

Automated Code Provenance via Input Telemetry Analysis

Ramprasad Anandam Gaddam
Vouch Protocol Project
<https://vouch-protocol.com>
ram@vouch-protocol.com

January 2026

Abstract

This paper presents a system for automatically classifying code provenance—distinguishing human-authored code from AI-generated code—based on input telemetry analysis. By monitoring keystroke velocity, paste events, and completion triggers within development environments, the system creates cryptographically signed provenance maps without requiring manual tagging. We formalize the classification model, define the telemetry signals and decision boundaries, specify the attestation format, and analyze privacy properties. The system addresses growing regulatory and compliance needs for AI contribution transparency in software development.

This paper is part of the Vouch Protocol Defensive Disclosure Series. Full index: <https://vouch-protocol.com/docs/disclosures/>

Keywords: code provenance, keystroke dynamics, AI-generated code, telemetry, automated attestation

1 Introduction

The proliferation of AI coding assistants—GitHub Copilot, ChatGPT, Claude—has transformed software development. Code is increasingly co-authored by humans and AI. However, current version control systems attribute all code in a commit to the human committer, creating a “provenance gap.”

This gap has significant implications:

- **Legal:** Intellectual property ownership of AI-generated code remains unsettled
- **Compliance:** Regulated industries may require human code authorship
- **Quality:** Review processes may differ for AI-generated vs. human code
- **Transparency:** Users deserve to know the composition of code they run

We present “Ghost Signature,” a system that automatically classifies code origin based on input patterns and produces cryptographic attestations of provenance.

1.1 Contributions

- Classification model for human vs. AI-generated code based on input telemetry
- Telemetry signal specification and decision boundaries
- Cryptographically signed provenance attestation format
- Privacy-preserving local processing architecture

2 Background

2.1 Keystroke Dynamics

Definition 1 (Keystroke Velocity). *Keystroke velocity v is the rate of character input:*

$$v = \frac{\Delta \text{characters}}{\Delta \text{time}} \quad (1)$$

measured in characters per second (cps).

Human typing exhibits characteristic velocity ranges (40-100 cps) with variance, pauses, and correction patterns [Bonneau and Preibusch, 2010].

2.2 AI Completion Patterns

AI-generated code insertions exhibit distinct patterns:

- Large blocks appear instantaneously (high velocity)
- No backspace/correction sequences within the block
- Often triggered by explicit accept events (Tab, Enter)

3 Classification Model

3.1 Telemetry Signals

The system monitors the following signals:

Definition 2 (Input Event). *An input event $e = (t, \tau, c, m)$ where:*

- t : timestamp
- τ : event type $\in \{\text{keystroke}, \text{paste}, \text{completion}, \text{delete}\}$
- c : character count
- m : metadata (e.g., source)

3.2 Decision Boundaries

Pattern	Classification	Criteria
$v < 15$ cps, backspaces present	human	Natural typing
$v > 50$ cps, completion event	ai_assisted	Copilot/completion
$v > 200$ cps, paste event	synthetic	Pasted from AI
Mixed patterns	mixed	Human edits to AI base

3.3 Region Segmentation

Code is segmented into regions with consistent classification:

Definition 3 (Provenance Region). *A region $R = (l_s, l_e, o, \gamma)$ where:*

- l_s, l_e : Start and end line numbers
- o : Origin classification
- γ : Confidence score $\in [0, 1]$

4 Attestation Format

4.1 Provenance Map Schema

```
1 {  
2   "version": 1,  
3   "file": "src/main.py",  
4   "timestamp": "2026-01-10T15:30:00Z",  
5   "signer": "did:web:alice.example.com",  
6   "regions": [  
7     {"lines": "1-25", "origin": "human",  
8      "confidence": 0.95},  
9     {"lines": "26-75", "origin": "ai-assisted",  
10      "confidence": 0.99, "model_hint": "copilot"},  
11     {"lines": "76-80", "origin": "human",  
12      "confidence": 0.92}  
13   ],  
14   "signature": "eyJhbGciOiJIJFZERTQ5Sj9..."  
15 }
```

4.2 Attachment Methods

Ghost signatures can be attached via:

- Git notes: `git notes add -ref=provenance`
- Sidecar files: `.vouch/main.py.provenance.json`
- Commit trailers: `Provenance-Signature: base64...`

5 Privacy Analysis

Property 1 (Local Processing). *All telemetry analysis occurs locally within the IDE extension. Raw keystroke data is never transmitted.*

Property 2 (Minimal Attestation). *The attestation contains only region classifications and line numbers—no content, no timing data.*

Opt-out: Users can disable telemetry collection. Code is then marked `origin: unknown`.

6 Security Analysis

Property 3 (Signer Accountability). *The provenance map is signed by the developer, creating non-repudiable attestation of classification.*

Threat: Evasion. A developer could simulate slow typing of AI-generated code. Mitigation: This requires deliberate effort; the system provides best-effort classification, not forensic-grade evidence.

7 Related Work

Software Composition Analysis (SCA) identifies third-party dependencies; our work addresses first-party code provenance.

Keystroke Dynamics for Authentication [Bonneau and Preibusch, 2010] uses typing patterns for identity verification; we adapt these signals for origin classification.

AI Detection Tools (GPTZero, etc.) analyze content; our approach uses behavioral signals, which are orthogonal and complementary.

8 Implementation

Reference implementation available at <https://github.com/vouch-protocol/vouch>.

IDE extensions available for:

- Visual Studio Code
- Cursor
- JetBrains IDEs

9 Conclusion

This paper presented Ghost Signature, a system for automated code provenance classification based on input telemetry. By analyzing keystroke dynamics and completion events, the system distinguishes human-authored from AI-generated code and produces cryptographic attestations. The approach enables transparency and compliance in AI-assisted software development while preserving developer privacy.

Prior Art Declaration

This document is published as a defensive prior art disclosure under the Creative Commons CC0 1.0 Universal Public Domain Dedication. The methods and systems described herein are hereby released into the public domain to prevent patent monopolization.

Any party implementing similar functionality after the publication date of this document cannot claim novelty for patent purposes.

Reference Implementation: <https://github.com/vouch-protocol/vouch>

Cryptographically Signed Document

Signed by: Ramprasad Anandam Gaddam (github:rampyg)

Verify: <https://v.vouch-protocol.com/p/pad007>

This document's authenticity can be verified by computing its SHA-256 hash and checking against the signature registered at the verification URL above.

References

Joseph Bonneau and Sören Preibusch. The Password Thicket: Technical and Market Failures in Human Authentication on the Web. In *Workshop on the Economics of Information Security (WEIS)*, 2010.