SWARMBLAME: CONTENT-ONLY ATTRIBUTION FOR ADVERSARIAL AGENT SWARMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Coding agents increasingly work as swarms that edit one shared repository, so an operator who finds a planted bug or a covert message must determine which agent wrote it. Provenance records such as logs and version history can be rewritten by the suspects and detach from an artifact once it is copied, whereas a watermark puts the evidence into the text each agent generates, under a secret key. A core requirement of attribution is a *framing bound*, which states that an audit wrongly names an innocent agent only with a small, fixed probability. However, re-testing hundreds of watermark keys while an auditor reads file by file breaks this bound, a rewrite by another agent erases the author's mark, and one short artifact carries too little evidence. We propose SWARMBLAME, an attribution protocol in which every agent samples under its own key with the standard KGW watermark and an anytime-valid test over all keys adds up evidence across files, naming an agent as soon as the evidence suffices while bounding framing for any reading order and stopping time. Composed keys record the chain of agents that rewrote a file, and seen-pair exclusion blocks framing by marks stolen from a victim's text. Across twenty models and six prior schemes, sixteen synthetic pull requests name their author in 99.9% of audits with 7 false accusations in 21,000, whereas re-testing detectors frame innocents in up to 8.93%, and optional learned weights nearly double code attribution without the prompt at audit time.

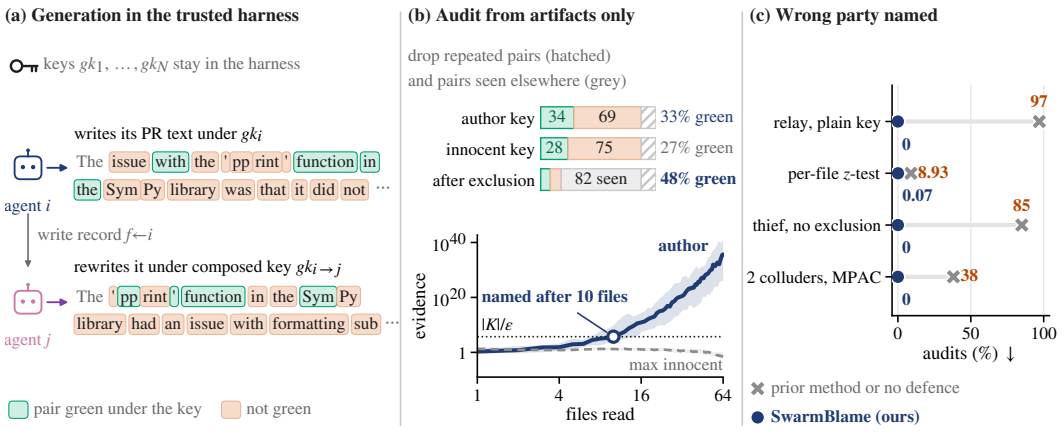


Figure 1: **One file is too weak, but a repository names its author without framing innocents.** (a) The trusted harness keys every sampler: agent i's real PR text under gk_i, and agent j's rewrite of it under the composed key $gk_{i \to j}$. (b) The auditor sees only artifacts: one text barely separates the author's key from an innocent key (chance is 25%), seen-pair exclusion drops pairs a thief could have read, and evidence summed over files crosses $|K|/\varepsilon$. (c) Audits naming the wrong party.

1 INTRODUCTION

Coding agents increasingly work as swarms that plan, write, review and commit to one repository, and one agent often rewrites another's file. When shared artifacts carry planted bugs or a covert

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

message, the operator of the swarm must know which agent produced them, as when agents evaluated by one provider coordinated through the directory names of an internal registry and attempted to tamper with their own activity logs (OpenAI, 2026). Such swarms need *attribution*, which names the *author* of a part of the repository, the agent whose model output produced it, so that the operator can hold that agent responsible and trace the chain of agents that handled a file. An *audit* performs attribution, as it reads a *bundle* of artifacts and names one agent, a chain of agents, or abstains.

Attribution is hard because one short artifact, such as a commit message, carries far less evidence than singling out one of hundreds of agents requires, so an audit must combine evidence over many files. Provenance records, such as logs, version histories and write records, can supply attribution, but they are large, spread across the version-control system, the tools and the *harness*, the orchestration software that calls the model for every agent and writes its outputs to the repository, and they no longer describe an artifact once it is copied, merged or rewritten.

Watermarks put evidence, a *mark*, into the artifact itself by biasing generation toward tokens that a secret key marks, and a detector with the key tests for this bias (Kirchenbauer et al., 2023; Dathathri et al., 2024). However, these detectors test one key on one text of fixed length, so re-testing hundreds of keys after every file names innocent agents, which we call *framing*, far above the intended rate.

Multi-bit watermarks write an agent's index into one shared key and name a third agent when two agents mix their texts (Yoo et al., 2024), and style classifiers name whichever agent an impostor imitates. The closest work traces one text to one of many adaptive users (Cohen et al., 2025) or the segments of one final text to cooperative agents (Nian et al., 2026), whereas swarm agents read and rewrite each other's files and forge each other's marks. An audit that adds up evidence across a repository without inflating its error would need no provenance records.

Our key observation is that evidence from independent per-agent keys adds up across files and can be tested sequentially, so attribution needs no new watermark, only a new way to assign keys and to decide. We propose SWARMBLAME, a `git blame` for agent swarms that the agents cannot rewrite, as Figure 1 shows, in which every agent samples with the standard KGW watermark (Kirchenbauer et al., 2023) under its own key. The auditor runs an *anytime-valid* test over all keys, whose bound on ever naming an innocent holds for any reading order and stopping time, so it names an agent as soon as its evidence suffices. Two defences complete the protocol, as *composed keys*, one per ordered pair of agents, make a rewrite by a *relay*, an agent that rewrites another's file, record the chain, and *seen-pair exclusion* ignores evidence that a *thief*, an agent that learns a victim's mark to forge it, could have copied. Optionally, a text-only student model weights evidence toward the tokens where the generator was uncertain, and its weights provably keep the framing bound.

We evaluate SWARMBLAME on branch names, commit messages, PR texts and programs that eight keyed agents write for the same tasks, audited against 512 agent keys with 20 models from 1.5B to 34B parameters. In an interacting swarm whose agents plan, write, review and commit SWE-bench Verified fixes in turn, sessions of 32 artifacts name their agent in 98.6% of audits and never a teammate. Against the exact binomial test on the same marks, the strongest simple baseline, SWARMBLAME stays valid when the auditor stops early, whereas re-running that test after every file frames innocents in 218 of 14,000 audits and a per-file z-test in up to 8.93%. Across 21,000 *fresh-key audits*, which test unmarked text against new, hence innocent, keys, SWARMBLAME named an innocent agent 7 times, below the target rate of 0.1%, and sixteen synthetic pull requests name their author in 99.9% of audits. Learned weights raise attribution of 64 programs from 46% to 85%, and with a stronger mark on uncertain tokens, gated marking names four synthetic pull requests more often than SWEET, which needs the generator and its prompt. Composed keys name the chain in 96% of audits, and seen-pair exclusion lowers framing by stolen marks from 85% to 0%.

1.1 CONTRIBUTIONS

1. **A protocol with a framing bound.** We formalize attribution in agent swarms whose agents attack the audit, and design SWARMBLAME, which keeps standard KGW marks and adds an anytime-valid test over all agent keys, so the chance of ever naming an innocent stays below ε for any reading order (Sections 3 and 4).
2. **Two defences that only swarms need.** Composed keys make a rewritten text name the chain of agents behind it in 96% of audits, where a plain key blames the relay in 97%, and seen-pair

exclusion cuts framing by stolen marks from 85% to 0%, even when the thief deletes its sources (Section 4).

3. **An attribution environment.** We release the SWARMBLAME environment, a swarm of keyed agents on SWE-bench Verified and MBPP across 20 models, with an attack suite of rewrites, relays, colluders, thieves and planted edits and six prior schemes run as full alternatives, where re-testing detectors frame innocents in up to 8.93% of audits (Section 3.2).
4. **What makes an artifact attributable.** Evidence per artifact follows the generator's token entropy (Spearman $\rho = 0.89$), not model size, and a text-only student nearly doubles code attribution, from 46% to 85%, without the prompt at audit time (Sections 4.3 and 5.1).

2 RELATED WORK

Detecting a watermark. A detector must rarely accuse unmarked text, or attribution frames innocents. KGW marks a *green list* of tokens per key and previous token, biases sampling toward it and tests the green count with a z-test (Kirchenbauer et al., 2023), Unigram fixes one green list for robustness to edits (Zhao et al., 2023), and Gumbel sampling and other distortion-free schemes leave the output distribution unchanged in expectation (Aaronson & Kirchner, 2022; Kuditipudi et al., 2023; Christ et al., 2024; Dathathri et al., 2024). Fernandez et al. (2023) replace z-tests, miscalibrated at low false-positive rates, with exact tests on unique tokens, and SWARMBLAME keeps the KGW mark and this deduplication but changes how keys are assigned and tested.

Tracing many users. Multi-bit schemes write a user index into one key by key shifts (Fernandez et al., 2023), position allocation (Yoo et al., 2024), balanced vocabulary splits (Wang et al., 2024) or segment assignment (Qu et al., 2025), and collusion-secure codes trace *coalitions* that mix copies of one work (Boneh & Shaw, 1998; Tardos, 2008). Cohen et al. (2025) trace one text to one of many adaptive users with provable guarantees, while Fu & Russell (2025) show that unmarked text matches some user key more often as users grow. These schemes trace one text, whereas SWARMBLAME adds up evidence over a bundle's files and names relay chains.

Where evidence sits. Code carries little evidence, because most of its tokens are forced. SWEET marks and scores only high-entropy code tokens, recomputing their entropy with the generator and the prompt (Lee et al., 2024), EWD weights tokens by entropy (Lu et al., 2024), Invisible Entropy predicts high-entropy positions with a tagger shared by marking and detection (Gu et al., 2025), and code watermarks embed messages under grammar constraints or semantics-preserving transformations (Guan et al., 2024; Yang et al., 2024; Xu et al., 2025) that renaming can erase (Suresh et al., 2024). Our optional student instead weights standard KGW evidence from the text alone and keeps the framing bound.

Attacks and sequential tests. Paraphrasing removes a mark (Krishna et al., 2023; Sadasivan et al., 2023), no strong watermark survives every quality-preserving rewrite (Zhang et al., 2024), and watermark stealing learns green (previous token, token) pairs from marked text to forge or scrub a mark, even for small context widths (Jovanović et al., 2024). *E-processes* are running evidence measures whose false-alarm bound holds wherever one stops reading (Ramdas et al., 2023), and recent work uses them to detect one watermark in one growing text (Su et al., 2026; Huang et al., 2026a), whereas SWARMBLAME tests every agent key across a bundle's files.

Provenance of agent outputs. Failure attribution reads the interaction log (Zhang et al., 2025), version history and signed supply-chain records bind files to writers (The Git Project, 2026; Torres-Arias et al., 2019), AgentMark watermarks planning decisions decoded from logs (Huang et al., 2026b), and TRACE marks one agent's action trajectory against a reseller who controls the log (Gao et al., 2026). Implicit execution tracing recovers segment-level provenance from the final text of one dialogue with a keyed signal per agent (Nian et al., 2026), and behavioral fingerprints name the coding-agent product behind a pull request (Ghaleb, 2026b;a), not the instance of one model, with no guarantee against imitation. Unlike these, SWARMBLAME names the author of repository artifacts from content alone while the agents themselves relay, mix and forge each other's marks, with a framing bound for any reading order.

3 THREAT MODEL AND ENVIRONMENT

We consider an operator who runs a swarm of N coding agents through a trusted harness, in which every agent writes into one shared repository and a coalition C of agents may work against the others. Artifacts, or files, are the texts that land in the repository, namely code, branch names, commit messages and pull-request (PR) descriptions.

3.1 THREAT MODEL

A content-only attribution scheme consists of three functions.

- $K \leftarrow \textsc{KeyGen}(s)$ derives the key registry K from a secret s, with one key gk_i per agent i.
- $x \leftarrow \textsc{Generate}(p, k)$ samples a response x to prompt p with a watermark under key k.
- $a \leftarrow \textsc{Identify}(\mathcal{B}; K, R, \varepsilon)$ names an agent, a chain of agents or $\perp$ for a bundle $\mathcal{B}$.

Defender's capabilities and goals. The harness holds s, samples every output of agent i under gk_i, and mediates every write, so it knows which agent last wrote a file that another agent rewrites. Such a harness could also sign every write, but watermarks also cover text copied out of the repository, rewrites outside the harness, and records that an agent deletes or falsifies. The auditor, such as the operator after an incident, receives a bundle $\mathcal{B}$ believed to come from one agent, with K and the recovered repository R but no logs, prompts or timestamps. We call an agent *innocent* for $\mathcal{B}$ if it lies outside C and outside $\mathrm{Part}(\mathcal{B})$, the agents whose samplers produced the bundle's tokens, which for a relayed file are its author and its relay. The auditor's goals are to keep the probability of naming an innocent agent below a target rate ε and to name the author, or its chain, from few artifacts.

Adversary's capabilities. The adversary controls the agents in C. *(Knowledge)* It knows the algorithm and its parameters but no key, including its own, because keys never leave the harness. *(Data access)* It reads every file it reaches, including innocent agents' marked text, from which it can learn their green pairs (Jovanović et al., 2024). *(Actions and tools)* It rewrites, relays, mixes and deletes files, falsifies logs, and calls unkeyed tools, such as a model that paraphrases text or prefers pairs it believes are green for a *victim*, the innocent agent it wants to frame. *(Adaptive)* It knows the defences of Section 4 and tailors its forgeries to them. *(Resources)* It cannot read s, sample under another agent's key, query IDENTIFY, which runs after the incident, or search the 128-bit key space.

Adversary's objective. The adversary *frames* when IDENTIFY names an innocent agent, and it *evades* when a bundle of its own artifacts yields $\perp$. We exclude two trivial wins, since a bundle of a victim's own artifacts names the victim correctly and an unkeyed rewrite removes any watermark (Krishna et al., 2023; Zhang et al., 2024), so we bound framing and measure evasion.

Framing game and bound. Algorithm 1 draws the secret and the keys (line 1), lets the adversary corrupt a coalition, drive it through the swarm and pick a bundle and a reading order (lines 2 to 4), and the adversary wins if the audit names an innocent agent, counting a chain as its two agents (lines 5 and 6). A scheme meets the *framing bound* if $\mathrm{Succ}_{\textsc{Frame}}(\mathcal{A}) = \Pr[\textsc{Frame}_{\mathcal{A}} = 1] \leq \varepsilon$ for every adversary $\mathcal{A}$, and framing on unmarked text is a *false accusation*. Theorem 1 proves the bound against every adversary that reads no text of the agents it could frame, covering falsified logs, rewrites, relays and coalitions of any size. Seen-pair exclusion extends it to thieves that read only text in R, though not to one that deletes its sources, which Table 2 measures. Lines spliced unmarked into a victim's file are blamed on the victim, a *piggyback* attack (Pang et al., 2024) that no watermark attributes to the planter.

Algorithm 1: The framing game $\textsc{Frame}_{\mathcal{A}}$.

1: $s \xleftarrow{\$} \{0,1\}^{128}$; $K \leftarrow \textsc{KeyGen}(s)$ ▷ kept secret
2: $C \leftarrow \mathcal{A}(N)$ ▷ corrupt a coalition
3: $R \leftarrow \textsc{Run}^{\mathcal{A}}(K, C)$ ▷ $\mathcal{A}$ drives C
4: $(\mathcal{B}, \pi) \leftarrow \mathcal{A}(R)$ ▷ bundle, reading order
5: $a \leftarrow \textsc{Identify}(\pi(\mathcal{B}); K, R, \varepsilon)$ ▷ no logs
6: **return** $a \neq \perp \wedge a \not\subseteq \mathrm{Part}(\mathcal{B}) \cup C$ ▷ innocent named

3.2 THE SWARMBLAME ENVIRONMENT

We design the SWARMBLAME environment, in which a swarm of keyed coding agents writes into one repository and an attack suite plays the framing game against the audit, as Figure 2 shows. Watermark benchmarks score one generator's texts under one key (Piet et al., 2025; Tu et al., 2024), and failure-attribution benchmarks for agent systems read a trusted

log (Zhang et al., 2025), so neither lets agents relay, mix or forge each other's artifacts behind an untrusted log. It comprises (i) a keyed swarm on software tasks, (ii) bundles and key registries, (iii) an attack suite and (iv) baselines and metrics, detailed in Section C.

Swarm and tasks. Eight agents share `Qwen2.5-Coder-7B-Instruct` and receive identical prompts, so only their keys differ. For each of the 500 SWE-bench Verified issues (Jimenez et al., 2024), every agent writes a branch name, a commit message and a PR description, and for each of 500 MBPP tasks (Austin et al., 2021) and 500 TACO problems (Li et al., 2023) it writes a program that we execute against the reference tests. The sampler uses the KGW green fraction $\gamma = 0.25$ and logit bias $\delta \in \{1, 2, 3, 4\}$, and an unwatermarked sampler answers every prompt as the paired control. Nineteen further models from 1.5B to 34B parameters by nine developers repeat the main condition, and in an interacting variant four agents act in turn per issue as planner, author, reviewer and committer.

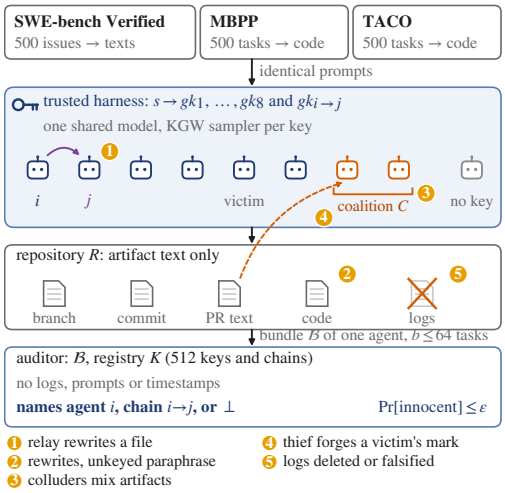


Figure 2: **The environment plays the framing game with real agents and attacks.** A trusted harness keys eight agents of one model on three task sets, a coalition C attacks a victim, and the auditor sees only artifacts. Badges mark where each attack enters.

Bundles and registries. A bundle holds one agent's artifacts for $b \le 64$ random tasks in random order, and a *PR bundle* groups the branch name, commit message and PR text of one issue with one function, a synthetic stand-in for one pull request, so b PR bundles hold $4b$ artifacts. The registry K holds 8 active keys and idle, hence innocent, keys up to $N = |K| = 512$, plus the active agents' composed keys for relayed files, and $\varepsilon = 10^{-3}$.

Attack suite. Formatters, canonicalisation and renaming rewrite code, and an unkeyed model paraphrases text (*Actions and tools*). A relay rewrites another agent's file, and a thief forges a victim's mark from 50 to 500 of its artifacts (*Data access*). Two colluders mix their artifacts, and a planter edits two lines of another agent's program under its own key (*Actions and tools*). A thief that deletes its sources forges only from pairs absent from the repository, and an agent regenerates its text while avoiding its estimated green pairs (*Adaptive*).

Baselines. Each prior scheme runs as a full alternative on the same swarm, bundles and registry with its detector corrected for testing every registry key, namely the KGW z-test and the exact binomial test (Fernandez et al., 2023) on our marks, swarms regenerated with Unigram, Gumbel sampling, SWEET and MPAC, and a character n-gram style classifier without watermarks.

Metrics. *Attribution* at bundle size b is the fraction of audits naming the bundle's author, the chain of a relayed file or a colluder of a mixed bundle, and we report the smallest bundle reaching 90%. Attribution under an attack measures robustness, and the *framing rate*, the fraction of audits naming an innocent agent, is capped at ε by the bound and called the false-accusation rate on unmarked bundles. *Utility* is the paired change in pass rate against the unwatermarked control, in points.

4 Method

SwarmBlame takes a bundle $\mathcal{B} = (x_1, \ldots, x_T)$ read in any order, the key registry K, the recovered repository R and a target rate ε, and outputs an agent, a chain of agents or $\perp$, as Figure 3 shows. It keeps the KGW sampler unchanged and adds three components, namely keys that record which agent sampled each token, including relays, an audit that tests every key as evidence accumulates and bounds framing at any stopping time, and optional key-free weights that concentrate the evidence, which Sections 4.1 to 4.3 describe in turn.

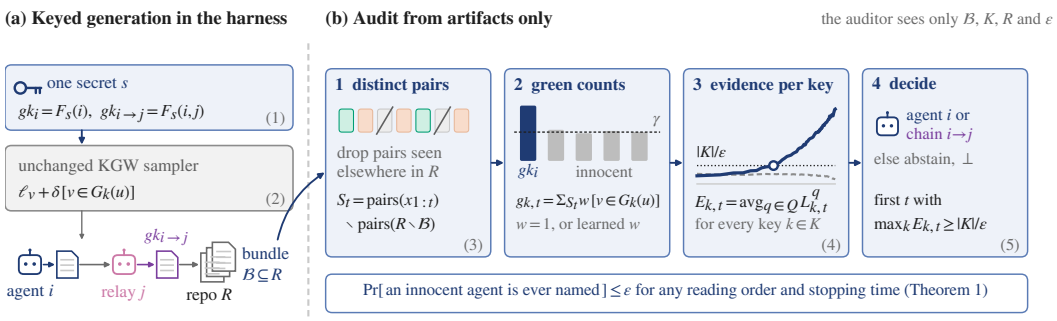


Figure 3: **SWARMBLAME (ours) keeps the KGW sampler and adds keys that record who wrote each token and an audit that tests every key.** (a) Keys from one secret (Eq. 1) and the unchanged sampler (Eq. 2). (b) The audit scores distinct unseen pairs (Eq. 3), adds up evidence per key (Eq. 4) and names the first key over $|K|/\varepsilon$ (Eq. 5). Tinted: what SWARMBLAME adds.

4.1 KEYED GENERATION

A shared key cannot tell agents apart, and a rewrite under the relay's key erases the author's mark. The harness therefore derives every key from one secret s with a pseudorandom function F,

$$gk_i = F_s(i), \qquad gk_{i\to j} = F_s(i,j), \tag{1}$$

where gk_i is the key of agent i and $gk_{i\to j}$ the *composed key* under which relay j rewrites a file of agent i. Agents sample with the unchanged KGW watermark (Kirchenbauer et al., 2023), which for key k and previous token u marks a *green list* $G_k(u)$ and biases the logit ℓ_v of every next token v,

$$\tilde{\ell}_v = \ell_v + \delta\,[v \in G_k(u)], \qquad G_k(u) = \{v : F_k(u,v) < \gamma\}, \tag{2}$$

where $[\cdot]$ is the indicator, γ the green fraction and δ the bias. Because the harness mediates every write, it knows the last writer i of every file it routes to agent j and samples j under $gk_{i\to j}$.

4.2 ANYTIME-VALID AUDIT

One artifact carries too little evidence to single out one of hundreds of keys, and re-testing every key after every file frames innocents, so the audit adds up evidence with a test valid wherever it stops. It retokenizes each artifact and keeps its distinct (previous token, token) pairs, except pairs that also occur outside the bundle in R, since a thief can only learn the colours of pairs it has read,

$$S_t = \mathrm{pairs}(x_1,\dots,x_t) \setminus \mathrm{pairs}(R \setminus \mathcal{B}), \tag{3}$$

which we call *seen-pair exclusion*. After t artifacts the auditor holds $n_t = |S_t|$ pairs, of which $g_{k,t}$ are green under key k, and the evidence, or e-value, of k is the mixture likelihood ratio

$$E_{k,t} = \frac{1}{|Q|}\sum_{q\in Q} L^q_{k,t}, \qquad L^q_{k,t} = \left(\frac{q}{\gamma}\right)^{g_{k,t}} \left(\frac{1-q}{1-\gamma}\right)^{n_t - g_{k,t}}, \tag{4}$$

where Q is a fixed grid of five green rates of a marked text, listed in Section A.2, and each $L^q_{k,t}$ compares how likely the green count is if the text was marked with k at rate q against if it was not. The auditor stops at the first artifact at which some key crosses the threshold $|K|/\varepsilon$,

$$\tau = \min\{t : \max_{k\in K} E_{k,t} \ge |K|/\varepsilon\}, \qquad \textsc{Identify}(\mathcal{B}) = \arg\max_{k\in K} E_{k,\tau}, \tag{5}$$

which names agent i for gk_i and the chain $i \to j$ for $gk_{i\to j}$, and abstains ($\perp$) if no key crosses.

Theorem 1 (Framing bound). *Let every distinct pair be green under a key with probability γ, independently of all other pairs and of the text, as Assumption 1 states, and let k be a key whose owner did not produce the bundle, so that the bundle's text is independent of k. Then, for every reading order and every stopping time τ, $\Pr\left[\sup_{t\le\tau} E_{k,t} \ge |K|/\varepsilon\right] \le \varepsilon/|K|$, and the probability that* IDENTIFY *ever names an innocent agent is at most ε.*

For such a key every pair is green with probability γ, so each $L^q_{k,t}$, and hence their average $E_{k,t}$, is a nonnegative martingale that starts at one, Ville's inequality (Ville, 1939) bounds by $\varepsilon/|K|$ the chance that $E_{k,t}$ ever reaches $|K|/\varepsilon$, and a union bound over K gives ε, as Section A.3 proves. Counting each distinct pair once follows Fernandez et al. (2023), and $E_{k,t}$ follows anytime-valid detection of one key (Su et al., 2026; Huang et al., 2026a), which we extend to a registry of agent and composed keys, and composed keys and seen-pair exclusion work with any valid multi-key test.

4.3 Optional learned evidence weights

A green list moves probability only where the model had a choice, so most evidence sits in the few tokens where the generator was uncertain, and code has few of them. SWEET finds them by recomputing the generator's entropy, which needs the generator and prompt at audit time (Lee et al., 2024). SWARMBLAME-W instead distills this per-token entropy into a text-only student model and gives each pair (u, v) a weight $w_{uv} \in [0, 1]$ before any key is read (Figure 7, Section B), so that

$$L^{q,w}_{k,t} = \prod_{(u,v)\in S_t} \left(\frac{q}{\gamma}\right)^{w_{uv}[v\in G_k(u)]} \left(\frac{1-q}{1-\gamma}\right)^{w_{uv}[v\notin G_k(u)]} \tag{6}$$

replaces $L^q_{k,t}$ in Equation (4). Such key-free weights keep every factor a supermartingale, whose expected value never grows, so Theorem 1 holds unchanged, as Proposition 1 shows. Because the students train on text marked with the eight active keys, the weights are key-free by construction for the 504 idle keys, and the audits of unmarked text in Table 1 check the active keys. SWARMBLAME-E is the one variant that changes marking, as it marks only the tokens whose generator entropy exceeds a threshold, the entropy gate of SWEET, and uses a stronger bias of $\delta = 8$ on them.

5 Experiments

In the environment of Section 3.2 we ask (i) whether the audit keeps its framing bound where prior detectors fail, (ii) how much evidence artifacts carry and what keys cost, (iii) whether attribution survives rewriting, relays, collusion and stolen marks, and (iv) whether learned weights recover evidence without the generator. Section D.6 ablates deduplication, the grid Q and registry size.

5.1 Keeping the framing bound

Re-testing after every file frames innocents unless looks are corrected. Figure 4a plots how often each detector of Table 1 frames an innocent in fresh-key audits against the target ε. Across 21,000 fresh-key audits SWARMBLAME named an innocent agent 7 times, a 95% upper bound of 0.063%, and never when two colluders mixed their artifacts. The exact binomial test stays near the target only at a bundle size fixed before reading, and re-running it after every file frames innocents in 218 of 14,000 audits (Section D.1), a per-file z-test in up to 8.93%, and MPAC in 38% of colluding bundles. Our test decides after 8 PR texts on average, and scheduling looks in advance makes a binomial test valid with more power, so the e-process trades some power for stopping at any file.

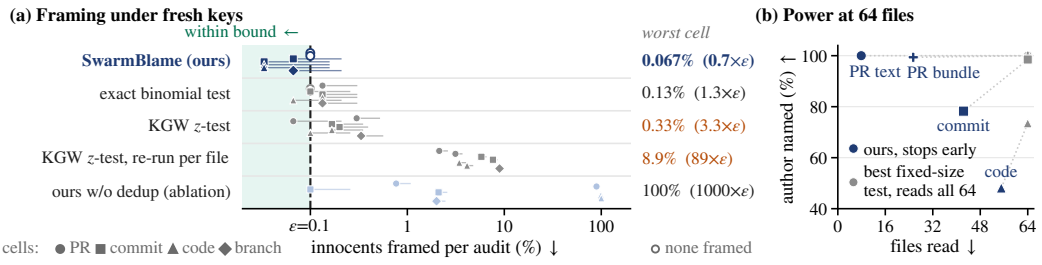


Figure 4: **Uncorrected re-testing frames innocents; SWARMBLAME (ours) stays within ε at any stopping time.** (a) Framing, 3,000 fresh-key audits per cell (95% bounds). (b) Power, files read.

Table 1: **SWARMBLAME (ours) against prior watermarks**, each a full scheme on the same swarm and bundles (7B, δ=2 unless stated, N=512, ε=10^{-3}). Author named: 1,600 audits per cell (Table 8). False acc.: audits of unmarked text naming an agent (learned rows add 2,000 fresh-key audits). *Prior tests on our marks, bundle size fixed before reading. ‡Decodes and tests on disjoint pair halves; uncalibrated, it falsely accuses in 1,493 of 8,000 audits (Section D.7). Small numbers: gain over the best baseline (÷ times fewer, + points more, none: no false accusation); †95% interval excludes zero.

		Author named (%)			False acc.	Innocent	MBPP	Laundered
Method	Audit needs	PR@16	PR bundle@4	Code@64	unmarked	2 colluders (%)	Δ (pp)	64 PR (%)
Unigram	keys	50	16	3	28/8k	0.3	−3.2	8
Gumbel	keys	96	35	60	6/8k	1	+0.1	59
SWEET	keys, model, prompt	100	83	100	19/8k	0.3	+0.4	8
MPAC‡, split-calibrated	one shared key	80	39	75	22/8k	38	−2.4	46, 24 wrong
Stylometry	labelled texts	51	8	75	>1%	1	0.0	29, 8 wrong
KGW z-test*	keys	100	64	72	19/8k	0.3	−2.1	95
Exact binomial*	keys	100	57	69	4/8k	0	−2.1	94
SWARMBLAME (ours)	keys	97	26	46	**1/8k** ÷4.0	**0**	−2.1	78
SWARMBLAME-W (ours), learned	keys + student	99.6	66	85	**1/6.8k** ÷3.4	**0**	−2.1	90
SWARMBLAME-E (ours), gated δ=8	keys + student	100	**90** +6.7†	85	**0/6.8k** none	**0**	−2.0	**97** +1.5
SWARMBLAME (ours), true entropy	keys, model, prompt	100	**92** +9.4	100	**0/6.8k** none	**0**	−2.1	**99** +3.5

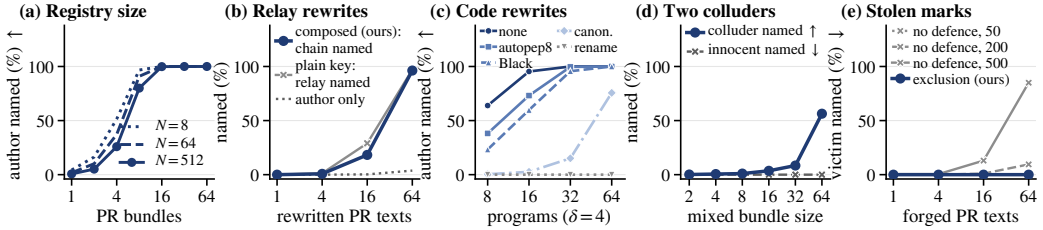


Figure 5: **SWARMBLAME (ours) keeps naming the author and blocks framing by stolen marks.**

5.2 EVIDENCE PER ARTIFACT AND ITS COST

A repository carries enough evidence. Table 6 in Section D measures the bits of evidence that real artifacts carry, the mean number of evidence pairs times the divergence between the author's green rate and γ. At $\delta = 2$ a branch name gives 0.49 bits, code 0.33 and a PR text 3.84, all far below the 19.0 bits needed among 512 agents, so no single artifact suffices. Summed over files, eight PR bundles name their author in 80% of audits and sixteen in 99.9%, and across all 315,000 audits 27 named a wrong agent (0.009%). A 30B model at $\delta = 4$ marks each TACO program with 47 bits, so four programs name their author in 90% of audits at a cost of 0.3 TACO points.

Keys cost about two MBPP points. On MBPP at $\delta = 2$ keys move the pass rate from 72.4% to 70.2%, a paired -2.1 points, and on 320 single-file SWE-bench Verified fixes by a 30B model the resolve rate changes by -0.3 points at $\delta = 2$ and -3.1 at $\delta = 4$, neither interval excluding zero. A stronger mark at $\delta = 4$ costs -13.4 MBPP points on the 7B model, which SWARMBLAME-E avoids.

Evidence follows entropy, not size. We find that across 20 models, bits per PR text follow the model's token entropy (Spearman $\rho = 0.89$) but not its size ($\rho = -0.31$, $p \approx 0.18$), as Figure 9d shows. A confident model needs a stronger mark, and the 30B mixture of experts absorbs $\delta = 4$ at 0.3 MBPP points while naming the author of 64 PR bundles in 100% of audits.

5.3 REWRITING, RELAYS, COLLUSION AND STOLEN MARKS

Marks survive rewriting. A rewrite by a keyed agent keeps a mark, so when the author rewrites its own PR texts, 8 of them still name it in 63% of audits. An unkeyed paraphrase copies marked phrases, so 64 laundered PR texts name their author in 97% of audits with gated marking and 78% with the default anytime rule, against at most 59% for any other scheme, and a test fixed at 64 texts reaches 94% on the same marks, as Table 1 shows. On code at $\delta = 4$, 64 programs name their author in 100% of audits after reformatting and 76% after AST canonicalisation, as Figure 5c shows.

Composed keys name the chain. When agent j rewrites 64 PR texts of agent i under its plain key, the audit names j in 97%, whereas under the composed key it names the chain $i \to j$ in 96% and never a wrong chain, and in 97.4% at the higher threshold of all N^2 composed keys of 512 agents, as the chain key overtakes the author's key, whose phrases the relay copies (Section D.1).

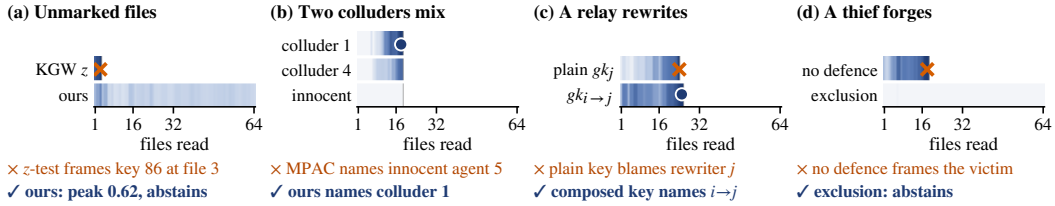


Figure 6: **Where a prior method names the wrong party, SWARMBLAME (ours) names the right one or abstains.** The first real audits in a fixed draw order (Section E); colour: statistic over threshold.

Collusion and stolen marks cannot frame an innocent. Two colluders that mix their artifacts frame no innocent in 8,400 audits, and the audit names a colluder in 56% of 64-artifact bundles (Figure 5d). A thief that reads the victim's PR texts on 500 tasks frames the victim in 85% of 64-text audits, and seen-pair exclusion lowers this to 0% at every size and amount of stolen text (Figures 5e and 6d), also with personas and against thieves that delete their sources, with no framing in 20,000 audits per setting (95% bound 0.015%). Even a thief that knows the defence, deletes its sources and forges only from pairs absent from the repository frames the victim in at most 3.5% of 64-text audits, against 70 to 88% without exclusion (Table 2).

Table 2: **Adaptive thieves and distinct prompts** (7B, δ=2, PR texts; % of 800 audits; the thief reads 500 victim tasks, and Figure 5e audits these forgeries 200 times, hence 85% there).

	Same prompt		Personas	
Bundle size	16	64	16	64
Honest, author named ↑	97.0	100	97.9	100
with exclusion (ours)	38.4	100	45.6	100
Thief, victim framed ↓	14.5	87.8	18.5	69.5
with exclusion (ours)	**0**	**0**	**0**	**0**
deletes its sources (ours)	**0**	**0**	**0**	**0**
deletion-aware (ours)	0	0.6	0.1	3.5
Self-scrub, own key ↑	57.5	100	69.2	100

5.4 AN INTERACTING SWARM

When four agents plan, describe, review and commit each of 500 SWE-bench Verified issues in turn, each reading its teammates' text, sessions of 32 artifacts name their agent in 98.6% of audits and 64 in 99.97%, and no audit names a teammate whose text an agent read (Table 3). An agent that plants a two-line bug in another agent's program under its own key is named by the changed tokens of 64 edits in 67.8% of audits, and the host whose file it edited never is. Section D.5 gives the setup and every outcome.

Table 3: **SWARMBLAME (ours) in an interacting swarm** (7B, δ=2, $|K|$=568; % of 3,200 audits).

Artifacts	8	16	32	64
Session, agent named ↑	26.9	68.0	98.6	**99.97**
teammate named ↓	0	0	0	0
Planted edits, planter ↑	0.2	2.5	20.6	**67.8**
host named ↓	0	0	0	0

5.5 EVIDENCE WITHOUT THE GENERATOR

Learned weights nearly double attribution on code. On low-entropy code, where anytime validity costs power, 64 programs name their author in 46% of audits against 69% for the exact binomial test, and learned weights from the 7B student lift this to 85%, above both fixed-size tests on the same marks, with more from larger students (Section B). With a stronger mark on its gated tokens, entropy-gated marking (SWARMBLAME-E) names four PR bundles in 90% of audits from the text alone, against 83% for SWEET, which needs the generator and the prompt.

6 CONCLUSION

We introduce SWARMBLAME, an attribution protocol on standard watermarks that names the agent or chain of agents behind a swarm's artifacts from the artifacts alone, while bounding the chance of ever naming an innocent agent. We encourage evaluating attribution schemes by their framing rate under early stopping, relays and stolen marks, the settings in which prior detectors fail. Section F discusses extending the harness to identifier renaming and to tool calls.

ETHICS STATEMENT

SWARMBLAME attributes text to software agents, not to people. Attribution can protect a swarm against a planted bug and can also be used to monitor agents or their operators; an operator who maps agents to people takes on the obligations of that mapping. The audit reads only artifacts already in a shared repository and collects no prompts, logs or telemetry. All guarantees rest on the harness keeping its key secret, so a deployment needs key storage, rotation and access control of the kind used for signing keys, and cryptographic key sizes (Section A.2). Our experiments use public benchmarks and model outputs only, with no human subjects.

REPRODUCIBILITY STATEMENT

All prompts, request files and sampling seeds are fixed, and every agent key is derived from one secret with HMAC-SHA256 (Section C). Every table and figure is regenerated by scripts from stored result files. Section C gives benchmarks, models, baselines and grading sandboxes; Section D.7 the statistical procedures; and Section B the training of the entropy students. Code, the library and the trained students will be released with the paper.

AI USE STATEMENT

We used large language models to explore related work and the literature, and to polish the writing of this paper. The authors reviewed all content and take full responsibility for the content presented in the paper.

REFERENCES

Scott Aaronson and Hendrik Kirchner. Watermarking gpt outputs. *Retrieved September*, 5:2024, 2022.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

Dan Boneh and James Shaw. Collusion-secure fingerprinting for digital data. *IEEE transactions on information theory*, 44(5):1897–1905, 1998.

Miranda Christ, Sam Gunn, and Or Zamir. Undetectable watermarks for language models. In *The Thirty Seventh Annual Conference on Learning Theory*, pp. 1125–1139. PMLR, 2024.

Aloni Cohen, Alexander Hoover, and Gabe Schoenbach. Watermarking language models for many adaptive users. In *2025 IEEE Symposium on Security and Privacy (SP)*, pp. 2583–2601. IEEE, 2025.

Sumanth Dathathri, Abigail See, Sumedh Ghaisas, Po-Sen Huang, Rob McAdam, Johannes Welbl, Vandana Bachani, Alex Kaskasoli, Robert Stanforth, Tatiana Matejovicova, et al. Scalable watermarking for identifying large language model outputs. *Nature*, 634(8035):818–823, 2024.

Pierre Fernandez, Antoine Chaffin, Karim Tit, Vivien Chappelier, and Teddy Furon. Three bricks to consolidate watermarks for large language models. In *2023 IEEE international workshop on information forensics and security (WIFS)*, pp. 1–6. IEEE, 2023.

Zihao Fu and Chris Russell. Multi-use llm watermarking and the false detection problem. *arXiv preprint arXiv:2506.15975*, 2025.

Zheng Gao, Xiaoyu Li, Xiaoyan Feng, Jiaojiao Jiang, Yang Song, Yulei Sui, Zhenchang Xing, and Liming Zhu. Trace: A two-channel robust attribution watermark via complementary embeddings for llm-agent trajectories. *arXiv preprint arXiv:2607.08400*, 2026.

Taher A Ghaleb. Agentag: Attribution of ai coding agents from behavioral fingerprints. *arXiv preprint arXiv:2608.00966*, 2026a.

Taher A Ghaleb. Fingerprinting ai coding agents on github. In *Proceedings of the 23rd International Conference on Mining Software Repositories*, pp. 822–826, 2026b.

Tianle Gu, Zongqi Wang, Kexin Huang, Yuanqi Yao, Xiangliang Zhang, Yujiu Yang, and Xiuying Chen. Invisible entropy: Towards safe and efficient low-entropy llm watermarking. pp. 6727–6744, 2025.

Batu Guan, Yao Wan, Zhangqian Bi, Zheng Wang, Hongyu Zhang, Pan Zhou, and Lichao Sun. Codeip: A grammar-guided multi-bit watermark for large language models of code. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 9243–9258, 2024.

Baihe Huang, Eric Xu, Kannan Ramchandran, Jiantao Jiao, and Michael I Jordan. Towards anytime-valid statistical watermarking. *arXiv preprint arXiv:2602.17608*, 2026a.

Kaibo Huang, Jin Tan, Yukun Wei, Wanling Li, Zipei Zhang, Hui Tian, Zhongliang Yang, and Linna Zhou. Agentmark: Utility-preserving behavioral watermarking for agents. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 12581–12603, 2026b.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. Swe-bench: Can language models resolve real-world github issues? In *International Conference on Learning Representations*, volume 2024, pp. 54107–54157, 2024.

Nikola Jovanović, Robin Staab, and Martin Vechev. Watermark stealing in large language models. 2024.

John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. A watermark for large language models. In *International conference on machine learning*, pp. 17061–17084. PMLR, 2023.

Kalpesh Krishna, Yixiao Song, Marzena Karpinska, John Wieting, and Mohit Iyyer. Paraphrasing evades detectors of ai-generated text, but retrieval is an effective defense. volume 36, pp. 27469–27500, 2023.

Rohith Kuditipudi, John Thickstun, Tatsunori Hashimoto, and Percy Liang. Robust distortion-free watermarks for language models. *arXiv preprint arXiv:2307.15593*, 2023.

Taehyun Lee, Seokhee Hong, Jaewoo Ahn, Ilgee Hong, Hwaran Lee, Sangdoo Yun, Jamin Shin, and Gunhee Kim. Who wrote this code? watermarking for code generation. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 4890–4911, 2024.

Rongao Li, Jie Fu, Bo-Wen Zhang, Tao Huang, Zhihong Sun, Chen Lyu, Guang Liu, Zhi Jin, and Ge Li. Taco: Topics in algorithmic code generation dataset. *arXiv preprint arXiv:2312.14852*, 2023.

Yijian Lu, Aiwei Liu, Dianzhi Yu, Jingjing Li, and Irwin King. An entropy-based text watermarking detection method. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 11724–11735, 2024.

Yi Nian, Haosen Cao, Shenzhe Zhu, Henry Peng Zou, Qingqing Luan, and Yue Zhao. When only the final text survives: Implicit execution tracing for multi-agent attribution. *arXiv preprint arXiv:2603.17445*, 2026.

OpenAI. OpenAI–Hugging Face incident: Technical report. Technical report, OpenAI, 2026. `https://cdn.openai.com/pdf/67869394-cb91-4c12-888c-5cbd85c7814c/OpenAI-Hugging-Face%20Incident-Technical-Report.pdf`.

Qi Pang, Shengyuan Hu, Wenting Zheng, and Virginia Smith. No free lunch in llm watermarking: Trade-offs in watermarking design choices. volume 37, pp. 138756–138788, 2024.

Julien Piet, Chawin Sitawarin, Vivian Fang, Norman Mu, and David Wagner. Markmywords: Analyzing and evaluating language model watermarks. In *2025 IEEE Conference on Secure and Trustworthy Machine Learning (SaTML)*, pp. 68–91. IEEE, 2025.

Wenjie Qu, Wengrui Zheng, Tianyang Tao, Dong Yin, Yanze Jiang, Zhihua Tian, Wei Zou, Jinyuan Jia, and Jiaheng Zhang. Provably robust multi-bit watermarking for {AI-generated} text. In *34th USENIX Security Symposium (USENIX Security 25)*, pp. 201–220, 2025.

Aaditya Ramdas, Peter Grünwald, Vladimir Vovk, and Glenn Shafer. Game-theoretic statistics and safe anytime-valid inference. *Statistical Science*, 38(4):576–601, 2023.

Vinu Sankar Sadasivan, Aounon Kumar, Sriram Balasubramanian, Wenxiao Wang, and Soheil Feizi. Can ai-generated text be reliably detected? *arXiv preprint arXiv:2303.11156*, 2023.

Weijie Su, Ruodu Wang, and Zinan Zhao. Online llm watermark detection via e-processes. *arXiv preprint arXiv:2602.14286*, 2026.

Tarun Suresh, Shubham Ugare, Gagandeep Singh, and Sasa Misailovic. Is the watermarking of llm-generated code robust? 2024.

Gábor Tardos. Optimal probabilistic fingerprint codes. *Journal of the ACM (JACM)*, 55(2):1–24, 2008.

The Git Project. git-blame: Show what revision and author last modified each line of a file. `https://git-scm.com/docs/git-blame`, 2026. Git reference manual.

Santiago Torres-Arias, Hammad Afzali, Trishank Karthik Kuppusamy, Reza Curtmola, and Justin Cappos. in-toto: Providing farm-to-table guarantees for bits and bytes. In *28th USENIX Security Symposium (USENIX Security 19)*, pp. 1393–1410, 2019.

Shangqing Tu, Yuliang Sun, Yushi Bai, Jifan Yu, Lei Hou, and Juanzi Li. Waterbench: Towards holistic evaluation of watermarks for large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1517–1542, 2024.

Jean Ville. *Etude critique de la notion de collectif*, volume 3. Gauthier-Villars Paris, 1939.

Lean Wang, Wenkai Yang, Deli Chen, Hao Zhou, Yankai Lin, Fandong Meng, Jie Zhou, and Xu Sun. Towards codable watermarking for injecting multi-bits information to llms. In *International Conference on Learning Representations*, volume 2024, pp. 39375–39401, 2024.

Rui Xu, Jiawei Chen, Weizhi Liu, Zhaoxia Yin, Cong Kong, and Xinpeng Zhang. Clasp: Training-free llm-assisted source code watermarking via semantic-preserving transformations. *arXiv preprint arXiv:2510.11251*, 2025.

Borui Yang, Wei Li, Liyao Xiang, and Bo Li. Srcmarker: Dual-channel source code watermarking via scalable code transformations. In *2024 IEEE Symposium on Security and Privacy (SP)*, pp. 4088–4106. IEEE, 2024.

KiYoon Yoo, Wonhyuk Ahn, and Nojun Kwak. Advancing beyond identification: Multi-bit watermark for large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 4031–4055, 2024.

Hanlin Zhang, Benjamin L Edelman, Danilo Francati, Daniele Venturi, Giuseppe Ateniese, and Boaz Barak. Watermarks in the sand: impossibility of strong watermarking for language models. In *Forty-first International Conference on Machine Learning*, 2024.

Shaokun Zhang, Ming Yin, Jieyu Zhang, Jiale Liu, Zhiguang Han, Jingyang Zhang, Beibin Li, Chi Wang, Huazheng Wang, Yiran Chen, et al. Which agent causes task failures and when? on automated failure attribution of llm multi-agent systems. 2025.

Xuandong Zhao, Prabhanjan Ananth, Lei Li, and Yu-Xiang Wang. Provable robust watermarking for ai-generated text. 2023.

Algorithm 2 IDENTIFY: anytime-valid audit of a bundle

Require: artifacts $x_1, x_2, \ldots$ in audit order; registry K of agent and composed keys; rate ε; recovered repository R

$S \leftarrow \emptyset$; $n \leftarrow 0$; $g_k \leftarrow 0$ for all $k \in K$
for $t = 1, 2, \ldots$ **do**
  for each distinct pair (u, v) of $\mathrm{tokens}(x_t)$ not in S **do**
    $S \leftarrow S \cup \{(u, v)\}$
    if exclusion is on and (u, v) occurs in R outside the bundle **then continue**
    end if
    $n \leftarrow n + 1$; $g_k \leftarrow g_k + \mathbf{1}[v \in \mathrm{Green}_k(u)]$ for all $k \in K$
  end for
  $E_k \leftarrow \frac{1}{|Q|} \sum_{q \in Q} (q/\gamma)^{g_k} \big((1-q)/(1-\gamma)\big)^{n - g_k}$ for all $k \in K$
  if $\max_k E_k \geq |K|/\varepsilon$ **then return** $\arg\max_k E_k$ ▷ an agent i or a chain $i \to j$
  end if
end for
return $\perp$

A THREAT MODEL AND PROOFS

A.1 THREAT MODEL

Table 4 lists each adversary capability of Section 3.1, whether the framing bound covers it, and what the evaluation measured. Only sampler output is attributed, so bytes copied from elsewhere or written by a tool carry the mark of their original generator or none.

Table 4: **Adversary capabilities** against the framing bound of SWARMBLAME (ours) and the evaluation (7B, δ=2 unless stated; % of audits).

Capability	Framing bound	Evaluated	Outcome
Falsify or delete logs	covered	by design	the audit reads no logs
Rewrite own files (keyed)	covered	yes	8 self-rewritten PR texts still name the author in 63%
Tool rewrites of code	covered	yes	Black 100%, canonicalisation 76%, renaming 0% at 64 (δ=4)
Unkeyed paraphrase (erasure)	framing only	yes	benign paraphrase of 64 texts 78%
Relay under a plain key	covered	yes	64 rewrites name the relay in 97%; composed key names the chain in 96%
Two colluders mix artifacts	covered	yes	no innocent in 8,400 audits; a colluder named in 56%
Steal marks and forge (spoof)	needs exclusion	yes	victim framed in 85% without and 0% with seen-pair exclusion
Steal marks, then delete the sources	**not covered**	yes	a deletion-aware thief frames the victim in 0.6% of 64-text audits, 3.5% with personas
Plant keyed lines in another's file	covered	yes	changed tokens of 64 edits name the planter in 68%, never the host
Splice unmarked lines into another's file	**not covered**	no	the file's marked author is named, not the planter
Key search	covered by key size	no	experiments use 31-bit keys for speed; deployment needs 128-bit keys

A.2 ALGORITHM

Algorithm 2 lists the audit. The sampler colours a vocabulary entry green when a hash of the entry and a seed from the key and the previous token falls below $\gamma 2^{31}$, the composed key $gk_{i \to j}$ is an HMAC of the secret and the ordered pair, and the e-value mixes the alternative green rates $Q = \{0.3, 0.4, 0.5, 0.65, 0.8\}$, which we fixed before the main runs. Keys are HMAC-SHA256 outputs of a 32-byte secret truncated to 31 bits, which keeps the sampler fast but must be raised to 128 bits with a cryptographic PRF in a deployment.

A.3 PROOF OF THE FRAMING BOUND

Notation. K is the key registry, γ the green fraction, Q the finite grid of alternative green rates and ε the target framing rate. For a key k, after t artifacts the auditor holds n_t distinct pairs, of which $g_{k,t}$ are green under k; $\mathcal{F}_t$ is the σ-algebra of everything the auditor has observed so far, and

$$E_{k,t} \;=\; \frac{1}{|Q|} \sum_{q \in Q} L^q_{k,t}, \qquad L^q_{k,t} \;=\; \Big(\frac{q}{\gamma}\Big)^{g_{k,t}} \Big(\frac{1-q}{1-\gamma}\Big)^{n_t - g_{k,t}}. \tag{7}$$

Assumption 1 (Random PRF)**.** Each distinct (previous token, token) pair is green under key k with probability γ, independently of all other pairs and of the text.

Theorem 1 in Section 4.2 states the framing bound.

Proof of Theorem 1. Step 1 (each alternative is a martingale). Fix $q \in Q$. A new distinct pair multiplies $L^q_{k,t}$ by

$$X_{t+1} \;=\; \begin{cases} \dfrac{q}{\gamma} & \text{if the pair is green under } k, \\ \dfrac{1-q}{1-\gamma} & \text{otherwise,} \end{cases} \tag{8}$$

and, since the pair is green with probability γ given $\mathcal{F}_t$ by Assumption 1,

$$\mathbb{E}\big[X_{t+1} \mid \mathcal{F}_t\big] \;=\; \gamma \cdot \frac{q}{\gamma} + (1-\gamma) \cdot \frac{1-q}{1-\gamma} \;=\; 1. \tag{9}$$

Artifacts that add no new pair leave $L^q_{k,t}$ unchanged. Hence $L^q_{k,t}$ is a nonnegative martingale with $L^q_{k,0} = 1$.

Step 2 (the mixture is a martingale). $E_{k,t}$ is an average of the nonnegative martingales $L^q_{k,t}$, $q \in Q$, each starting at 1, so it is a nonnegative martingale with $E_{k,0} = 1$.

Step 3 (Ville's inequality). For a nonnegative (super)martingale (M_t) with $M_0 = 1$ and any $c > 0$ (Ville, 1939),

$$\mathbb{P}\Big(\sup_{t \geq 0} M_t \geq c\Big) \;\leq\; \frac{1}{c}. \tag{10}$$

With $M_t = E_{k,t}$ and $c = |K|/\varepsilon$ this gives the claimed bound uniformly over time, hence for every stopping time and audit order.

Step 4 (union bound). IDENTIFY names an innocent agent only if some innocent key crosses the threshold, so

$$\mathbb{P}(\text{an innocent is named}) \;\leq\; \sum_{k \text{ innocent}} \mathbb{P}\Big(\sup_t E_{k,t} \geq \frac{|K|}{\varepsilon}\Big) \;\leq\; |K| \cdot \frac{\varepsilon}{|K|} \;=\; \varepsilon. \tag{11}$$

□

The proof uses only that distinct pairs are independent coins under an innocent key, which gives three immediate consequences. Pairs must be deduplicated across the bundle, because a repeated pair reuses the same PRF output. Composed keys are independent random functions of the ordered pair of agents, so Theorem 1 holds with a registry of $N + N(N-1) = N^2$ keys among N agents. Seen-pair exclusion keeps only pairs that occur nowhere else in R, so a thief that learns only from text the auditor holds cannot know the colours of the kept pairs, and Theorem 1 holds for its forged bundle.

Proposition 1 (Weighted evidence)**.** *Let each deduplicated pair i carry a weight $w_i \in [0,1]$ computed from the text alone, before any key is read, and let $g_i \in \{0,1\}$ be its colour under k. Define*

$$L^{q,w}_{k,t} \;=\; \prod_{i \leq n_t} \Big(\frac{q}{\gamma}\Big)^{w_i g_i} \Big(\frac{1-q}{1-\gamma}\Big)^{w_i(1-g_i)}, \qquad E^w_{k,t} \;=\; \frac{1}{|Q|} \sum_{q \in Q} L^{q,w}_{k,t}. \tag{12}$$

Then Theorem 1 holds with $E^w_{k,t}$ in place of $E_{k,t}$.

Proof. A new pair i multiplies $L^{q,w}_{k,t}$ by X^{w_i}, where X is the unweighted factor of Step 1 with $\mathbb{E}[X \mid \mathcal{F}_t] = 1$. The weight w_i is fixed by the text and so independent of the pair's colour, and $x \mapsto x^w$ is concave on $[0,\infty)$ for $w \in [0,1]$. By Jensen's inequality,

$$\mathbb{E}\big[X^{w_i} \mid \mathcal{F}_t\big] \;\leq\; \big(\mathbb{E}[X \mid \mathcal{F}_t]\big)^{w_i} \;=\; 1. \tag{13}$$

Thus $L^{q,w}_{k,t}$ and its mixture are nonnegative supermartingales starting at 1, and Steps 3 and 4 apply unchanged. □

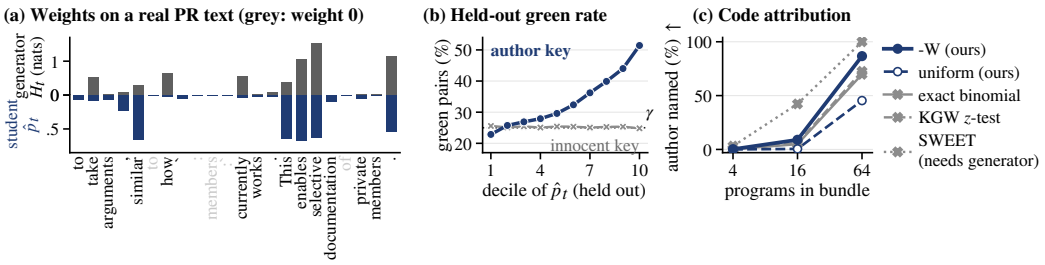


Figure 7: **A text-only student finds the uncertain tokens that carry evidence.** (a) Entropy and student prediction $\hat{p}_t$ on a real PR text. (b) Held-out green rate. (c) Code audits naming the author.

B Learned evidence weights

SwarmBlame-W distills the generator's per-token entropy into a student that reads only the text, and SwarmBlame-E marks a token only when the generator's entropy exceeds τ=0.9 nats. The student predicts $\log(1+H_t)$ of the generator's entropy H_t, computed with the prompt, with a Huber loss, and the gate $\mathbf{1}[H_t > \tau]$ with cross-entropy. We train students of 0.5B, 1.5B and 7B parameters with AdamW on sequences of at most 512 tokens and hash tasks into five folds, so that every artifact is scored by a student that never saw its task. At audit time a pair gets weight 1 when its predicted value is in the top half over the audited text and weight 0 otherwise. Because the students train on text marked with the eight active keys, these weights are key-free by construction for the 504 idle keys, so Proposition 1 keeps the framing bound for every idle key.

Held out by task, the student's prediction tracks the generator's entropy, and the pairs it keeps are green under the author's key far more often than those it drops, while an innocent key stays at γ (Figure 7b). Table 5 shows that learned weights nearly double code attribution, more so with larger students, without adding false accusations, and that gating lets a strong mark cost no more utility than the default mark while naming four PR bundles more often than SWEET, which needs the generator and the prompt.

Table 5: **Learned and gated variants of SwarmBlame (ours) against SWEET** (7B; % of 1,600 audits naming the author, the draw of Tables 1 and 8; false accusations over unwatermarked and fresh-key audits).

Variant	Audit needs	PR@16	Bundle@4	Commit@64	Code@64	False acc.	MBPP Δ
SwarmBlame (ours), δ=2	keys	97	26	83	46	1/8.0k	−2.1
+ learned weights, 0.5B student	keys + student	99	59	97	54	1/6.8k	−2.1
+ learned weights, 1.5B student	keys + student	99	62	97	73	1/6.8k	−2.1
+ learned weights, 7B student	keys + student	99.6	66	96	85	1/6.8k	−2.1
+ generator entropy (oracle)	keys + model + prompt	100	92	100	100	0/6.8k	−2.1
SwarmBlame-E (ours), gated δ=8, 1.5B student	keys + student	100	88	99.6	76	1/6.8k	−2.0
7B student	keys + student	100	90	99	85	0/6.8k	−2.0
proxy entropy, generator w/o prompt	keys + generator	100	82	96	82	1/6.8k	−2.0
SWEET	keys + model + prompt	100	83	–	100	19/8.0k	+0.4

C Experimental details

Environment. The trusted harness of Figure 2 runs every agent on one shared model, stores the extracted text of each artifact, and hands the auditor only bundles of that text and the key registry. The repository is this store of artifact text rather than a git repository, and the eight agents of the main swarm generate in parallel without reading each other, whereas in the interacting swarm of Section D.5 they act in turn on each issue.

Models and tasks. The main swarm uses `Qwen2.5-Coder-7B-Instruct`, and 19 further instruction-tuned backbones from 1.5B to 34B parameters repeat the main condition, with every agent and the unwatermarked control sampling at temperature 0.7 and top-p 0.95. For each of the 500 SWE-bench Verified issues (Jimenez et al., 2024), every agent writes a branch name, a commit message and a PR description, and for each of the 500 MBPP tasks (Austin et al., 2021) it writes a function that we execute against the three reference tests in a network-free sandbox, of which the prompt shows the first, as is standard for MBPP. TACO (Li et al., 2023), 500 problems judged by stdin and stdout tests, serves both attribution and utility, and single-file SWE-bench fixes measure

utility. The 30B model prefaces each TACO answer with a prose explanation of its approach, which is why its TACO answers carry evidence while its other artifacts carry almost none (Table 6).

Baselines. Each prior scheme runs as a full alternative on the same swarm, tasks, bundles and 512-key registry, extended by the composed keys of the active agents where files are relayed, with its detector corrected for the registry by a Bonferroni factor. The KGW z-test and the exact binomial test score our marks, Unigram, Gumbel, SWEET and MPAC regenerate the swarm with their own samplers, and a character n-gram classifier serves as a stylometry baseline.

Implementation. One batched logits processor serves rows with different keys, so marking adds no model call, and programs run in a sandbox with a 10 s limit and 1 GiB of memory. Every table and figure is regenerated by scripts from stored requests, outputs and results.

D Further results

Table 6: **Attribution and utility of SwarmBlame (ours):** bits per artifact, smallest bundle named in 90% of audits (N=512, ε=10^{-3}; PR b.: PR bundles) and pass-rate change. Green: $\leq$16 artifacts.

	Bits per artifact						*Bundle for 90% attribution*						*Pass-rate change (pp)*	
Model, mark	Branch	Commit	Code	PR	PR b.	TACO	Branch	Commit	Code	PR	PR b.	TACO	MBPP	TACO
7B, δ=1	0.11	0.13	0.04	0.92	1.20	0.11	>64	>64	>64	64	64	>64	+0.2	−1.4
7B, δ=2	0.49	0.62	0.33	3.84	5.27	0.79	>64	>64	>64	**16**	**16**	64	−2.1	−0.9
7B, δ=4	2.24	3.24	3.78	22.92	32.18	8.88	32	**16**	**16**	**4**	**2**	**8**	−13.4	−2.5
30B, δ=2	0.03	0.04	0.00	0.56	0.64	8.34	>64	>64	>64	>64	>64	**8**	−0.1	−1.6
30B, δ=4	0.13	0.20	0.10	2.55	2.98	46.97	>64	>64	>64	32	**16**	**4**	+0.3	−0.3

D.1 Validity, decision rules and audits at scale

Each fresh-key audit of Figure 4a draws a fresh registry of 512 random keys and a random unwatermarked bundle, so audits are independent over keys, and SwarmBlame names an innocent agent at most twice in 3,000 audits per cell. The decision rule is pluggable: an exact binomial test stays valid when the auditor fixes in advance to test only after 1, 2, 4, 8 and so on files, with a Bonferroni correction over keys and looks, and such fixed-size tests name the author more often on weak channels, whereas the e-process reads about as few files and stays valid when the auditor checks after every file, the setting of our audit (Figure 8a). Re-running the exact binomial test after every file without this correction names an innocent in 218 of 14,000 fresh-key audits over seven cells of 2,000, up to 2.4% per cell, against 2 for the e-process on the same audits. When the registry holds the 512 agent keys and every composed key of an active agent ($|K|$=8,632), 64 relayed PR texts name the chain in 97.5% of audits, and in 97.4% when the threshold counts all N^2=262,144 keys of 512 agents, with no wrong key or chain named in 1,000 audits per cell, and honest bundles that mix two authors name one of the two in 99.9 to 100% of 64-text audits.

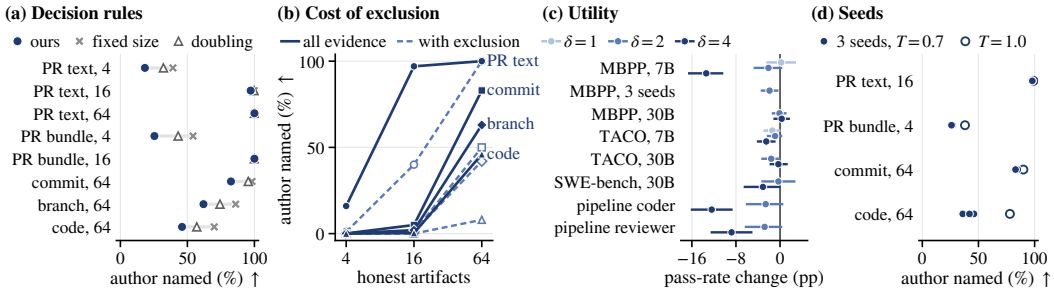


Figure 8: **Four checks of SwarmBlame (ours).** (a) Author named by three decision rules; innocents named in 45,000 fresh-key audits: 3 (ours), 25 (fixed size) and 20 (doubling looks). (b) Honest bundles with all evidence and with seen-pair exclusion. (c) Paired pass-rate change with 95% intervals. (d) Three sampling seeds and a higher temperature.

D.2 Backbones

We repeat the main condition at δ=2 on 20 models by nine developers, including a Qwen coder size ladder, general models and large dense models from 24B to 34B parameters. Sixteen PR texts

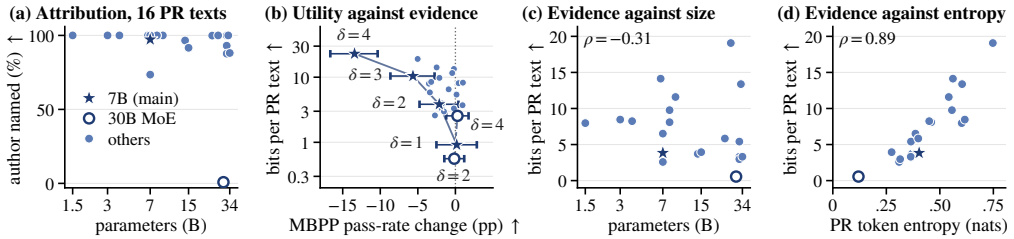


Figure 9: **Evidence of SWARMBLAME (ours) follows token entropy, not model size.** (a) Attribution. (b) Utility against bits. (c, d) Bits against size and entropy, over 20 backbones at δ=2.

name their author in 88 to 100% of audits on 18 of the 20 backbones, and the two exceptions, the general `Qwen2.5-7B` and the confident 30B mixture of experts, carry the fewest bits per PR text (Figure 9c,d). Confident models absorb a stronger mark, so that at δ=4 `Qwen2.5-Coder-32B` names the author of four PR bundles in 98.7% of audits.

D.3 UTILITY AND SEEDS

Keys cost about two MBPP points at δ=2, the stronger δ=4 costs markedly more on the 7B model but not on the 30B mixture of experts, whose confident sampling absorbs the mark, and single-file SWE-bench Verified fixes by the 30B model change by -0.3 points at δ=2 and -3.1 at δ=4, neither interval excluding zero (Figure 8c). Three sampling seeds give the same picture, and a higher temperature adds evidence (Figure 8d).

D.4 STOLEN MARKS AND ADAPTIVE ADVERSARIES

The thief reads the victim's artifacts on 50, 200 or 500 tasks, estimates the victim's green pairs by contrast with the other agents, and forges text with an unkeyed model that adds 4 to their logits. Seen-pair exclusion removes the pairs a thief could have read, which also costs honest attribution on weak channels, while 64 honest PR texts still name their author in every audit (Figure 8b). With a distinct persona per agent attribution is unchanged, exclusion keeps framing at 0% against a thief that deletes the files it read, and an agent that regenerates its own text while avoiding its estimated green pairs is still named in every 64-text audit (Table 2).

D.5 AN INTERACTING SWARM AND PLANTED EDITS

For each of the 500 SWE-bench Verified issues, four of the eight agents, drawn at random, act in turn: a planner writes a plan, an author reads it and writes the PR text, a reviewer reads that text and rewrites it under the composed key of the author and the reviewer, and a committer reads the reviewed text and writes the branch name and commit message. A session bundle holds what one agent wrote under its own key across issues, and the registry holds 512 agent keys and the 56 composed keys of the active agents, all counted in the threshold. In a second experiment, agent j receives the MBPP program of agent i and changes at most two lines so that it contains a subtle bug, under its own key or under $gk_{i\to j}$, and the auditor scores the whole file or only the pairs that are new relative to the file it received (1,000 edits per key). Sessions name their agent as in the independent swarm and never a teammate whose text an agent read, and the changed tokens of 64 planted edits name the planter without ever naming the host (Table 7). Reviewed texts carry the chain key, with 29.9% green pairs against 25.0% under an innocent key, but each chain occurs in only nine rewrites on average, too few to name it, so these audits abstain, as do composed planted edits, and with seen-pair exclusion, which also drops copied phrases, 64 artifacts name the agent in 91.3% of audits.

D.6 ABLATIONS

Deduplication is required for validity: without it repeated pairs reuse one coin and frame innocents in 8800 of 21,000 fresh-key audits. The mixture over green rates matters for power: a single alternative

Table 7: **SWARMBLAME (ours) in an interacting swarm** (7B, δ=2, $|K|$=568, ε=10^{-3}; % of audits, 3,200 per cell).

Bundle	Outcome	8	16	32	64
Session bundle of one agent	agent named	26.9	68.0	98.6	**99.97**
	teammate named	0	0	0	0
	innocent named	0	0	0	0.03
Session bundle, seen-pair exclusion	agent named	1.4	8.5	44.9	91.3
Changed tokens of planted edits, own key	planter named	0.2	2.5	20.6	**67.8**
	host named	0	0	0	0
	another agent named	0	0.03	0	0.03
Whole files with planted edits, own key	anyone named	0	0	0.06	0.1
Reviewed PR texts of one reviewer	anyone named	0	0.03	0.03	0

$q = 0.5$ names 16 PR texts in 32% of audits, our grid in 96%. Growing the registry from 8 to 32,768 keys lowers attribution of 16 PR texts only from 100% to 84%, with no innocent named.

D.7 STATISTICAL ANALYSIS

Table 8: **Attribution with 95% task-cluster bootstrap intervals** and false accusations with exact upper bounds (% of 1,600 audits per cell; 7B, N=512, ε=10^{-3}; learned rows include 2,000 fresh-key audits).

Method	PR@16	Bundle@4	Code@64	False acc. (upper 95%)
SWARMBLAME (ours)	97.2 [96.0, 98.4]	25.9 [21.9, 27.6]	45.5 [37.7, 53.9]	**1/8000 (0.059%)**
SWARMBLAME-W (ours)	99.6 [99.3, 99.9]	65.9 [62.2, 68.6]	85.4 [79.4, 90.5]	**1/6800 (0.070%)**
SWARMBLAME-E (ours), gated δ=8	**100.0 [99.9, 100.0]**	**89.7 [87.0, 90.9]**	85.3 [80.2, 91.4]	**0/6800 (0.044%)**
KGW z-test	100.0 [99.8, 100.0]	63.6 [58.7, 65.2]	72.0 [65.5, 79.9]	19/8000 (0.348%)
Exact binomial	100.0 [99.7, 100.0]	57.2 [52.0, 58.6]	68.6 [62.2, 77.0]	4/8000 (0.114%)
Unigram	50.4 [46.8, 54.9]	15.6 [13.9, 18.1]	3.4 [1.4, 6.7]	28/8000 (0.480%)
Gumbel	95.7 [93.8, 96.9]	35.1 [30.6, 37.0]	60.4 [51.6, 67.9]	6/8000 (0.148%)
SWEET	100.0 [100.0, 100.0]	83.0 [81.3, 86.0]	100.0 [99.3, 100.0]	19/8000 (0.348%)
MPAC, split-cal.	80.0	38.9	74.9	22/8000 (0.392%)
MPAC, uncal.	97.1 [96.0, 98.0]	74.6 [72.1, 77.0]	92.2 [89.1, 95.4]	1493/8000 (19.394%)

Audits reuse tasks, so we resample tasks in a task-cluster bootstrap with 2,000 draws that is paired across methods, and report false accusations with exact one-sided Clopper–Pearson bounds (Table 8), and a difference in Table 1 is marked significant only when its 95% interval excludes zero. MPAC as published decodes each of its nine bits as the majority colour at that position and then z-tests the matched count on the same pairs, so on unmarked text the matched count exceeds half the pairs by about $\sqrt{18/\pi} \approx 2.4$ standard deviations, and this uncalibrated test names an agent in 1,493 of 8,000 unmarked audits. Table 1 therefore decodes on a hash-selected half of the pairs and tests on the other, which lowers this to 22 of 8,000; its row in Table 8 has no bootstrap interval.

E WORKED EXAMPLE

Each audit of Figure 6 is the first seeded draw of its setting in which the compared method names a wrong party: (a) 64 unmarked commit messages under the per-file z-test, (b) two colluders under MPAC, (c) the first relay bundle and (d) the first forged bundle that names the victim without a defence. Figure 10 shows the artifacts behind (c) and (d), unedited, for the first relay chain and task in sorted order and the first forged PR text in sorted order whose green share under the victim's key reaches $z \geq 2$, with every token coloured by its pair under the stated key. One rewrite is too short to decide, but over 500 rewrites the plain rewrite carries the rewriter's key and the composed rewrite the chain key, and most pairs of the forged text also occur elsewhere in the repository, so seen-pair exclusion removes them before any key is tested.

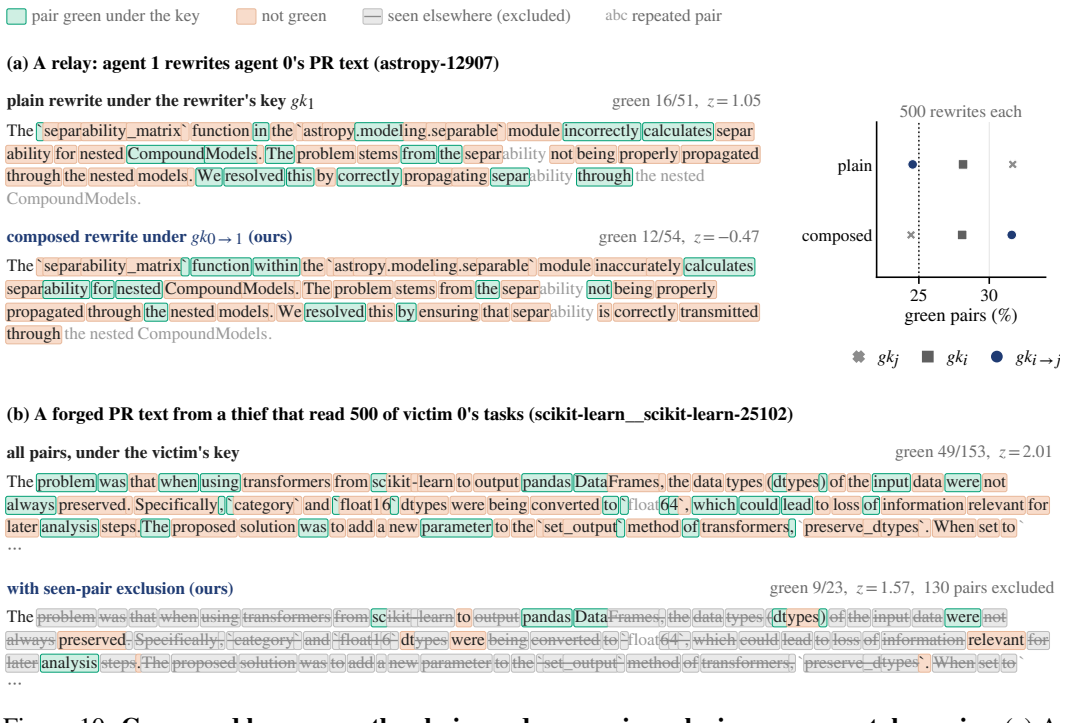


Figure 10: **Composed keys carry the chain, and seen-pair exclusion removes stolen pairs.** (a) A plain and a composed relay rewrite, with mean green shares over 500 rewrites. (b) A forged PR text with all pairs and after exclusion (first three lines shown). Figure 6c,d audits their bundles.

F Discussion

Towards identifier-level robustness. Identifier renaming rewrites most of the uncertain tokens of code and so removes its evidence, whereas reformatting and AST canonicalisation keep attribution at 100% and 76% (Figure 5c). The commit message and PR text of the same change keep their evidence, and a natural extension marks the edit itself at the harness, for example by keying the choice of identifier names or by combining token marks with semantics-preserving code watermarks (Yang et al., 2024), so that evidence survives renaming.

Unkeyed tools. Only sampled text carries a mark, so a harness that routes every model call through keyed samplers leaves an agent no unkeyed rewrite, and an agent that regenerates its text under its own key is still named in 100% of 64-text audits, as Table 2 shows. An unkeyed rewrite outside the harness leaves no mark, as for any watermark (Krishna et al., 2023; Zhang et al., 2024), and the harness's write record can flag a bundle whose known author leaves no evidence.