moderate text reuse.

The regulatory landscape is shifting. The EU Markets in Crypto-Assets Regulation (MiCA) [13], fully operational since December 2024, mandates detailed whitepaper disclosures and will require machine-readable iXBRL format, creating direct infrastructure for automated verification. In the US, the GENIUS Act [14] requires reserve backing and audits for stablecoin issuers. Despite $3.4 billion stolen in crypto during 2025, no major jurisdiction yet requires systematic verification of whitepaper claims against deployed smart contract implementations. SEC enforcement has been volatile: FY 2024 produced a record $8.2 billion in remedies, but new enforcement actions in 2025 fell to the lowest in a decade. This volatility underscores the need for technical auditing tools that operate independently of enforcement posture.

The specification-implementation gap is concrete and documented. A whitepaper claims "total supply capped at 100 million," but the contract has no supply cap. A whitepaper promises "team tokens locked for 12 months," but the contract has no time-lock mechanism. A whitepaper states "no admin can modify the contract," but the code contains an `onlyOwner` function that can pause transfers. Each of these scenarios has been documented in the literature. Cohney et al. [10] manually audited 50 ICOs and found significant discrepancies between promises and implementations. Existing tools cannot detect these discrepancies because they analyze only one side, text or code, in isolation. Figure 1 shows a real example from our dataset.

3. System Architecture

Figure 2 shows the ICOAudit pipeline. It takes an ICO whitepaper and a smart contract source file as input and produces a per-claim consistency report. The pipeline has four stages: section parsing, claim extraction, contract analysis, and consistency scoring.

TABLE 1. DATASET COMPOSITION

Attribute	Value
Total ICO pairs	60
Legitimate	31
Fraud (regulatory-confirmed)	6
Fraud (community-labeled)	6
Borderline	17
Cohney ground truth overlap	19*
Contracts analyzed (Slither)	57/60
Total extracted claims	758
Claims mapped to code	732
Inconsistencies detected	86

*ENG excluded (non-token document).

3.1. Dataset Construction

We assembled a paired dataset of ICO whitepapers and their corresponding smart contract source code (Table 1). Starting from the Token Offerings Research Database (TORD v4, 10,807 entries), we filtered to 1,153 entries with both a whitepaper URL and a contract address. We then cross-referenced Sourcify for verified contract source code, SEC enforcement actions and DOJ press releases for fraud labels, and the Cohney et al. [10] corpus for ground truth validation. Three enrichment rounds using the Morin whitepaper corpus, CoinGecko API, smart-contract-sanctuary, and DOJ News API yielded 60 pipeline-ready pairs: 31 legitimate, 12 fraud-labeled (6 with SEC/DOJ enforcement actions, 6 from community sources), and 17 borderline. To our knowledge, no prior work pairs whitepaper text with corresponding smart contract source code and fraud labels.

3.2. Claim Extraction (Stages 1–2)

Whitepapers are parsed into structured sections via pymupdf, retaining headings and body text. Image-based PDFs required OCR as a fallback (1 of 60). Five whitepapers required re-extraction during development due to input-quality issues: an audit report indexed as a whitepaper, raw HTML from a Wayback Machine capture, a job posting, and PDFs with garbled text.

Claims are then extracted using a 122-billion-parameter LLM (Qwen3.5-122B) with a nine-category taxonomy designed to capture verifiable token-governance assertions: C1 Supply (caps, inflation limits), C2 Distribution (allocation percentages), C3 Vesting (lockup periods, escrow), C4 Burning/Minting (supply-change functions), C5 Fee/Reward (transaction fees), C6 Governance (voting, DAO), C7 Upgrade (proxy patterns), C8 Access Control (owner privileges, pause mechanisms), and C9 Timeline (dates, milestones). Each claim is extracted with a verbatim whitepaper quote, a confidence level, and a verification type. The taxonomy extends Cohney et al.'s four categories (scarcity, burning, vesting, modification) to capture the full range of verifiable assertions. Across 60 ICOs, the pipeline extracted 758 claims from 54 projects; 6 produced zero

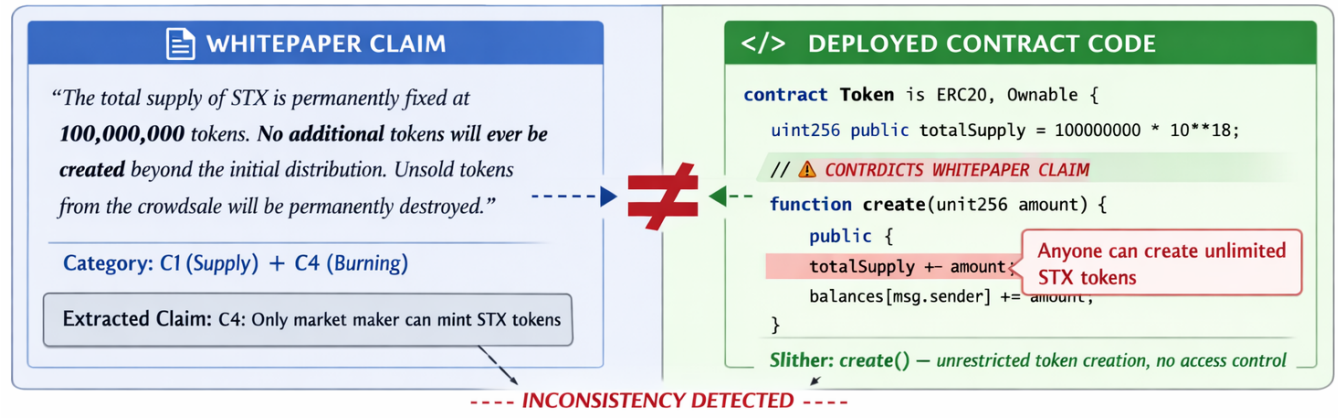


Figure 1. A real specification-implementation inconsistency from our dataset. The Stox (STX) whitepaper claims the market maker has "sole permission to create STX," but the deployed contract exposes `create()` as a public function callable by anyone during the sale. ICOAudit detects this by extracting the minting-permission claim (C4) and cross-referencing it against the contract's function signatures.

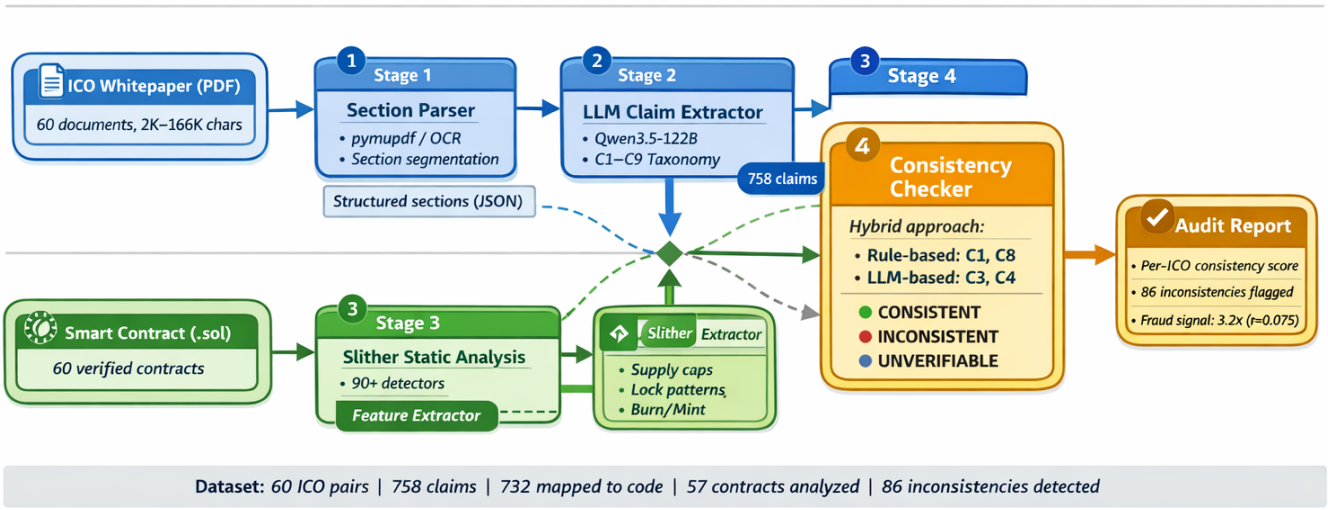


Figure 2. ICOAudit pipeline architecture. The text-side pipeline (top) extracts structured claims via section parsing and LLM-based extraction across 9 categories. The code-side pipeline (bottom) analyzes contract source via Slither and maps outputs to claim-relevant features. Stage 4 uses hybrid checking: rule-based for C1/C8, LLM-based for C3/C4.

claims due to content limitations (marketing-heavy text, extremely short documents).

3.3. Contract Analysis (Stage 3)

All contracts with verified Solidity source are analyzed using Slither [7], the industry-standard static analysis framework with 90+ built-in detectors. For each contract, we extract function signatures with visibility and modifiers, state variables (`totalSupply`, `cap`, `owner`), access control patterns (`onlyOwner`, `require` statements, role-based checks), and detector findings grouped by severity. Of 60 contracts, 57 compiled and analyzed successfully (95%). Three failed due to a missing source file, unresolvable multi-file imports, and a Solidity version incompatibility. Slither produced approximately 3,200 findings across the 57 contracts. Solidity versions ranged from 0.4.2 to 0.8.23, with the majority using 0.4.x-era compilers. We chose Slither over symbolic execution tools (Mythril, Manticore) because static analysis scales to the full dataset without path-explosion issues, and Slither's SSA-based intermediate representation provides precise function-level feature extraction.

Slither outputs are post-processed into structured feature vectors mappable to claim categories. C1 maps to `totalSupply()` values and the presence of a `cap` variable. C3 maps to time-lock patterns, `vestingSchedule`, and `releaseTime` state variables. C4 maps to `burn()` and `mint()` functions with caller restrictions. C8 maps to `onlyOwner` modifier usage and `Ownable` inheritance. Feature extraction coverage varies by category: C1 features are present in 100% of contracts, C8 in 88%, C3 in 67%, C4 in 61%. C5 fee patterns appear in only 4% of contracts because most ICO-era ERC20 tokens did not

implement fee-on-transfer logic in the token contract itself. C6 (Governance) and C7 (Upgrade) lack reliable code-level patterns with our current approach and are marked UNVERIFIABLE. We retain these categories in the extraction taxonomy for completeness and to support future extensions with richer analysis.

3.4. Consistency Scoring (Stage 4)

Each extracted claim is checked against its mapped code features. We use a hybrid approach: deterministic rules for categories with well-defined code patterns (C1 Supply, C8 Access Control) and LLM-based comparison for categories requiring semantic judgment (C3 Vesting, C4 Burn/Mint). Every claim receives one of four verdicts: CONSISTENT (code implements the claim), INCONSISTENT (code contradicts the claim), PARTIAL (some evidence of implementation but incomplete or ambiguous), and UNVERIFIABLE (insufficient code-level features to check).

Rule-based scoring for C1 checks whether a claimed supply cap matches a value in the contract source. If the claim asserts fixed supply but the contract contains a `mint()` function, the verdict is INCONSISTENT. For C8, if the whitepaper claims "no admin privileges" but the code inherits `Ownable` or uses `onlyOwner` modifiers, the verdict is INCONSISTENT. LLM-based scoring for C3 and C4 presents the claim text alongside the relevant code snippet and asks for a verdict with a one-sentence justification.

Across 732 mapped claims, the pipeline produced: 14 CONSISTENT (1.9%), 86 INCONSISTENT (11.7%), 241 PARTIAL (32.9%), and 391 UNVERIFIABLE (53.4%). The high unverifiable rate reflects two realities: many whitepaper claims reference logic that lives in separate contracts (multi-contract architectures), and categories C5–C7 lack reliable code-level patterns in our current feature extractor.

4. Evaluation

4.1. Detected Inconsistencies

The 86 inconsistencies span 21 ICOs. The five highest-severity cases, where entire promised subsystems are absent from deployed code, all involve fraud-labeled projects.

Bitconnect (BCC). The whitepaper promises investors can "withdraw capital anytime after 299 days." The contract contains only an owner-controlled `freezeAccount()` function. No time-based withdrawal logic, no 299-day parameter, no investment tracking. The lending platform was entirely off-chain, consistent with the DOJ's Ponzi scheme charges.

Stox (STX). Two inconsistencies. First, the whitepaper claims the market maker has "sole permission to create STX." The contract exposes `create()` as `public payable onlyDuringSale`, callable by anyone during the sale. Second, the whitepaper describes a "per-event collateral system locking STX until 24 hours after game resolution." The contract implements only generic linear vesting with no event-based collateral mechanism. The SEC charged Stox's founder with fraud in 2018.

TABLE 2. REGULATORY-CONFIRMED FRAUD CASES AND PIPELINE FINDINGS

Symbol	Source	Claims	Inc.	Key Finding
BCC	DOJ+SEC	1	1	No 299-day withdrawal
CTR	DOJ	2	0	Claims unverifiable
PRG	SEC	38	0	3 consistent, 11 partial
CMCT	SEC	1	0	Claim unverifiable
GLA	SEC	4	0	All claims partial
KIN	SEC	16	4	Foundation lock missing

Kin (KIN). The whitepaper states "the remaining six trillion kin will be locked under the Kin Rewards Engine schema." The contract has a per-user vesting/grant mechanism (`unlockVestedTokens`) but no Foundation-level lock, no 6-trillion allocation tracking, and no Rewards Engine. The SEC settled with Kik Interactive (Kin's issuer) in 2020 for operating an unregistered securities offering.

Bee Token (BEE). The whitepaper describes a staking mechanism where "participants stake BEE tokens to the arbitration protocol." The contract is a standard ERC20 with a crowdsale `buy()` function. No staking infrastructure, no arbiter registry, no dispute resolution logic.

These inconsistencies share a common pattern: the whitepaper describes a complex mechanism (time-locked withdrawal, staking protocol, foundation-level lock, collateral system), but the contract implements only basic ERC20 functionality with none of the promised logic present.

The pipeline also produces correct consistency verdicts. For example, Bancor (BNT) claims a fixed initial supply of 79,323,978 tokens. The contract defines `uint256 public constant TOKEN_AMOUNT = 79323978 * 10**18`, and our C1 rule-based checker matches this value exactly, returning CONSISTENT. Across 732 mapped claims, 14 received CONSISTENT verdicts (1.9%), 86 INCONSISTENT (11.7%), 241 PARTIAL (32.9%), and 391 UNVERIFIABLE (53.4%). The low consistent count reflects the pipeline's conservative design: a CONSISTENT verdict requires a direct, unambiguous match between claim and code, while PARTIAL is assigned when relevant code structures exist but do not fully confirm the claim.

Table 2 summarizes findings for all six regulatory-confirmed fraud cases. Notably, CTR (Centra Tech, DOJ-indicted) and PRG (Paragon Coin, SEC-settled) produced no inconsistencies despite being confirmed fraud. Both have relatively well-implemented token contracts; their fraud involved misrepresentation of team credentials and business partnerships, not contract-level deception. This illustrates a fundamental limitation: ICOAudit detects specification-implementation gaps, not all fraud vectors.

4.2. Fraud Discrimination

We compare three signal modalities for distinguishing fraud from legitimate projects (Figure 3). Text-only analysis (total claim count per ICO) shows no discrimination:

TABLE 3. CLAIM EXTRACTION AND VERIFICATION COVERAGE BY CATEGORY

Cat	Description	Claims	F1	Cov.
C1	Supply	76	0.71	100%
C2	Distribution	64	–	100%
C3	Vesting	172	**0.76**	67%
C4	Burn/Mint	115	0.67	61%
C5	Fee/Reward	114	–	4%
C6	Governance	66	–	0%
C7	Upgrade	10	–	0%
C8	Access Ctrl	66	0.46	**88%**
C9	Timeline	75	–	63%
Total		**758**	**0.72**	**56.7%**

fraud projects average 14.1 claims versus 13.4 for legitimate (p=0.569, Mann-Whitney U). This baseline tests whether the volume of verifiable promises alone separates fraud from legitimate projects; it is not a substitute for the richer NLP models in prior work [3], [4], which use additional textual features. Code-only analysis (Slither high-severity findings) shows marginal discrimination: fraud contracts average 1.89 high-severity findings versus 1.00 for legitimate (p=0.075). The combined signal (inconsistency rate plus normalized Slither high findings) achieves the strongest separation: fraud projects score 0.623 ± 0.197 versus 0.208 ± 0.046 for legitimate, a 3.0× difference (p=0.075, Cohen's $d \approx 0.95$).

The p-values are marginal, not significant at the 0.05 level. With only n=9 fraud projects (after excluding 3 with zero claims or compilation failures), statistical power is approximately 0.35 for detecting a medium effect. The large effect size (d=0.95) suggests a real signal that is underpowered in this sample. A dataset with $n \geq 20$ fraud projects per group would likely achieve significance.

4.3. Claim Extraction Quality

We validated claim extraction against the Cohney et al. [10] ground truth, which provides hand-labeled binary assessments across four governance dimensions (scarcity, burning, vesting, modification) for 50 ICOs. Nineteen of these overlap with our dataset. For each ICO-dimension pair, we check whether our pipeline extracted at least one relevant claim matching Cohney's label.

Table 3 shows per-category results. Overall F1 is 0.72 (precision 0.74, recall 0.70). C3 Vesting achieves the highest F1 (0.76), likely because vesting language in whitepapers tends to be specific and unambiguous. C8 Access Control has the lowest F1 (0.46), reflecting a taxonomy mismatch: Cohney's "modification" dimension captures contract upgradeability, while our C8 captures ownership and access control more broadly. The "Cov." column shows what fraction of claims in each category have code-level features available for consistency checking.

Six ICOs produced zero claims, all due to input quality: marketing-heavy text without verifiable assertions, extremely short documents, or non-token content. These are correctly handled as edge cases rather than pipeline failures.

5. Lessons Learned

Input quality dominates pipeline reliability. Five of 60 whitepapers required manual re-extraction. Root causes included an audit report indexed as a whitepaper, raw HTML from a Wayback Machine capture, and image-based PDFs. Such input-quality problems are endemic to unregulated disclosure and would affect any production system. A related observation from data acquisition: sourcing verified contract source code disproportionately excluded suspected fraud projects. Many ICOs flagged as scams had no verified Solidity source on Etherscan or Sourcify. Our dataset required verified source by construction, creating a survivorship bias toward better-documented projects. Extending the pipeline to bytecode-only analysis via decompilation would broaden fraud coverage. MiCA's iXBRL requirement would eliminate most parsing issues by mandating machine-readable whitepapers with standardized taxonomies.

LLM extraction requires careful calibration. Claim extraction with a 122B-parameter model achieved F1=0.72, adequate for a first-pass pipeline but below the threshold for fully automated auditing. Two failure modes dominated. Recall failures: the LLM missed claims buried in dense sections or stated implicitly. Precision failures: the LLM extracted marketing aspirations as verifiable claims. Adding targeted recall instructions and few-shot examples for underperforming categories (C3 Vesting) improved F1 from 0.68 to 0.72. The LLM-based consistency scorer (Stage 4) operates at lower temperature (0.3 vs. 0.7 for extraction) to reduce verdict variability.

The verifiability gap is the largest limitation. Over half (53.4%) of mapped claims received UNVERIFIABLE verdicts. Two structural causes explain this. First, ICO-era tokens frequently use multi-contract architectures: the token contract handles ERC20 logic while crowdsale, vesting, and governance logic live in separate contracts that we do not currently analyze. Second, categories C5 (Fee), C6 (Governance), and C7 (Upgrade) lack reliable code-level feature patterns in our regex-based extractor. Addressing the first cause requires cross-contract analysis (following `delegatecall` and proxy patterns). Addressing the second requires richer static analysis, possibly through GPTScan-style [8] LLM-augmented program understanding.

Practical deployment is feasible. The full pipeline processes 60 ICOs in approximately 48 minutes: 10 minutes for PDF parsing, 30 minutes for LLM-based claim extraction (the bottleneck), 5 minutes for Slither analysis, and under 3 minutes for feature extraction and consistency scoring. Using vLLM with 6-thread concurrency reduced Stage 2 from an estimated 20 hours (single-threaded) to 30 minutes, a 40× speedup. For a production deployment targeting MiCA-compliant whitepapers, Stages 1–2 would simplify dramatically because iXBRL-formatted documents provide structured claims directly. The tool could integrate into investor due-diligence workflows or regulatory review processes.

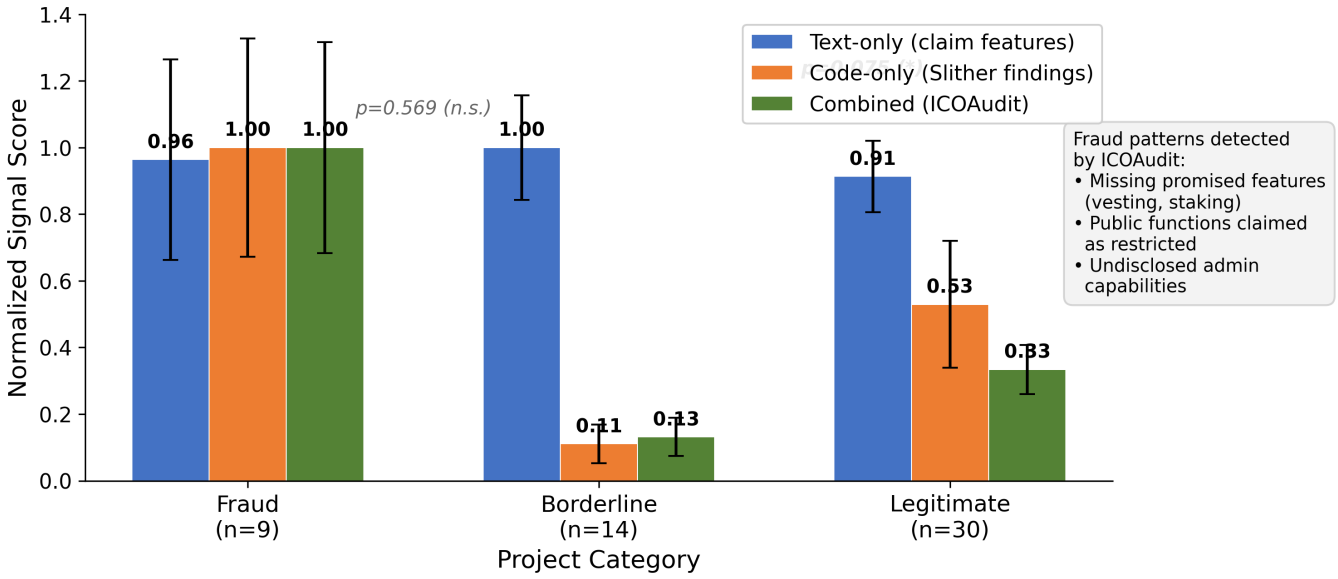


Figure 3. Fraud discrimination across three signal modalities. Each signal is normalized to $[0, 1]$ by its maximum value across groups. The combined signal (inconsistency rate plus normalized Slither high-severity findings) achieves 3.0× separation between fraud and legitimate projects (p=0.075), while text-only claim counts show no discrimination (p=0.569). Error bars indicate standard error of the mean.

5.1. Threats to Validity

External and construct validity. Our dataset of 60 Ethereum-based ICOs from the 2017–2018 era may not generalize to newer DeFi protocols or non-Ethereum chains. The requirement for verified Solidity source creates survivorship bias, excluding the worst fraud projects that never published source code. Our fraud labels combine SEC/DOJ enforcement actions (6 projects, high confidence) with community databases (6 projects, lower confidence). "Inconsistency" does not always indicate fraud: legitimate projects may show discrepancies due to post-whitepaper design changes or multi-contract deployments.

Internal and conclusion validity. LLM-based extraction and scoring introduce nondeterminism, mitigated by low temperature settings (0.3 for scoring, 0.7 for extraction) but not characterized via inter-run variance. The hybrid scoring approach (rule-based for C1/C8, LLM-based for C3/C4) may introduce systematic bias across categories. With n=9 usable fraud projects, statistical power is approximately 0.35 at the 0.05 level. The p=0.075 results and large effect size (Cohen's $d\approx$0.95) are suggestive but not conclusive; a dataset with $n\geq$20 fraud projects per group would be needed to draw firm conclusions.

6. Related Work and Conclusion

Three research streams converge in this work but have not previously intersected. NLP-based ICO fraud detection [3], [4], [5], [6] analyzes whitepaper text and metadata but ignores smart contract code. Smart contract security analysis [7], [8], [9], [15] examines code for vulnerabilities but has no access to developer promises. Blockchain forensics and on-chain analysis [2], [16] traces transaction patterns but does not connect them to whitepaper claims.

The closest prior work is Cohney et al. [10], who manually audited 50 ICOs across four governance dimensions. ICOAudit automates and extends their approach to nine categories across 60 projects. GPTScan [8] pioneered combining LLMs with static analysis for smart contract auditing but targets generic vulnerability detection rather than specification conformance. CRPWarner [9] detects rug pull patterns in bytecode but cannot assess whether detected mechanisms contradict whitepaper promises. Luo et al. [2] explicitly identify NLP-based and code-based fraud detection as separate tracks in their 2025 survey, confirming the gap that ICOAudit addresses.

ICOAudit bridges these communities by treating the whitepaper as an informal specification and the smart contract as the deployed system, casting ICO auditing as specification-implementation consistency verification. Our results show that cross-referencing text and code produces a stronger fraud signal than either source alone, and that the highest-severity inconsistencies correspond to confirmed fraud. Three directions for future work: (1) multi-contract analysis to reduce the 53.4% unverifiable rate, (2) expanding the dataset beyond 60 ICOs to achieve statistical significance, and (3) adapting the pipeline for MiCA-compliant iXBRL whitepapers where structured claims are machine-readable by design.

Acknowledgment

This work was supported by Faculty Research Grant from the UNC Charlotte (Fund No.111222).

References

[1] S. T. Howell, M. Niessner, and D. Yermack, "Initial coin offerings: Financing growth with cryptocurrency token sales," *Review of Financial Studies*, vol. 33, no. 9, pp. 3925–3974, 2020.

[2] B. Luo, Z. Zhang, Q. Wang, A. Ke, S. Lu, and B. He, "AI-powered fraud detection in decentralized finance: A project life cycle perspective," *ACM Computing Surveys*, vol. 57, no. 4, pp. 96:1–96:38, 2025.

[3] S. Bian, Z. Deng, F. Li, W. Monroe, P. Shi, Z. Sun, W. Wu, S. Wang, W. Y. Wang, A. Yuan, T. Zhang, and J. Li, "IcoRating: A deep-learning system for scam ICO identification," in *arXiv preprint arXiv:1803.03670*, 2018.

[4] A. Dürr, M. Griebel, G. Welsch, and F. Thiesse, "Predicting fraudulent initial coin offerings using information extracted from whitepapers," in *Proc. 28th European Conference on Information Systems (ECIS)*, 2020.

[5] Y. Zheng and M. Wang, "Imbalanced problem in initial coin offering fraud detection," in *Proc. Int. Conf. Pioneering Computer Scientists, Engineers and Educators (ICPCSEE)*, ser. Communications in Computer and Information Science, vol. 1629. Springer, 2022.

[6] L. Hornuf, T. Kück, and A. Schwienbacher, "Initial coin offerings, information disclosure, and fraud," *Small Business Economics*, vol. 58, no. 4, pp. 1741–1759, 2022.

[7] J. Feist, G. Grieco, and A. Groce, "Slither: A static analysis framework for smart contracts," in *Proc. IEEE/ACM Int. Workshop on Emerging Trends in Software Engineering for Blockchain (WETSEB)*, 2019.

[8] Y. Sun, D. Wu, Y. Xue, H. Liu, H. Wang, Z. Xu, X. Xie, and Y. Liu, "GPTScan: Detecting logic vulnerabilities in smart contracts by combining GPT with program analysis," in *Proc. IEEE/ACM 46th Int. Conf. Software Engineering (ICSE)*, 2024, pp. 2048–2060.

[9] Z. Lin, J. Chen, J. Wu, W. Zhang, Y. Wang, and Z. Zheng, "CRPWarner: Warning the risk of contract-related rug pull in DeFi smart contracts," *IEEE Transactions on Software Engineering*, vol. 50, pp. 1534–1547, 2024.

[10] S. Cohney, D. Hoffman, J. Sklaroff, and D. Wishnick, "Coin-operated capitalism," *Columbia Law Review*, vol. 119, no. 3, pp. 591–676, 2019.

[11] T. Bourveau, E. T. D. George, A. Ellahie, and D. Macciocchi, "The role of disclosure and information intermediaries in an unregulated capital market: Evidence from initial coin offerings," *Journal of Accounting Research*, vol. 60, no. 1, pp. 129–167, 2022.

[12] A. Morin, M. Vasek, and T. Moore, "Detecting text reuse in cryptocurrency whitepapers," in *Proc. IEEE International Conference on Blockchain and Cryptocurrency (ICBC)*, 2021.

[13] European Parliament and Council, "Regulation (EU) 2023/1114 on markets in crypto-assets (MiCA)," 2023, fully operational December 30, 2024.

[14] United States Congress, "GENIUS act: Guiding and establishing national innovation for U.S. stablecoins," 2025, signed July 18, 2025.

[15] T. Durieux, J. F. Ferreira, R. Abreu, and P. Cruz, "Empirical review of automated analysis tools on 47,587 Ethereum smart contracts," in *Proc. IEEE/ACM Int. Conf. Software Engineering (ICSE)*, 2020.

[16] F. Cernera, M. L. Morgia, A. Mei, and F. Sassi, "Token spammers, rug pulls, and sniper bots: An analysis of the ecosystem of tokens in Ethereum and in the Binance Smart Chain (BNB)," in *Proc. USENIX Security Symposium*, 2023.