Does 'loss of provenance' in AI-generated patches fundamentally denote the erosion of traceable reasoning chains linking code changes to architectural intent and dependency graphs, or does it primarily signify the absence of cryptographic verification tying commits to specific human or agent identities?

July 15, 2026 | SnugLab Research

Executive Summary

"Loss of provenance" in AI-generated patches fundamentally denotes both the erosion of traceable reasoning chains linking code changes to architectural intent and dependency graphs, and the absence of cryptographic verification tying commits to specific human or agent identities. While cryptographic verification provides a foundational trust layer for identity and tamper resistance [4, 11], the active erosion of semantic reasoning chains is the primary driver of provenance loss, leading to structural regressions and "intent drift" that cryptographic identity alone cannot capture [3, 7, 11, 12]. Robust provenance solutions require the interdependent application of both mechanisms to ensure comprehensive auditability and integrity [3, 11, 12].

Key Findings

Loss of Provenance as Two Distinct Failure Modes

"Loss of provenance" in AI-generated patches is not a single unified breakdown but rather two distinct failure modes that require complementary diagnostic frameworks [3, 11].

The first mode, **logical lineage erosion**, involves the breakdown of causal links between code changes, architectural intent, and dependency graphs [3, 11, 14, 18]. This erosion obscures the rationale behind changes and complicates dependency mapping [3, 14, 18]. Diagnostic frameworks for this mode focus on capturing semantic lineage through Software Bills of Materials (SBOMs), knowledge graphs, and hash-chained audit trails that track model updates and unlearning operations [2, 11, 14, 18].

The second mode, **identity verification gaps**, severs the binding between commits and specific human or agent identities [1, 4, 7, 15]. This absence enables spoofed authorship

and unverified delegation by autonomous AI agents [4, 7, 16]. Diagnostic frameworks for this mode utilize next-generation signing platforms, cryptographic workload identities, and content provenance standards to cryptographically bind every commit to a verified corporate identity [1, 4, 7, 10, 15, 16]. These two dimensions are complementary pillars of provenance, with cryptographic identity verification providing the foundational trust layer for the integrity of reasoning chains [4].

Erosion of Reasoning Chains as the Primary Driver

The erosion of traceable reasoning chains actively drives the loss of provenance in AI-generated patches, functioning as a primary cause rather than a mere symptom of missing cryptographic identity verification [3, 7, 11, 12]. AI models, relying on pattern-matching within ephemeral context windows, create a "context gap" that obscures underlying design rationale and dependency lineage [3, 8, 13]. This directly introduces structural errors, such as silent architectural drift, where code passes automated tests but diverges from its original design [3, 5, 13]. While cryptographic identity verification is essential for establishing accountability and preventing malicious injection, verifying the author of a commit does not inherently capture the semantic rationale or dependency mappings behind the change [4, 7, 12, 15]. Historical evidence suggests that silent architectural drift from lost reasoning chains is more costly and persistent to remediate than identity spoofing [3, 7].

Insufficiency of Cryptographic Verification Alone

Cryptographic verification of human and agent identities is a foundational requirement for establishing the "who" and "when" of code changes, but it is insufficient on its own to ensure robust provenance [4, 7]. Relying solely on cryptographic identity binding inherently fails to capture the semantic reasoning and architectural intent behind the patches [3, 8, 11, 12]. Robust provenance requires traceable reasoning chains that systematically link code changes to dependency graphs, build metadata, and the underlying design rationale [3, 11, 14, 18]. Generative AI tools blur the lines of code origin and lack native semantic awareness of data transformations [3, 8]. Therefore, cryptographic identity verification must be paired with mechanisms like agentic provenance graphs and dependency mapping to capture the "why" behind the changes and prevent silent architectural drift [3, 11, 12, 18].

Interdependence of Provenance Mechanisms

Traceable reasoning chains and cryptographic identity verification are interdependent, not independent mechanisms [3, 11]. Cryptographic verification serves as the foundational trust layer that secures the integrity and reliability of broader reasoning chains, such as dependency graphs and architectural intent envelopes [11]. In AI-driven workflows, traceable reasoning chains capture the lineage of thought and dependencies, often structured as agentic provenance graphs or hash-chained audit trails [2, 12]. For these chains to provide meaningful auditability, they must be cryptographically signed to bind changes to specific human or agent identities [4, 7]. Cryptographic methods, including digital signatures, hash chains, and attestations, ensure that the recorded lineage remains unaltered and authentic [1, 2, 4, 7, 10, 11]. Thus, cryptographic signing guarantees the trustworthiness of the data, while intact reasoning chains provide the semantic context and architectural rationale that make the signed data meaningfully auditable [3, 11, 12].

Tools and Standards for Provenance Implementation

Complementary diagnostic frameworks for provenance utilize various tools and standards:

- **For Traceable Reasoning Chains**: The C2PA Standard encodes provenance metadata through cryptographically signed manifests [9, 10, 13, 19]. Software Bills of Materials (SBOMs) and AI Bills of Materials (AIBOMs) map dependencies and model components [11, 18]. Knowledge Graphs (KGs) organize structured information for dependency analysis [14]. Hash-chained audit trails, such as those in AuditableLLM, record model updates and unlearning operations [2]. Agentic Provenance Graphs capture verifiable lineages of thought from initial goals to final artifacts [12]. Zero-Knowledge Proofs (ZKPs) enable verifiable AI pipelines, including Proofs of Training (ZKPoTs) and Proofs of Inference (ZKPoIs) [19].
- **For Cryptographic Identity Verification**: Sigstore and Open PubKey are next-generation signing platforms that link public keys to trusted OpenID Connect identities [1]. SPIFFE and SPIRE provide unique, short-lived cryptographic identities (SVIDs) to workloads and AI agents [15, 16]. The Supply-chain Levels for Software Artifacts (SLSA) framework demands verifiable proof of authorship [4, 7, 11].

Adoption and Costs: The C2PA standard has over 200 members, including major tech companies like Adobe, Microsoft, Google, Intel, OpenAI, and Meta [19]. A valid signing certificate costs approximately $289 per year [19]. AI now generates or assists with 41% of all code, with 82% of developers using AI tools weekly; 25% of Google's code and 30%

of Microsoft's code are AI-written [4, 7]. Regulatory drivers like the EU AI Act (effective August 2026) and NIST AI 100-4 (November 2024) are accelerating adoption [10, 11].

Quantifiable Erosion and Gaps

While specific benchmarks for architectural intent loss or direct correlations between regression bugs and missing cryptographic verification are not established, quantifiable metrics demonstrate the erosion of traceable reasoning chains and the prevalence of identity gaps. AI tools assist with 41% of all code, with 25% of Google's code and 30% of Microsoft's code being AI-written [4, 7]. This scale increases code velocity and expands the attack surface at the commit stage [4, 7]. The erosion of traceable reasoning is compounded by the proliferation of non-human identities, which outnumber human users 45 to 1 on average and reach 144 to 1 in cloud-native environments [16]. Research indicates that 97% of these non-human identities carry excessive privileges, complicating accountability [19]. AI agents fail approximately 70% of the time on multi-step tasks [26]. An analysis of 33,596 agent-authored pull requests showed a 71.48% overall merge rate, with OpenAI Codex achieving 82.59% and Copilot 43.04% [24]. Context degradation can cause significant accuracy drops, with some models experiencing substantial performance declines on longer prompts [25].

Successful Implementations and Measurable Improvements

Several frameworks and projects have successfully integrated traceable reasoning chains with cryptographic verification for AI-generated code:

- **AuditableLLM** implements hash-chain-backed, tamper-evident audit trails to record model updates and code generation steps, enhancing third-party verification of AI outputs [2].
- **Agentic Provenance Graphs** capture a verifiable lineage of thought from initial goals to final code artifacts, with each agent appending trace nodes containing its ID, execution context, result hash, cryptographic signature, and parent lineage to form a Directed Acyclic Graph (DAG) [12]. This structure ensures full auditability and reproducibility [12].
- **Enterprise Adoption** by Google and Microsoft, where 25% and 30% of their code is AI-written respectively, involves implementing cryptographic identity verification using tools like Sigstore and the C2PA standard [4, 7, 10]. This binds every commit to a verified corporate identity or agent workload, ensuring traceable provenance at scale [1, 4, 7].

Measurable improvements include enhanced auditability through cryptographically proven code changes [2, 12], and efficiency gains from AI-driven verification agents that automate tasks like regression triage and root-cause analysis, reducing manual error and schedule delays [19]. On-device cryptographic verification frameworks demonstrate negligible latency, completing provenance manifest validation and hash computations in under 500 ms with peak memory usage below 45 MB [19].

Remediation Costs of Architectural Drift vs. Identity Spoofing

Silent architectural drift due to lost reasoning chains has been documented in enterprise and open-source projects. Apiiro's Deep Code Analysis found AI-assisted developers increased architectural design flaws by 153% and privilege escalation paths by 322% across Fortune 50 enterprises between December 2024 and June 2025 [3, 6]. An OpenAI internal project experienced "drift" and "entropy" in AI-generated code, requiring 20% of the team's week for cleanup [21]. Remediation costs for architectural drift accumulate as chronic technical debt; fixing AI-generated code took three times as long as human-written code in a 2024 enterprise case study [22]. For a 200-engineer organization, avoidable rework translates to a $3.2 million annual velocity loss [23]. Compounded debt can lead to massive system meltdowns, such as Southwest Airlines' 2022 incident costing $750 million [20]. In contrast, identity spoofing incidents incur acute, immediate financial penalties, with breaches involving AI-generated logic costing between $4 million and $9 million per incident [22]. Unpatched flaws result in average compliance fines of $500,000 per month [22]. While the sources do not provide a direct ratio, architectural drift incurs hidden, accumulating operational expenses that degrade long-term system stability, whereas identity spoofing causes abrupt, high-impact financial and reputational damage [2, 12, 14, 17].

Implications

The findings imply that organizations must adopt a dual-pronged strategy to address provenance in AI-generated patches. Relying solely on cryptographic identity verification, while crucial for security and accountability, will not prevent the insidious and costly problem of architectural drift and "intent drift" caused by the erosion of reasoning chains [3, 7, 11, 12]. The increasing volume of AI-generated code and the proliferation of non-human identities necessitate robust, automated systems that can track both the "who" and the "why" of code changes [4, 7, 16]. Failure to implement comprehensive

provenance solutions will lead to significant technical debt, increased verification overhead, and persistent vulnerabilities that are difficult and expensive to remediate [19, 23]. The interdependence of these two provenance dimensions means that cryptographic signing should be applied to secure the integrity of the reasoning chains themselves, ensuring that the semantic context and architectural rationale are trustworthy and auditable [3, 11, 12].

Limitations and Caveats

The research findings, while comprehensive, do not provide direct comparative benchmarks for the frequency and integration failure rates of reasoning chain erosion versus cryptographic identity gaps. Specific quantifiable thresholds for the percentage of architectural intent lost or a direct correlation between regression bugs and missing cryptographic verification are also not established. While costs for architectural drift and identity spoofing are provided, a direct ratio comparing their remediation expenses is not available. Some specific numerical claims regarding AI model accuracy drops on long prompts were based on blog sources, and independent verification was silent on the exact figures, necessitating a softening of those claims.

Sources

[1] [preprint] Html - arxiv.org - AUTHORS UNAVAILABLE - https://arxiv.org/html/2407.03949v1

[2] [peer-reviewed] pdfs.semanticscholar.org - AUTHORS UNAVAILABLE - https://pdfs.semanticscholar.org/0474/5e98b699f7c7f66cd92cf0dca7989e1e36cf.pdf

[3] A Primer On Provenance - cacm.acm.org - https://cacm.acm.org/practice/a-primer-on-provenance/

[4] Code Provenance Is The Missing Control For Ai Generated Comm - nhimg.org - https://nhimg.org/articles/code-provenance-is-the-missing-control-for-ai-generated-commits/

[5] [preprint] Delivery.Cfm - papers.ssrn.com - AUTHORS UNAVAILABLE - https://papers.ssrn.com/sol3/Delivery.cfm/6430238.pdf?abstractid=6430238&mirid=1

[6] Post - moltbook.com - https://www.moltbook.com/post/ebdca9a2-8444-4829-8290-361c311b5203

[7] Why Is Code Provenance Non Negotiable In The Age Of Ai - beyondidentity.com - https://www.beyondidentity.com/resource/why-is-code-provenance-non-negotiable-in-the-age-of-ai

[8] Understanding Code Provenance - fortegrp.com - https://fortegrp.com/insights/understanding-code-provenance

[9] [blog] Digital Authenticity Provenance And Verification In Ai Gener - numbersprotocol.io - https://numbersprotocol.io/blog/digital-authenticity-provenance-and-verification-in-ai-generated-media

[10] Building The Verification Layer A Developers Guide To Crypto - dev.to - https://dev.to/veritaschain/building-the-verification-layer-a-developers-guide-to-cryptographic-ai-provenance-4f4f

[11] Software Provenance - jfrog.com - https://jfrog.com/learn/grc/software-provenance/

[12] [blog] Provenance As The Chain Of Accountability - engineeringagents.substack.com - https://engineeringagents.substack.com/p/provenance-as-the-chain-of-accountability

[13] [blog] Toward Reliable Provenance In Ai Generated Content Text Imag - medium.com - https://medium.com/@adnanmasood/toward-reliable-provenance-in-ai-generated-content-text-images-and-code-9ebe8c57ceae

[14] [preprint] Html - arxiv.org - AUTHORS UNAVAILABLE - https://arxiv.org/html/2405.20455v3

[15] Securing The Future Rethinking Identity For Ai Agents And No - nhimg.org - https://nhimg.org/community/agentic-ai-and-nhis/securing-the-future-rethinking-identity-for-ai-agents-and-non-human-identities/

[16] Csa Whitepaper Nonhuman Identity Agentic Ai Governance V1 Cs - labs.cloudsecurityalliance.org - https://labs.cloudsecurityalliance.org/research/csa-whitepaper-nonhuman-identity-agentic-ai-governance-v1-cs/

[17] Rec6VhmFPQLkpCrOb - sparai.org - https://sparai.org/projects/sp26/rec6VhmFPQLkpCrOb/

[18] [blog] Software Dependency Graph - puppygraph.com - https://www.puppygraph.com/blog/software-dependency-graph

[19] [preprint] Html - arxiv.org - AUTHORS UNAVAILABLE - https://arxiv.org/html/2503.22573v1

[20] [edu] The Hidden Costs Of Coding With Generative Ai - sloanreview.mit.edu - https://sloanreview.mit.edu/article/the-hidden-costs-of-coding-with-generative-ai/

[21] Harness Engineering - openai.com - https://openai.com/index/harness-engineering/

[22] [blog] Agentic Remediation The New Control - softwareanalyst.substack.com - https://softwareanalyst.substack.com/p/agentic-remediation-the-new-control

[23] [blog] Technical Debt Quantification Business Case - connectory.ai - https://www.connectory.ai/blog/technical-debt-quantification-business-case/

[24] [preprint] Html - arxiv.org - AUTHORS UNAVAILABLE - https://arxiv.org/html/2601.15195v1

[25] Context Rot The Emerging Challenge - understandingai.org - https://www.understandingai.org/p/context-rot-the-emerging-challenge

[26] Ai Agents Wrong 70 Of Time Carnegie Mellon Study - theregister.com - https://www.theregister.com/software/2025/06/29/ai-agents-wrong-70-of-time-carnegie-mellon-study/660959