ProvAuditChain: A Gas-Efficient On-Chain Provenance Framework for AI-Driven Smart Contract Audits

George Chidera Akor [1], Love Allen Chijioke Ahakonye [2], Jae Min Lee [1], Dong-Seong Kim [1] *
[1] IT-Convergence Engineering, *Kumoh National Institute of Technology*, Gumi, South Korea
[2] ICT Convergence Research Center, *Kumoh National Institute of Technology*, Gumi, South Korea
* NSLab Co. Ltd., Gumi, South Korea, *Kumoh National Institute of Technology*, Gumi, South Korea
(georgeakor, loveahakonye, ljmpaul, dskim@kumoh.ac.kr)

***Abstract*—The increasing use of AI-powered tools for smart contract security audits presents a critical challenge: ensuring the integrity and provenance of audit reports. Traditional methods lack cryptographic guarantees linking audit outputs to specific AI models and source code, creating vulnerabilities to tampering and misattribution. To address this, we propose ProvAuditChain, a gas-efficient, hybrid on-chain/off-chain framework that records immutable provenance of AI-driven audit reports on Layer 2 blockchain networks. ProvAuditChain utilizes lightweight smart contracts to securely anchor cryptographically signed audit report hashes on-chain, while storing the complete reports on decentralized IPFS storage. We deploy the system on the Arbitrum Sepolia testnet and benchmark gas consumption, latency, and throughput across 600 audit cycles. Our results demonstrate a stable average gas cost of approximately 173,000 per audit, equivalent to roughly $0.05 USD, alongside a throughput of nearly six audits per minute. These findings confirm the practical viability of ProvAuditChain for integration into automated CI/CD pipelines, providing a scalable foundation for trustworthy AI accountability in decentralized ecosystems.**

***Index Terms*—AI Audits, Blockchain, Decentralized storage, Gas efficiency, Layer 2, Provenance, Smart contract, Verifiable computing**

I. Introduction

The rapid growth of decentralized applications (dApps) has driven an unprecedented rise in the deployment of smart contracts, making blockchain ecosystems increasingly reliant on the correctness and security of these contracts [1]–[3]. However, traditional security auditing methods, which depend on manual expert review, cannot keep pace with the scale and speed of modern Web3 development [4]–[7]. This mismatch has created a critical bottleneck in the software development lifecycle, increasing the risk of deploying vulnerable contracts and undermining trust in the ecosystem [8]–[10].

To address the scalability challenge, AI-powered tools have emerged to automate smart contract security audits, offering faster and more consistent vulnerability detection [5], [6], [11]–[13]. Despite their promise, these AI-generated audit reports introduce a new trust gap: the provenance and integrity of the audit outputs are difficult to verify. Without cryptographic guarantees linking the audit report to the specific AI model and source code, stakeholders remain vulnerable to tampering or misrepresentation, limiting the adoption of fully automated security workflows.

Ensuring the verifiable provenance of AI-driven audit reports requires establishing a cryptographic link between the source code, the AI model version used for auditing, and the final audit output. We refer to this challenge as the audit provenance problem [14], which has seen little attention from traditional systems and research in the area of smart contract auditing. To close this gap, we propose ProvAuditChain, a hybrid on-chain/off-chain framework that guarantees the immutable and tamper-evident recording of audit provenance using gas-efficient smart contracts deployed on Layer 2 blockchain networks.

This paper makes three primary contributions. First, we design and implement ProvAuditChain, a modular and gas-efficient architecture that leverages an event-driven pattern and decentralized storage to minimize on-chain costs. Second, we provide the first public empirical benchmark of gas consumption, latency, and throughput for on-chain provenance recording of AI audit reports on the Arbitrum Sepolia testnet. Third, we discuss practical implications and optimizations for integrating ProvAuditChain into automated CI/CD pipelines, demonstrating its readiness for real-world adoption. The remainder of this paper is organized as follows: Section 2 reviews related work; Section 3 describes the ProvAuditChain framework; Section 4 outlines our evaluation methodology; Section 5 presents results; Section 6 discusses findings; and Section 7 concludes the paper.

II. Related works

A variety of AI-powered smart contract auditing tools have been developed to automate vulnerability detection, including Vulhunter [15], xFuzz [16], and MANDO-HGT [17], as well as recent approaches leveraging large language models (LLMs) [11], [13], [18]–[21]to generate security assessments. While these tools improve scalability and speed compared to manual reviews [6], they lack mechanisms to verify the authenticity and integrity of their outputs cryptographically. Current research on decentralized, tamper-evident audit trails demonstrates that cryptographic signing and blockchain-based

logging can provide such guarantees for automated systems [22]. Still, these approaches have not been integrated into mainstream AI-powered smart contract auditing tools. This limitation creates a risk that audit reports can be altered or misattributed, undermining confidence in their findings.

Blockchain technology has been widely employed to provide immutable provenance for assets such as supply chains and digital content, ensuring transparent and tamper-evident histories [23]. However, most existing provenance solutions focus on tracking physical goods or data lineage rather than the outputs of AI systems. The challenge of securely recording AI-generated audit reports on-chain remains largely unexplored, particularly in terms of minimizing gas costs and maintaining scalability on Layer 2 networks.

Prior efforts to bring audit information on-chain have typically focused on human-generated reports, bug bounty submissions, or upgrade governance logs, often using centralized oracles or off-chain attestations. These approaches fall short in offering full cryptographic traceability of AI audit pipelines and are rarely optimized for low-cost operation on Layer 2 environments. In contrast, ProvAuditChain introduces a purpose-built design that natively supports AI-generated audits, leverages decentralized storage, and emphasizes gas efficiency, bridging a gap left by existing solutions.

A key requirement for trustworthy AI audits is the ability to attribute the output to a specific AI model version under controlled conditions. Research on verifiable machine learning emphasizes techniques such as model fingerprinting, deterministic inference, and digital signatures over model checkpoints to support auditability [24]. However, these practices are not widely adopted in existing smart contract auditing pipelines, which typically treat the AI tool as a black box. Without a standardized way to bind audit results to specific models and inputs, claims of correctness or reproducibility remain unverifiable.

Advancements in verifiable computing and decentralized identity offer promising tools for establishing trust in AI-driven processes [25]. Protocols such as verifiable credentials, zero-knowledge proofs (ZKPs), and decentralized identifiers (DIDs) enable selective disclosure, reproducibility, and attribution without relying on centralized authorities [25], [26]. While these concepts have been explored in areas like identity management and privacy-preserving computation, they have yet to be applied comprehensively to the domain of smart contract audits. ProvAuditChain represents an early integration of these principles, specifically model-level authentication and hash-based report verification, into a coherent, developer-facing framework.

III. The ProvAuditChain Framework

ProvAuditChain adopts a hybrid on-chain/off-chain architecture designed to maximize verifiability while minimizing on-chain gas consumption. The core design principle is to perform computationally expensive and storage-intensive tasks off-chain, while anchoring essential integrity and provenance metadata on-chain using lightweight, modular smart contracts. This separation of concerns enables trust-minimized audit recording at low cost, while maintaining strong guarantees of tamper resistance and traceability.

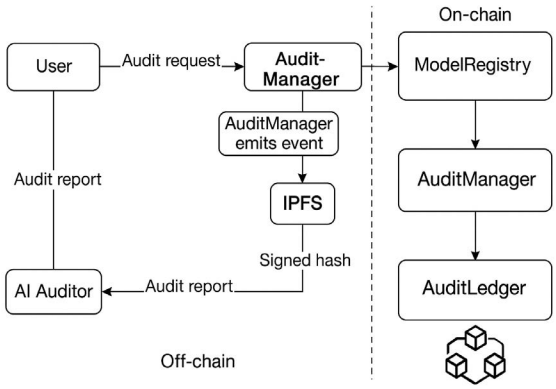

Fig. 1. System Architecture

Algorithm 1 ModelRegistry Contract: Register Model

```
procedure REGISTERMODEL(metadataHash)
    require(caller == owner)
    modelID ← keccak256(metadataHash, timestamp, caller)
    models[modelID] ← (metadataHash, active = true)
    emit ModelRegistered(modelID, metadataHash)
end procedure
```

The on-chain component of ProvAuditChain consists of three smart contracts: ModelRegistry, AuditManager, and AuditLedger. The ModelRegistry serves as an allowlist of authorized AI auditing models, each associated with a signing public key. This enables on-chain signature verification of audit outputs. The AuditManager is a minimal user-facing contract that receives audit requests and emits an AuditRequested event to trigger the off-chain process. The AuditLedger functions as an append-only registry of completed audits, storing signed hashes of audit reports and verifying their authenticity against the ModelRegistry.

Algorithm 2 AuditManager Contract: Request Audit

```
procedure REQUESTAUDIT(contractAddress, modelID)
    require(ModelRegistry.models[modelID].active)
    emit AuditRequested(contractAddress, modelID, msg.sender)
end procedure
```

The audit workflow begins when a user submits a smart contract address and a selected AI model identifier to the AuditManager, which emits an AuditRequested event. This event is picked up by an off-chain auditor service that performs the analysis using the specified model. Upon completion, the auditor stores the full audit report on IPFS and generates a cryptographic signature over the IPFS hash using the model's

registered private key. It then submits the signed hash to the AuditLedger, which verifies the signature against the corresponding entry in the ModelRegistry before permanently recording the audit result.

Algorithm 3 AuditLedger Contract: Record Verified Audit

```
procedure RECORDAUDIT(modelID, ipfsHash, signature)
    valid ← MODELREGISTRY.VERIFYSIGNATURE(modelID, ipfsHash, signature)
    if valid then
        audits.push((modelID, ipfsHash, signature, timestamp))
        emit AuditRecorded(modelID, ipfsHash)
    else
        revert("Invalid Signature")
    end if
end procedure
```

By cleanly separating responsibilities between the smart contracts and the off-chain audit engine, ProvAuditChain achieves a balance between trust, efficiency, and developer usability. The system is model-agnostic, allowing multiple AI tools to coexist under a single framework, and is compatible with any environment that supports Ethereum Virtual Machine (EVM) smart contracts. Its event-driven design facilitates easy integration with CI/CD pipelines, while the minimal gas footprint makes it suitable for routine use in resource-constrained L2 environments. This modular and extensible architecture lays a practical foundation for trust-minimized, verifiable AI auditing in decentralized ecosystems. Figure 1 gives an overview of the framework.

A. Implementation and Evaluation Methodology.

To evaluate the practical feasibility of ProvAuditChain, we deployed the system's smart contracts to the Arbitrum Sepolia testnet. This public Ethereum Layer 2 environment supports realistic gas modeling and throughput measurements. Our objective was to assess gas efficiency, latency, and scalability under continuous usage conditions representative of integration in an automated development pipeline. The evaluation focused on measuring the gas consumed by each contract function and the time taken to complete an entire audit cycle, from request submission to final on-chain recording.

We developed a custom benchmarking script in Node.js to simulate sustained audit activity and gather performance metrics. The script emulated 600 audit cycles at a constant rate of 20 requests per minute, modeling a realistic CI/CD deployment scenario. For each cycle, it triggered the requestAudit function on the AuditManager, awaited off-chain audit generation and IPFS upload, and then submitted the signed hash to the AuditLedger using the recordAudit function. Timestamps were logged at each stage to measure end-to-end cycle time, while transaction receipts were analyzed to extract gas consumption data.

Our evaluation captured three core metrics: gas consumption, latency, and throughput. Gas consumption was measured separately for the requestAudit and recordAudit functions to identify the cost of initiating and completing an audit. Latency was defined as the total time from the initial user request to the successful recording of the signed audit hash on-chain. Throughput was calculated as the number of complete audit cycles processed per minute under the given workload. Together, these metrics provide a comprehensive view of the system's economic and operational feasibility for integration into real-world development workflows.

The smart contracts were implemented in Solidity using the Hardhat development framework and deployed to the Arbitrum Sepolia testnet via its public RPC endpoint. Transactions were submitted from a MetaMask wallet connected to the testnet, enabling consistent gas benchmarking across all operations. The off-chain auditor service ran on a consumer-grade machine (quad-core CPU, 16 GB RAM), emulating a lightweight CI/CD integration scenario. While the evaluation was conducted under controlled network conditions, it does not account for concurrency, node instability, or adversarial interference. These aspects remain open for future exploration as part of a broader stress-testing strategy.

IV. Results and Discussion

Our benchmarking of ProvAuditChain on the Arbitrum Sepolia testnet, spanning 600 audit cycles, confirms its efficiency and stability for real-world applications. The system demonstrates a predictable and low-cost performance profile, making it highly suitable for integration into automated development pipelines. Table I provides a consolidated summary of the key performance indicators, which we analyze in detail throughout this section.

TABLE I
Arbitrum Sepolia Benchmark Metrics

Metric	Value
Avg. Gas per Request	32,796 units
Avg. Gas per Record	140,109 units
Total Gas per Audit	$\sim$173,000 units
Avg. Cycle Time	10.15 s
Throughput	5.9 audits/min

The metrics in Table I underscore the core value proposition of ProvAuditChain: delivering strong on-chain provenance guarantees at a trivial marginal cost. A total gas consumption of approximately 173,000 units per audit cycle on a Layer 2 network, such as Arbitrum, is exceptionally low, confirming the effectiveness of our hybrid architecture. This efficiency is paramount for adoption, as it allows development teams to log audit provenance for every code change or model update without incurring prohibitive costs. The throughput of nearly six audits per minute further demonstrates that the system can comfortably handle the demands of a typical, active CI/CD pipeline without creating a bottleneck.

Figure 2 illustrates the gas consumption for the two main on-chain interactions. The 'requestAudit' function is exceptionally lightweight, consuming an average of only 32,796

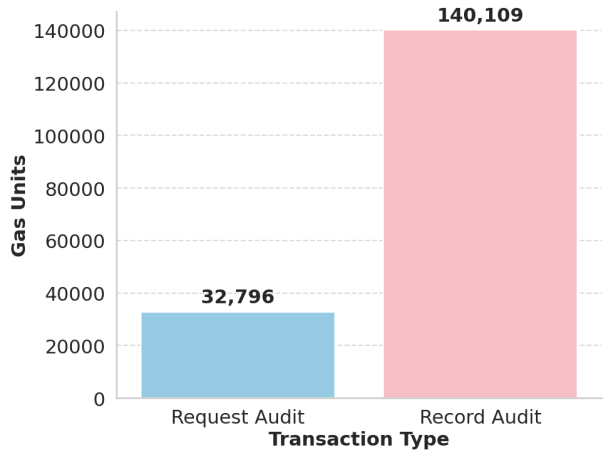

Fig. 2. Breakdown of average gas consumption for the two primary on-chain functions: 'requestAudit' and 'recordAudit'.

gas. This is expected, as its sole purpose is to emit an event—a highly gas-efficient operation for signaling off-chain systems. In contrast, the 'recordAudit' function requires an average of 140,109 gas. This higher cost is attributable to its more intensive workload, which includes cryptographic signature verification ('ecrecover') and, most significantly, a state-changing operation that writes the audit record to blockchain storage. Although more expensive than the other, this gas expenditure remains modest for an L2 transaction. It validates our design choice to store only a lightweight hash on-chain, while delegating the full report to IPFS. This architectural decision is fundamental to the system's economic viability.

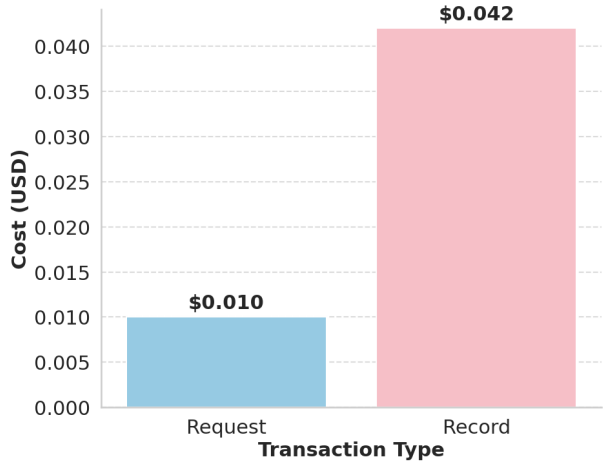

Fig. 3. Estimated transaction cost in USD, based on a representative L2 gas price of 0.1 Gwei and an ETH price of $3,000.

The practical economic impact is visualized in Figure 3. At a total cost of approximately $0.05 USD per audit, ProvAuditChain transforms provenance from a costly, occasional luxury into a routine, low-friction component of the development lifecycle. This affordability enables teams to adopt a "provenance-as-code" paradigm, where every commit in a repository can be automatically audited and its result immutably recorded without budgetary concerns. This capability represents a significant step forward from traditional, expensive manual audits, providing a level of continuous, granular assurance that was previously unattainable.

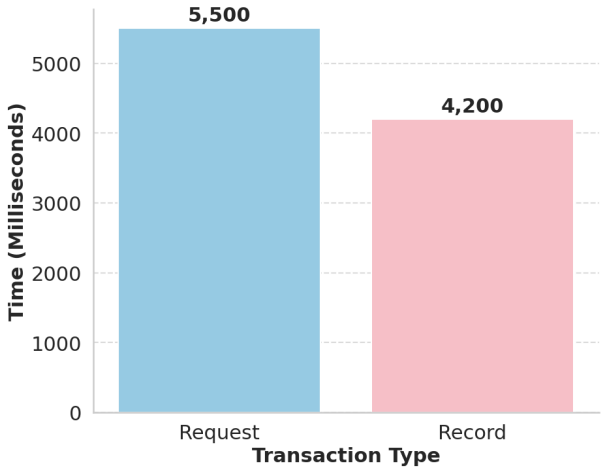

Fig. 4. End-to-end latency for a complete audit cycle, from initial request to the final on-chain record.

As shown in Figure 4, the average end-to-end cycle time was 10.15 seconds. This latency encompasses L2 transaction confirmation time for both the request and record transactions, as well as the off-chain processing time (AI model execution and IPFS upload). For an asynchronous process within a CI/CD pipeline, this latency is well within acceptable limits. Developers typically do not require synchronous feedback, and a 10-second delay for a permanent, cryptographic record of an audit is a negligible trade-off. While the current throughput of 6 audits/minute is sufficient for many use cases, it could be further optimized for high-frequency environments by batching multiple audit records into a single 'recordAudit' transaction. Such an enhancement could dramatically increase throughput while further reducing the amortized gas cost per audit.

A. Security model, auditor compromise, and mitigations

In our prototype, the AI Auditor signs the report's IPFS CID; contracts verify the signature against the model's registered verifier and active status, and reject duplicates for idempotent retries. As potential mitigations to reduce reliance on a single key, deployments may use an EIP-1271 multisig verifier, adopt epoch-based key rotation with a revocation threshold, and run the signer in a trusted execution environment with optional attestation in the metadata.

B. Operational conditions: congestion, instability, and adversaries

Under high L2 load, inclusion latency and fees rise, increasing time-to-confirmation and audit cost; adversarial flooding or soft censorship primarily degrades latency rather than integrity. Our client logic is resilient via idempotent submission. As

potential operational mitigations, deployments can set adaptive fee caps, batch multiple CIDs when permissible, use multi-provider failover with bounded retries and backoff, and restrict submissions to the registered verifier with rate caps.

C. Assumptions and defended surface

We assume standard cryptographic primitives (ECDSA, keccak256), collision-resistant IPFS addressing for the chosen CID format, eventual L1 finality for the rollup, audited contract deployments, and loosely synchronized clocks for worker timeouts. Under these assumptions, integrity is enforced on-chain through verifier checks, model-activation status, and duplicate rejection, while liveness and robustness to network variance are provided off-chain by idempotent retries, failover, and fee control.

V. Conclusion

ProvAuditChain presents a novel, gas-efficient framework for recording the provenance of AI-driven smart contract audits on public Layer 2 networks. Our modular architecture leverages event-driven on-chain contracts combined with off-chain auditing and decentralized storage to deliver tamper-evident, verifiable audit trails at a minimal cost of approximately $0.05 per audit. Evaluations on the Arbitrum Sepolia testnet demonstrate that ProvAuditChain achieves stable gas usage, acceptable latency, and throughput suitable for integration into automated CI/CD workflows. This work lays foundational infrastructure for trustworthy AI accountability in decentralized development environments, paving the way for further innovations in verifiable AI and secure blockchain applications.

Acknowledgment

This work was partly supported by Innovative Human Resource Development for Local Intellectualization program through the IITP grant funded by the Korea government(MSIT) (IITP-2025-RS-2020-II201612, 33%) and by Priority Research Centers Program through the NRF funded by the MEST(2018R1A6A1A03024003, 33%) and by the MSIT, Korea, under the ITRC support program(IITP-2025-RS-2024-00438430, 34%).

References

[1] G. Iuliano and D. D. Nucci, "Smart contract vulnerabilities, tools, and benchmarks: An updated systematic literature review," *arXiv preprint arXiv:2412.01719*, 2024.

[2] P. Qian, R. Cao, Z. Liu, W. Li, M. Li, L. Zhang, and Y. Xu, "Empirical review of smart contract and defi security: Vulnerability detection and automated repair," *arXiv preprint arXiv*, 2023.

[3] H. Key, "Automated reasoning in blockchain: Foundations, applications, and frontiers," *arXiv preprint arXiv:2503.20461*, 2025.

[4] L. Zhou, K. Qin, D. Song, L. Cavallaro, and A. Gervais, "Do you still need a manual smart contract audit?" *arXiv preprint arXiv*, 2023.

[5] Z. Wei, J. Sun, Z. Zhang, X. Zhang, M. Li, and Z. Hou, "Llm-smartaudit: Advanced smart contract vulnerability detection," *arXiv preprint arXiv*, 2024.

[6] Z. Wei, J. Sun, Z. Zhang, X. Zhang, and M. Li, "Ftsmartaudit: A knowledge distillation-enhanced framework for automated smart contract auditing using fine-tuned llms," *arXiv preprint arXiv*, 2024.

[7] M. Bouafif, C. Zheng, I. Qasse, and E. Zulkoski, "A context-driven approach for co-auditing smart contracts with the support of gpt-4 code interpreter," *arXiv preprint arXiv*, 2024.

[8] M. Lewis, "Architectural design for secure smart contract development," *Human Factors in Cybersecurity*, p. 121, 2023.

[9] K. Li, Y. Xue, S. Chen, H. Liu, K. Sun, M. Hu, H. Wang, Y. Liu, and Y. Chen, "Static application security testing (sast) tools for smart contracts: How far are we?" *Proceedings of the ACM on Software Engineering*, vol. 1, no. FSE, p. 1447–1470, Jul 2024.

[10] Z. Wei, J. Sun, Z. Zhang, X. Zhang, X. Yang, and L. Zhu, "Survey on quality assurance of smart contracts," *ACM Comput. Surv.*, vol. 57, no. 2, Oct 2024.

[11] J. Kevin and P. Yugopuspito, "Smartllm: Smart contract auditing using custom generative ai," in *2025 International Conference on Computer Sciences, Engineering, and Technology Innovation (ICoCSETI)*, 2025, pp. 260–265.

[12] M. Khodadadi and J. Tahmoresnezhad, "Hymo: Vulnerability detection in smart contracts using a novel multi-modal hybrid model," *arXiv preprint arXiv:2304.13103*, 2023.

[13] O. Zaazaa and H. El Bakkali, "Smartllmsentry: A comprehensive llm-based smart contract vulnerability detection framework," *Journal of Metaverse*, vol. 4, no. 2, p. 126–137, 2024.

[14] A. A. Battah, M. M. Madine, H. Alzaabi, I. Yaqoob, K. Salah, and R. Jayaraman, "Blockchain-based multi-party authorization for accessing ipfs encrypted data," *IEEE Access*, vol. 8, pp. 196 813–196 825, 2020.

[15] Z. Li, S. Lu, R. Zhang, Z. Zhao, R. Liang, R. Xue, W. Li, F. Zhang, and S. Gao, "Vulhunter: Hunting vulnerable smart contracts at evm bytecode-level via multiple instance learning," *IEEE Transactions on Software Engineering*, vol. 49, no. 11, pp. 4886–4916, 2023.

[16] Y. Xue, J. Ye, W. Zhang, J. Sun, L. Ma, H. Wang, and J. Zhao, "xfuzz: Machine learning guided cross-contract fuzzing," *IEEE Transactions on Dependable and Secure Computing*, vol. 21, no. 2, pp. 515–529, 2024.

[17] H. H. Nguyen, N.-M. Nguyen, C. Xie, Z. Ahmadi, D. Kudendo, T.-N. Doan, and L. Jiang, "Mando-hgt: Heterogeneous graph transformers for smart contract vulnerability detection," in *2023 IEEE/ACM 20th International Conference on Mining Software Repositories (MSR)*, 2023, pp. 334–346.

[18] J. Yu, "Retrieval augmented generation integrated large language models in smart contract vulnerability detection," *arXiv preprint arXiv:2407.14838*, 2024.

[19] S. Hu, T. Huang, F. İlhan, S. F. Tekin, and L. Liu, "Large language model-powered smart contract vulnerability detection: New perspectives," in *2023 5th IEEE International Conference on Trust, Privacy and Security in Intelligent Systems and Applications (TPS-ISA)*, 2023, pp. 297–306.

[20] W. Ma, D. Wu, Y. Sun, T. Wang, S. Liu, J. Zhang, Y. Xue, and Y. Liu, "Combining fine-tuning and llm-based agents for intuitive smart contract auditing with justifications," 2024.

[21] 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 *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering*, ser. ICSE '24. New York, NY, USA: Association for Computing Machinery, 2024.

[22] V. Kulothungan, "Using blockchain ledgers to record ai decisions in iot," *IoT*, vol. 6, no. 3, 2025.

[23] M. A. Alqarni, M. S. Alkatheiri, S. H. Chauhdary, and S. Saleem, "Use of blockchain-based smart contracts in logistics and supply chains," *Electronics*, vol. 12, no. 6, 2023.

[24] M. R. Swamy, P. Vijayalakshmi, and V. Rajendran, "Signature verification based on machine learning and deep learning techniques: A review," in *AIP Conference Proceedings*, vol. 3162, no. 1. AIP Publishing LLC, 2025, p. 020042.

[25] C. Brunner, U. Gallersdörfer, F. Knirsch, D. Engel, and F. Matthes, "Did and vc: Untangling decentralized identifiers and verifiable credentials for the web of trust," in *Proceedings of the 2020 3rd International Conference on Blockchain Technology and Applications*, ser. ICBTA '20. New York, NY, USA: Association for Computing Machinery, 2021, p. 61–66.

[26] G. Cohen, M. Sporny, M. Jones, and I. Herman, "Verifiable credentials data model v2.0," *W3C, Candidate Recommendation*, 2025.