What Evidence Do AI Agent Audit Logs Need After Redaction?

A Matched-Pair Pilot on Verifying Consent from Masked and Aliased Logs

Anish Prashun
Independent researcher

Independent technical report Version 1.2 November 2026
DOI: 10.5281/zenodo.24163831

Abstract

Audit logs for AI agents often have to be redacted before a reviewer sees them, yet the reviewer must still decide whether each recorded action was permitted. We study this tension in a deliberately small, fully reproducible setting. Twenty scripted runs of a fictional CRM agent's completed `share_document` action form ten matched pairs. Each pair contains one authorised and one unauthorised share that differ in a single policy-relevant fact: recipient, document, expiry or revocation. Every run is rendered as a full log, a uniformly masked log and a log with consistent within-run aliases, plus two ablations that drop expiry or revocation evidence. Each exact input is given to an isolated three-valued rule reviewer.

Uniform masking made 6 of 10 pairs byte-identical: every recipient pair and every document pair. The reviewer could confirm none of the 10 authorised shares and detected only the 4 violations that rest on timing. Consistent aliases restored all 20 verdicts while exposing none of the 66 planted raw values. Removing expiry or revocation evidence left 12 of 20 decisions unresolved, including every authorised share.

These outcomes follow largely from the design. The pilot's value is to make the mechanism measurable, to separate *distinguishability* from *verifiability*, and to show which evidence this rule needs from a redacted log: identity equality within the run, expiry, and an explicit revocation state. It does not need raw identifiers, names or contents.

We then turn the comparison into a reusable, standard-library toolkit. Users describe their own rule as conditions tagged with one of five *evidence needs* (equality, order, explicit state, raw value, presence), mark their private fields, supply a few authorised logs, and list the redaction schemes to compare. The toolkit generates boundary violations, reports for every condition and scheme whether verification stays possible, and reproduces the pilot's tables byte for byte. In exploratory runs, hashing identifiers with a fresh salt per event, which is the documented behaviour of a common redactor's hash operator without a fixed salt, gave a perfect ceiling and no identical pairs. Yet a reviewer unaware of the scheme flagged all 10 authorised shares as violations. Redacted logs should therefore declare how they were redacted. On a third, 12-condition payment policy, the toolkit found a design that kept every condition verifiable with no private value visible: per-run keyed pseudonyms, with times and amounts replaced by their order within the run.

1 Introduction

Agent deployments increasingly keep run records so that a later reviewer can decide whether an action was allowed. Those records often contain personal data, so organisations minimise or redact them before review. Redaction answers the privacy question but raises an audit question: after redaction, can the reviewer still tell a permitted action from a violation?

Recent work frames this as *policy checkability*: whether a record contains enough information for compliance to be determined mechanically [1]. That work also notes that complete audit records can conflict with data minimisation. We make the question concrete for one consent-based authorisation rule and three common log representations, measuring on matched pairs what each representation lets a reviewer determine. The study is a retrospective audit of recorded effects: the mock executor completed every scripted share, including deliberately unauthorised ones. We test what the logs support, not how well a person or a language model uses them.

Contributions.

1. A matched-pair design and metric suite that separates three questions: whether inputs can be told apart (identical-input pairs and a forced-binary ceiling), whether consent can be verified (a three-valued reviewer) and what is directly exposed (planted values).
2. A reproducible implementation using only the Python standard library. It includes a protocol pre-specified and hash-frozen before the main run (self-attested), reviewer isolation, 91 automated checks and mutation tests of its safeguards.
3. Pilot results. Uniform masking makes every share whose verdict depends on identity unverifiable and leaves only timing violations detectable. Consistent aliases restore every verdict with zero direct exposure. Omitting a single temporal field prevents confirming any authorised share.
4. Two properties of the ceiling metric. Merging identical-input groups can never raise it. For views in which every unauthorised run that shows no failing condition is byte-identical to its authorised partner, as holds for all five views tested here, it equals the number of authorised runs plus the number of unauthorised runs whose own input shows a failing condition, so collisions between pairs leave it unchanged.
5. A reusable toolkit and vocabulary (§11):
   - five evidence needs;
   - per-field redaction declarations;
   - generated boundary violations;
   - a condition-by-scheme matrix with verified, unverifiable and unsafe statuses.

   It also includes two further worked policies, a support agent on trace-shaped logs and a payment agent with 12 conditions, and exploratory results for common redaction choices.

2 Related work

Auditability of agents. Nian et al. [1] separate accountability, auditability and auditing.

- They define *policy checkability* as whether "the record contains enough information for compliance to be mechanically determined". Its outcomes are comply, violate or undecidable ($\perp$).
- Their Proposition 1 shows that omitting a required field can make a policy undecidable.
- Their SPDR metric is the fraction of a policy set that is decidable from a record.

- They treat privacy as a constraint on evidence design, list redaction among ways to reduce evidence cost, and state that they do not develop "concrete mechanisms for reconciling high-fidelity audit records with data-minimization requirements".
- Their closest empirical result is a recovery experiment (§4.3) in which sensitive information is replaced with placeholders. It measures how much responsibility attribution can be recovered, not policy checkability under redaction.

This pilot is a *focused extension* of that framework: one policy, three concrete redaction formats and matched pairs. Our reviewer's three outcomes mirror theirs, and our definitive-answer coverage is the counterpart of SPDR taken across records for a single policy. Our ablations illustrate Proposition 1 for *confirming* compliance. They also show an asymmetry the proposition does not draw: for a rule with several conditions, a missing field blocks every "authorised" verdict but not every "unauthorised" one, because any other condition known to fail still establishes a violation.

Audit over incomplete or protected logs. Logic-based auditing has long allowed for logs that lack the information needed to decide a policy. Garg et al. treat facts as true, false or unknown and leave a residual policy for what cannot yet be decided [2]. Our three-valued reviewer follows that tradition. Chowdhury et al. ran an existing privacy-policy violation finder over logs encrypted under schemes that "largely hide data but reveal enough information" for the relational queries the audit needs [3]. They thereby established that policy audit can survive protection only if the transformation keeps some structure. Our consistent aliases keep exactly that structure (equality) in non-cryptographic form, and our uniform-mask baseline shows empirically what is lost without it. Pseudonyms scoped to a transaction in audit data date back at least to Biskup and Flegel [4]. Lakkaraju and Slagell compared single-field anonymisation policies on network traces and identified the fields that matter most for intrusion-detection utility [5]. Our setting differs because authorisation depends on equality *across* events.

Paired evaluation of monitors. SLEIGHT-Bench [6, 7] pairs attack transcripts with benign variants that remove the malicious action and are otherwise identical. It uses the benign variants to calibrate monitor thresholds and checks that "the information necessary to flag the attack is present, even if hidden". Our identical-pair metric asks whether that information survives redaction.

Redaction that preserves structure. SlotGuard [8] rewrites sensitive structural bindings in agent transcripts as typed, suffix-aware slots before the transcripts reach a model provider. It compares this with generic redaction and with a typed-placeholder baseline. On its 200-task probe, generic redaction dropped to 2.5% task success for each of four models, while SlotGuard scored 89.5–100%. Our contrast between uniform masks and consistent aliases is the audit-side counterpart. We measure a later reviewer's ability to verify consent, not the agent's task success.

Purpose-bound privacy and contextual integrity. ToolPrivacyBench [9] audits whether private data reaches only authorised tools and sinks. It compares recorded tool arguments and backend audit logs with a policy knowledge base, on the principle that each tool should receive only the information necessary for its stated purpose. We ask the same minimisation question about the reviewer. AgentCIScope [10] grounds disclosure constraints in privacy policies, persona agreement and a taxonomy of sensitive attributes. Our rule is deliberately narrower: one explicit grant per case.

Adjacent fields and guidance.

- **Process mining.** Privacy transformations of event logs can produce misleading analysis [12], and case variants added by anonymisation are potential false positives for conformance checking [13].
- **Network traces.** Deterministic prefix-preserving schemes such as CryptoPAn keep structure, yet fingerprinting attacks can exploit fields left unchanged, such as timestamps [14]. Anonymised traces also allow topology inference and server de-anonymisation [15].
- **Guidance.** ENISA treats the choice of pseudonymisation technique as risk-based [17]. The EDPB recommends that the sets of data pseudonymised consistently be "as small as possible" [18]. ISO/IEC TS 27560 specifies the structure of consent records [19].

Novelty. Searches on 12 November 2026 found no prior study that measures, on matched pairs of authorised and unauthorised agent actions differing in a single policy-relevant fact, how uniform masking and within-run equality-preserving aliases change a rule-based reviewer's ability to verify a consent-based authorisation policy (grant, expiry, revocation) from the recorded log, together with direct exposure of planted values. Prior work established the ingredients:

- that audit over protected logs needs equality-revealing transformations [3];
- that constant and consistent anonymisation can be compared for analytic utility [5];
- that incomplete logs call for three-valued verdicts [2, 1];
- that consistent substitution beats generic masking for model utility [8].

We do not claim these as new. The search was not exhaustive.

3 Setting and consent rule

Each case contains one consent grant and one completed share. A share is **authorised** if and only if:

1. the grant and share match on document identifier, recipient identifier and purpose;
2. $t_{\text{grant}} \leq t_{\text{share}} < t_{\text{expiry}}$;
3. no revocation is effective at execution, i.e. there is no revocation with $t_{\text{revoke}} \leq t_{\text{share}}$.

Times are simulated integers. Expiry and revocation take effect at their recorded time, including when that equals the share time. A known absence of revocation is recorded explicitly as `{"status":"not_revoked"}`. An omitted revocation field is *missing evidence*. Grant and share facts are trusted values from the mock application, and tampering is out of scope.

4 Fixtures

Table 1 lists the ten hand-specified pairs. Within each pair, every detail except the varied fact is the same. The pairs include the two boundary cases the design requires, a share exactly at expiry (P07) and a revocation exactly at execution (P09), plus two more: a share exactly at grant time (P04) and a revocation recorded after the share, which is therefore not yet effective (P03). Across the recipient pairs, the consent holder rotates through A, B, C and D. No two pairs share the same

Table 1: Matched pairs. Times are integers. NR means explicitly not revoked; "rev@t" means revoked, effective at t.

Pair	Family	Purpose	Grant	Expiry	Revocation	Share	Authorised vs. unauthorised
P01	recipient	case review	10	30	NR	20	to A vs. to B (grant: A)
P02	recipient	claim assessment	5	50	NR	12	to B vs. to A (grant: B)
P03	recipient	account onboarding	100	200	rev@180	140	to C vs. to D (grant: C)
P04	recipient	complaint follow-up	40	90	NR	40	to D vs. to C (grant: D)
P05	document	audit preparation	10	60	NR	30	D5 vs. sibling file D6
P06	document	billing dispute	200	400	NR	250	D7 vs. unrelated file D8
P07	expiry	case review	10	31 / 30	NR	30	expiry 31 vs. 30 (= share)
P08	expiry	renewal check	0	60 / 40	NR	45	expiry 60 vs. 40
P09	revocation	fraud review	10	100	rev@21 / @20	20	rev@21 vs. rev@20 (= share)
P10	revocation	settlement review	50	150	NR / rev@60	70	NR vs. rev@60

combination of purpose and timeline, so no inputs collide across pairs. Individual values do recur: P01 and P07, for example, both use "case review" and grant time 10.

Each record holds 6 planted fictional strings, or 8 for unauthorised recipient and document runs:

- 2–3 raw identifiers;
- 2–3 names or titles;
- 2 content strings: the document excerpt and the share message.

Expected verdicts and rationales live in a separate manifest and are joined to reviewer outputs only after the reviewer has run. Case identifiers are opaque hashes, and records are stored in identifier order, so neither an identifier nor a record's position reveals its label.

Five sanity cases leave the verdict unchanged while changing a name or title (three cases), the stored key order (one case, seeded) or the identifier format (one case).

5 Log views

All views are generated from the same underlying record. They drop case and run identifiers, preserve event order (grant, then share) and are serialised as canonical JSON (sorted keys, no whitespace, ASCII). The reviewer receives exactly these bytes, and the identical-input comparisons use them too.

A. Full evidence. All synthetic raw identifiers, names, contents and facts.

B. Uniform masking. Document and recipient identifiers become `[MASKED:document_id]` and `[MASKED:recipient_id]`. Names and contents become `[MASKED:name]` and `[MASKED:content]`. Action, outcome, purpose and timing are kept.

C. Consistent aliases. Text is masked as in B. Identifiers become `DOC_ALIAS_`n and `RCP_ALIAS_`n, numbered by first appearance in a fixed traversal: events in order, then document before recipient within each event. Each type has its own counter, and the counters reset for every run.

C ablations. `C_no_expiry` drops the grant's `expires_at`. `C_no_revocation` drops the whole revocation object, including an explicit "not revoked".

Private fields. The private fields are document and recipient identifiers, titles, names, excerpts and messages. Purpose and timestamps are kept as non-private policy facts. This choice was made

and recorded before the main run.

6 Reviewer

The reviewer is a standalone script that imports only `json` and `sys`. It scores six conditions as *pass*, *fail* or *unknown*: document match, recipient match, purpose match, at or after the grant, before expiry, and not revoked. Three rules decide each condition:

- a uniform mask is *unknown*, even when the grant's and the share's masks are equal strings;
- aliases and raw identifiers are compared by equality;
- a missing field is unknown.

Any *fail* yields **unauthorised**, six *pass*es yield **authorised**, and anything else yields **insufficient evidence**.

Each call runs in a fresh temporary directory that holds only a copy of the script, under `python3 -I` with an empty environment, with the input on standard input. Calls run in a seeded random order. This is process-level isolation, not an operating-system sandbox. Two checks guard against a reviewer that consults anything beyond its input: a static allowlist on the reviewer's imports, attributes and calls, and the invariant that its correct count cannot exceed the ceiling. A supplementary diagnostic, not pre-specified, runs a *naive* variant that treats equal mask strings as matching identities.

7 Metrics

Let $N = 20$ be the number of runs.

1. **Identical pairs:** matched pairs whose two exact inputs are byte-identical while their labels differ (out of 10, and by family).
2. **Ceiling:** partition the runs into groups g of identical inputs, with a_g authorised and u_g unauthorised runs in each. The ceiling is $\frac{1}{N}\sum_g \max(a_g, u_g)$: the highest accuracy any forced binary decision based only on the input can reach *on this fixed set*.
3. **Reviewer outcomes:** correct (of 20), missed violations (unauthorised judged authorised, of 10), false alarms (of 10), insufficient evidence (of 20, split by label) and definitive-answer coverage.
4. **Direct exposure:** distinct planted values that appear verbatim, or JSON-escaped, in the exact input.

Three properties of the ceiling.

(i) **Merging never raises it,** since $\max(a + a', u + u') \leq \max(a, u) + \max(a', u')$.

(ii) **For views in which every unauthorised run that shows no failing condition is byte-identical to its authorised partner** (true of all five views here), **it equals** $(n_{\mathbf{auth}} + d)/N$**,** where d counts unauthorised runs whose own input shows a failing condition.

- If an unauthorised run shows a failure, every byte-identical run shows it too, so its group is entirely unauthorised.

- Otherwise, by the condition, its input equals its authorised partner's, so its group has at least as many authorised runs as unauthorised ones.

Collisions between pairs therefore leave the ceiling unchanged. Because d is exactly the number of violations the faithful reviewer detects, the run checks this identity in every view. The condition is not automatic: a mask that carries a per-event random value makes every input distinct, so the ceiling reaches 20/20 while the reviewer still detects only the four timing violations.

(iii) **On a collapsed matched pair,** any forced binary rule gets exactly one of the two runs wrong, either as a missed violation or as a false alarm.

8 Validation

What the 91 automated checks cover. All passed in the reported run.

- Every hand label matches a separately implemented trusted-fact oracle.
- Each pair changes only its intended fact, and each unauthorised case fails exactly one condition.
- The boundary cases sit on their intended pairs.
- No reviewer input contains case, pair or run identifiers or label words.
- The exposure scanner finds 66/66 planted values in view A.
- The reviewer passes the static allowlist.
- The full-view reviewer matches all 20 labels. This is a gate: the run stops if it fails.
- A second complete pass, in a different shuffled order, reproduces every input and verdict.
- All five sanity cases behave as specified.
- The metric invariants hold in every view.
- The stored data match the fixture specification, and every frozen file matches `FREEZE.json`.
- 30 unit tests pass, including scoring tests with hand-computed answers.

Two checks are weak and are kept only as smoke tests: same-process re-rendering, and independence from the source key order, which sorted keys make nearly certain.

Mutation tests. We confirmed that five deliberate faults are each caught:

- an edited fixture;
- an alias table that persists across runs;
- a reviewer that always passes purpose;
- scoring that counts abstentions as missed violations;
- a reviewer that reaches the filesystem through `json.codecs`.

Table 2: Results per view (20 runs, 10 pairs). R/D/E/V: identical pairs by family (of 4/2/2/2). The 66 planted values are 22 identifiers, 22 names or titles and 22 content strings.

View	Identical pairs	R/D/E/V	Ceiling	Correct	Missed viol.	False alarms	Insuff. evidence	Auth. confirmed	Planted visible
A full	0/10	0/0/0/0	1.00	20/20	0/10	0/10	0/20	10/10	66/66
B uniform mask	**6/10**	4/2/0/0	0.70	4/20	0/10	0/10	16/20	**0/10**	0/66
C aliases	0/10	0/0/0/0	1.00	**20/20**	0/10	0/10	0/20	10/10	**0/66**
C, no expiry	2/10	0/0/2/0	0.90	8/20	0/10	0/10	12/20	0/10	0/66
C, no revocation	2/10	0/0/0/2	0.90	8/20	0/10	0/10	12/20	0/10	0/66
Supplementary: B, naive mask equality				14/20	**6/10**	0/10	0/20	10/10	–

Independent review. A separate AI agent that had not written the code reviewed version 1.0. It recomputed every metric and reproduced all 100 recorded verdicts with its own code. It confirmed byte-identical results under Python 3.12–3.14 and different hash seeds. It also found gaps in enforcement and wording, which version 1.1 fixes; `PROTOCOL.md` lists every change.

Pre-registration. The v1.0 freeze, made before the main run, hashed the protocol, fixtures and manifest but not the code. File timestamps are consistent with no code changes after it, but the release cannot prove this. The v1.1 freeze covers all code. The freeze is self-attested: `FREEZE.json` is written by the author, and the v1.0 hashes refer to files with the original case identifiers, which are not in the release.

Effect of the revision. No reported number changed between v1.0 and v1.1: the summary tables are byte-identical. Byte-identity of results across separate invocations was verified with `diff`, outside the automated checks.

9 Results

Table 2 reports every view. The faithful reviewer made no missed violations and no false alarms in any view. Every loss is an abstention, and in these five views every definitive answer was correct. That is a property of the views, which either keep a relation exactly or announce that they removed it; a transformation that silently changes values can produce confident wrong verdicts.

By family.

- **View B** abstained on all 12 recipient and document runs. In the expiry and revocation families it detected 2/2 violations each but confirmed 0/2 authorised shares each: a known timing failure establishes a violation, but a masked identity can never be confirmed.
- **C without expiry** abstained on all four expiry runs, yet still detected every recipient, document and revocation violation (8/8).
- **C without revocation** mirrors this, abstaining on the revocation family instead.

Hypotheses. All three pre-specified predictions matched exactly.

- **H1:** masking collapses identity-dependent pairs.
- **H2:** aliases restore those pairs with zero direct exposure.

- **H3:** omitting temporal evidence leaves decisions unresolved. The prediction, which held, was that all 10 authorised cases would be unresolved, because confirmation requires every condition to be known.

Mixed groups and sanity cases. The only identical-input groups with mixed labels were the predicted within-pair groups: P01–P06 under B, P07–P08 without expiry, and P09–P10 without revocation. All five sanity cases kept their verdict in every view. Masked and aliased inputs were byte-identical to the base case. The full view was identical only under the key-order change.

Supplementary: shared timeline. We moved every pair onto P01's purpose and timeline, keeping only its varied fact. This analysis was added after review, was not pre-specified, and applies the reviewer in-process rather than in isolated calls. Distinct inputs fell to 6 (B), 8 (C), 6 (no expiry) and 5 (no revocation), but every ceiling (20, 14, 20, 18, 18 out of 20) and every reviewer count stayed the same, as property (ii) predicts.

Supplementary: naive reviewer. A reviewer that reads equal masks as equal identities reaches the B ceiling (14/20) with full coverage. It gets there by clearing all 6 recipient and document violations, none of which shows another failure.

10 Worked examples

Identity (P01 and P05, B vs. C). In P01 the grant covers D1 for recipient A, valid from 10 to 30, and the share happens at 20.

- **Under B,** both runs read `[MASKED:recipient_id]` → `[MASKED:recipient_id]`. The inputs are byte-identical, so a reviewer that sees only its input must treat the two runs alike.
- **Under C,** the authorised run reads `RCP_ALIAS_1` → `RCP_ALIAS_1` and the unauthorised run reads `RCP_ALIAS_1` → `RCP_ALIAS_2`. Both are resolved correctly, with 0/6 and 0/8 planted values visible.

The document pair P05 behaves the same way: `DOC_ALIAS_1` → `DOC_ALIAS_1` against `DOC_ALIAS_1` → `DOC_ALIAS_2`. The reviewer needs to know whether the grant and the share name the same party or document, not which one it is.

Distinguishability versus verifiability (P07 without expiry). The share occurs at 30, and the grant expires at 31 (authorised) or exactly at 30 (unauthorised). With expiry present, C resolves both runs, including the boundary case, because expiry takes effect at its recorded time. Without expiry, the two inputs are identical.

The ablation's ceiling is 0.90, yet the faithful reviewer resolves only 8/20. The rule "authorised unless a visible condition fails" reaches the ceiling (18/20) because each pair varies only one fact, so it exploits the fixture design rather than verifying anything. A reviewer bound by the rule cannot confirm any share without knowing the expiry.

Explicit absence versus missing evidence (P10 without revocation). The authorised run records `not_revoked`, and the unauthorised run records a revocation at 60, before the share at 70. Once the revocation object is removed, the two runs are indistinguishable. A log that expresses "no revocation" only by leaving out a field cannot be told apart from one whose revocation evidence was redacted.

11 From pilot to a reusable method

The pilot hard-codes one rule, one log schema and five views, so no other team could run it on its own system. We therefore rebuilt the comparison as a small toolkit, `redaction_audit`. It is standard-library Python, takes four input files and runs as one command. Three ideas carry over from the pilot.

Evidence needs. Every condition of an authorisation rule depends on one kind of evidence (Table 3). A redaction scheme supports auditing a rule when, for every condition, it keeps the evidence that condition needs, or declares that it removed it. Users write their rule as a list of conditions, each with a need type and the fields it relates. The toolkit derives the reviewer, the generated violations and a checklist of evidence requirements from that list.

Table 3: Evidence needs. "Kept by" and "Lost by" summarise the toolkit's built-in reviewer semantics.

Need	Question	Kept by	Lost by
equality	Same person or document in two events?	raw values; aliases or keyed pseudonyms consistent over the run (or globally)	uniform masks; per-event pseudonyms; dropping
order	Did A happen before B?	exact times; within-run ranks	dropping; masking; coarsening at the boundary
explicit state	Was the authority revoked, closed or suspended at the time?	the state recorded explicitly, with its effective time	omission standing for the "clear" state
raw value	Is a value in an allowed set?	the value, or a non-identifying category	masking; pseudonymising
presence	Does a required record exist?	the field, even with a masked value	undeclared dropping

Declared redaction and two readings. Every built-in transform records what it did to each field in a `redaction` header that travels with the log. The header names the method: mask, alias, pseudonym with its scope, bucket with its size, rank, or drop. Custom redactors can supply the same declarations. The built-in reviewer reads the log in one of two ways:

- the *declared* reading uses the header, for example by treating a bucketed time as an interval;
- the *undeclared* reading compares values literally, as a person or model unaware of the scheme would.

Each condition and scheme then receives one of four statuses:

- *verified*: every case decided correctly;
- *unverifiable*: some cases left undecided;
- *unsafe*: at least one wrong verdict;
- *untested*: no case violates the condition.

Generated matched pairs. Users supply only authorised logs. For each log and condition, the toolkit applies a mutation that breaks exactly that condition, by default at the boundary: expiry set equal to the action time, revocation effective exactly at the action, the grant opened one tick after the action. It then checks that the mutated log fails that condition and no other, and rejects the mutation otherwise. Users can override a mutation with realistic decoy values. This removes the main cost of reuse, which is hand-writing matched pairs, and supplies boundary cases automatically.

Expressiveness and safeguards. Several further features cover common rule shapes and catch common input mistakes:

- **Four-eyes rules.** An equality condition can require two values to *differ*.
- **Numbers.** Order conditions compare any numbers, such as times or amounts.
- **Custom categories.** Users can mark their own categories of private field, for example amounts.
- **Partial masking.** A built-in op keeps the last characters of a value, as card and IBAN masking do; different kept endings prove a mismatch, equal ones prove nothing.
- **Realistic default violations.** Identifiers are changed in their first character, so the ending is kept and redaction that keeps the last characters is exposed. Several violations per condition, per-case decoys, and a mix of hand-written and generated pairs are supported.
- **Strict input checks.** Unknown keys, field selectors that match nothing and roles that match zero or several events stop the run with a precise message.
- **Both readings by default.** The built-in reviewer reports the declared and the literal reading side by side, names the cases behind every problem, and breaks down exposure by field.
- **CI gates.** A gate can fail on wrong verdicts, on undecided cases, on wrong verdicts once the header is lost, or on any visible private value.

Regression check. Given the pilot's 20 cases and the five views as data files, the toolkit reproduces the pilot's summary and family tables byte for byte. That shows the generalisation did not change the measurements. The toolkit has 29 tests of its own and runs in continuous integration with the pilot.

Exploratory results: common redaction choices

These results were not pre-specified. They use the pilot's 20 cases, with six further views built on view C's text masking (Table 4).

Four observations follow.

1. **Per-event pseudonyms look perfect on the distinguishability metrics but break verification.** No pair is identical and the ceiling is 20/20. Yet the declared reviewer can verify only the timing violations, and the undeclared reviewer flags every authorised share. This is the behaviour of a widely used redactor's hash operator, whose documentation states that the same value gets different hashes across entities and calls unless a salt is provided [20]. Its default *replace* operator, which substitutes the entity type, corresponds to view B. Identical pairs and the ceiling are therefore not safety indicators. The status, which counts wrong verdicts, is.

Table 4: Common redaction choices on the pilot cases (20 runs). The declared reviewer reads the redaction header; the undeclared reviewer compares values literally. Status is the worst status over the six conditions.

			Declared reviewer		Undeclared reviewer		
View	Identical pairs	Ceiling	Correct	Status	Correct	Wrong verdicts	Status
B uniform mask (type labels)	6/10	14/20	4/20	unverifiable	14/20	6 missed	**unsafe**
D hash, salt per event	0/10	20/20	4/20	unverifiable	10/20	10 false alarms	**unsafe**
E keyed hash per run	0/10	20/20	20/20	verified	20/20	0	verified
F keyed hash, global	0/10	20/20	20/20	verified	20/20	0	verified
G times bucketed to 10	2/10	18/20	14/20	unverifiable	18/20	2 false alarms	**unsafe**
H times as within-run ranks	0/10	20/20	20/20	verified	20/20	0	verified
I purpose masked	0/10	20/20	10/20	unverifiable	20/20	0	(untested)

2. **This rule needs the order of times, not their values.** Within-run ranks keep every verdict.
3. **Coarsening times, a natural defence against timestamp linkage, misleads a literal reader.** It does so exactly at the boundary pairs (P07, P09). A declared interval reading abstains there instead.
4. **A literal reader passes every masked purpose.** No pilot case violates purpose, so this error cannot surface (*untested*). That is why generated pairs cover every condition.

A second policy and log shape

The toolkit's starter audit uses a different rule and a different log shape: a support agent reading a customer record, in OpenTelemetry-style spans with an unrelated LLM span in each run. The rule exercises all five needs:

- agent and customer match the ticket assignment (equality);
- the read happens after the assignment opens and before its scheduled close (order);
- the ticket was not closed at or before the read (explicit state);
- the agent's role is allowed (raw value).

From 6 authorised logs it generates 36 boundary violations.

Results under the declared reviewer:

- Per-run aliases, per-run keyed hashes and time ranks verify every condition.
- Type labels and per-event hashes lose the identity conditions.
- Minute buckets lose every order and state condition at the boundary.
- Dropping the ticket's status loses the explicit-state condition.
- Masking the role loses the raw-value condition.
- An example custom redactor replaces customer identifiers with `<CUSTOMER>` without declaring it. It is *unsafe*, approving all 6 misdirected reads. Because it forgets agent and ticket identifiers, the exposure scan also shows 12 of 38 planted values still visible.

A third example: a payment agent

A payment agent executes supplier payments from logs with three event types: approval, vendor-master check and payment. The rule has 12 conditions and uses all five needs:

- same invoice, same cost centre and same currency as the approval;
- an amount within the approved amount;
- approval before payment and before its expiry;
- no withdrawal by the time of payment;
- a manager as approver;
- four-eyes (approver differs from requester);
- a payee IBAN that matches the vendor check, and a payee on the allowlist;
- an attached invoice document.

From 5 authorised logs, one of which contains a failed attempt and a retry, the toolkit generates 60 boundary violations. Results by view:

- **The log vendor's default type masking** loses the invoice, amount, four-eyes and payee checks.
- **An hourly time bucket in the current export** leaves the time conditions undecidable near their limits.
- **An in-house regex redactor** keeps IBAN endings without declaring it. It is *unsafe*, passing 4 payments to lookalike IBANs, and leaves 44 of 81 private values visible.
- **Per-run keyed pseudonyms, with masked text and with times and amounts replaced by their within-run ranks,** verify all 12 conditions under both readings, with 0 of 81 private values visible.

The policy, cases and redactor were written during the usability test described below.

A usability test

An AI agent acting as a first-time user followed the tutorial and audited a payment policy of its own design. It completed the audit and the CI gate, but estimated 2.5–4 hours for a person's first attempt, not the hour the draft tutorial suggested. It also found several problems:

- ignored keys that could invert a rule;
- field selectors that matched nothing yet showed "verified";
- integer-only comparisons, with a misleading error message;
- generated violations that a keep-the-last-four redactor could not miss;
- outputs that did not name the failing cases.

All of these are fixed in v1.2, each with a regression test. No outside team has used the toolkit yet.

Fit with existing formats

Three widely used formats leave out evidence that these needs require.

- **OpenTelemetry.** The GenAI semantic conventions, which are still marked as in development, make a tool call's arguments and result opt-in, flag them as potentially sensitive, and define no attribute for an authorisation decision [21]. Unless the evidence fields are promoted to dedicated attributes, a trace may not record the recipient and document at all.
- **OCSF.** OCSF 1.9.0's `delegation` object, "a durable authorization context that a principal issues to a delegate", has only an identifier, a creation time, an issuer and a parent [22]. An event that carries only this object is in the position of our ablated views: expiry and revocation must come from elsewhere.
- **ISO/IEC TS 27560 (through W3C DPV).** The community guide for 27560 consent records models withdrawal as a change of consent state [23]. That supports recording the state in force at the time of the action explicitly, as the explicit-state need asks.

The repository maps these formats to the toolkit's files.

12 Discussion

What the reviewer needs. The pilot shows by removal that verifying this rule needs three things in the log:

- equality-preserving references for the document and recipient within the run;
- the expiry;
- an explicit revocation state.

Purpose and grant time are needed by the rule's definition; they were never removed here. Raw identifiers, names and contents are not needed: view C verifies all 20 runs without them. Whether such a log is *safe* to share is a separate question that we did not measure.

How uniform masking fails. Uniform masking fails because it erases equality, not because it hides values. Its failure is also asymmetric: timing violations remain detectable, but no authorised share can ever be confirmed. The naive diagnostic shows the opposite hazard: a reviewer that reads equal masks as equal identities clears every misdirected share that shows no other failure.

Scope of consistency. Our within-run aliases make the consistently pseudonymised set one run: the smallest set that still lets a reviewer compare the grant with the share. That is in the spirit of the EDPB's recommendation to keep such sets "as small as possible" [18]. The same guidance prefers per-transaction pseudonyms. If the grant and the share count as separate transactions, those would give the same party different pseudonyms in the two events. In the exploratory runs (§11), pseudonyms salted per event indeed made the identity conditions unverifiable under a declared reading. Under a literal reading they turned every authorised share into a false alarm.

Scope of the exposure result. Zero direct exposure for B and C holds by construction, and the scanner counts only literal planted strings. It is therefore a lower bound: format-preserving substitutes, for example, can leak information about the original value [16]. The views also keep purposes, exact timestamps, event structure and, in C, the pattern of equalities within a run. Someone holding other records with matching timestamps could join on them and recover identities, as attacks that exploit unchanged trace fields illustrate [14, 15]. Purposes such as "fraud review" can themselves be sensitive. Resetting aliases for every run blocks linking across runs, but it also

makes cross-run questions unanswerable, such as whether one recipient received many documents. We did not measure this trade-off.

13 Limitations

- **Size and construction.** There are 20 hand-written runs, each with one grant and one share. All numbers are descriptive counts for these cases. Most follow from the design; byte-identity under uniform masking, for example, is guaranteed whenever a pair differs only in an identifier.
- **Fixed design choices.** One fact varies per pair, masks name themselves, clocks are integers and facts are trusted.
- **Untested conditions.** Purpose and grant-start conditions are implemented and unit-tested in every view, but no matched pair varies them.
- **Reviewer.** The reviewer is rule-based. Its isolation is process-level rather than an operating-system sandbox.
- **Exposure.** The scanner counts exact substrings only; it does not detect fragments, transformed values or inference.
- **Shortcuts a model could exploit.** Before any model study, two shortcuts must be removed. In view C, the token `_ALIAS_2` alone marks every identity violation. In view A, the document-family violations carry a message that does not fit the file sent.
- **Pre-specification.** The freeze is self-attested, and the v1.0 freeze did not cover the code (see Validation).
- **Agent framing.** Nothing in the method is specific to AI agents: there is no model, no free-text tool argument and no multi-step trajectory. We use an agent's action log because that is where the motivating problem arises.
- **Exploratory results.** The results in §11 were not pre-specified.
- **Toolkit scope.** The toolkit evaluates one authority event and one action per run, a conjunction of conditions, and literal exposure only. Its second example is synthetic. Its usability has been tested only by an AI agent acting as a first-time user, which estimated 2.5–4 hours for a person's first audit. It has not yet been used by outside teams.
- **Literature search.** The search was not exhaustive.

14 Future work

The following ideas are untested.

- **A local-model reviewer.** It would use the same views, a frozen prompt and schema, development and test templates kept separate, and repeated runs analysed as repeated measures. Logs should contain several events, so that alias numbering is not a giveaway. In forensic reconstruction of agent logs, "identifier renaming alone provided no consistent remedy" and explicit record bindings helped [11], which suggests that model reviewers may use aliases less reliably than our rule does.

- **More families and log shapes.** Purpose and grant-start families in the pilot; runs with several authorities and actions; delegated or partial grants; identifiers embedded in free-text tool arguments.
- **A linkage measure.** Measuring how pseudonym scope (event, run, global) and retained timestamps allow records to be joined across runs, to complement the verification measure.
- **Persistent pseudonyms.** A per-transaction or keyed-persistent pseudonym view that measures cross-run audit utility against linkage risk, together with a re-identification test that joins retained timestamps to auxiliary records.
- **Richer exposure measures,** based on fragments and inference.
- **Grounded labels.** Labels based on published consent-record structures [19] or on contextual-integrity norms [10].

Availability and reproducibility

The code, fixtures, manifest, protocol with revision log, freeze history and all outputs are at `https://github.com/Anishpras/redaction-audit-pilot`, archived on Zenodo at `https://doi.org/10.5281/zenodo.24163831` (all versions; version 1.2.0: `https://doi.org/10.5281/zenodo.24163832`). Running `python3 run_pilot.py` reproduces every number in the pilot. Running `python3 -m redaction_audit` reproduces the exploratory results and runs the toolkit on new rules (see `docs/tutorial.md`).

- **Environment:** Python 3.13.7 on macOS (arm64), about 7 seconds, 290 isolated reviewer calls across two passes.
- **Versions and seeds:** fixture version 1.1.0; sanity seed 20261112; call-order seeds 20261113 and 20261114.
- **Licences:** code under MIT; the paper, protocol, fixtures and results under CC BY 4.0.

All records are fictional.

Acknowledgements and AI-assistance statement

The experiment code, analysis and manuscript were prepared with substantial assistance from Claude (Anthropic), an AI assistant, working under the author's direction. Separate AI agents carried out the independent code review, the citation verification and the literature search described above. The author is responsible for the content. The work used no paid API, cloud compute or external funding.

References

[1] Y. Nian, A. Yuan, H. Zhang, J. Li, L. Li, X. Hu, H. Wei, X. Xiao, C. Xiao, and Y. Zhao. Auditable agents. arXiv:2604.05485v4 [cs.AI], 27 Sep. 2026. A condensed version was accepted to ACM AI Summit 2026 (Visionary Papers track). `https://arxiv.org/abs/2604.05485`

[2] D. Garg, L. Jia, and A. Datta. Policy auditing over incomplete logs: Theory, implementation and applications. In *Proc. ACM CCS*, 2011. `https://doi.org/10.1145/2046707.2046726`

[3] O. Chowdhury, D. Garg, L. Jia, and A. Datta. Equivalence-based security for querying encrypted databases: Theory and application to privacy policy audits. In *Proc. ACM CCS*, 2015. arXiv:1508.02448. https://arxiv.org/abs/1508.02448

[4] J. Biskup and U. Flegel. Transaction-based pseudonyms in audit data for privacy respecting intrusion detection. In *Proc. RAID 2000*, LNCS 1907, pp. 28–48, Springer, 2000. https://doi.org/10.1007/3-540-39945-3_3

[5] K. Lakkaraju and A. Slagell. Evaluating the utility of anonymized network traces for intrusion detection. arXiv:0712.1224, 2007 (published at SecureComm 2008). https://arxiv.org/abs/0712.1224

[6] E. Najt, C. Toft, T. Tracy, F. Roger, and J. Benton. SLEIGHT-Bench: Finding blind spots in AI monitors. Alignment Science Blog (Anthropic), 19 May 2026. https://alignment.anthropic.com/2026/sleight-bench/

[7] E. Najt, C. Toft, T. Tracy, F. Roger, and J. Benton. SLEIGHT-Bench: A benchmark of evasion attacks against agent monitors. arXiv:2605.16626v2, May 2026. https://arxiv.org/abs/2605.16626

[8] H. Xia and Y. Park. SlotGuard: Stop oversharing private local context in LLM agent transcripts. In *Second Workshop on Agents in the Wild (AIWILD) at ICML 2026*. arXiv:2607.17147v2 [cs.CR], 9 Nov. 2026. https://arxiv.org/abs/2607.17147

[9] S. Hu, L. Liu, Z. Meng, and Z. Zhao. ToolPrivacyBench: Benchmarking purpose-bound privacy in tool-using LLM agents. arXiv:2606.28061v1, 26 Jun. 2026. https://arxiv.org/abs/2606.28061

[10] C. Guo, Z. Chen, Z. Lin, X. Liu, and B. Li. AgentCIScope: Grounded contextual integrity evaluation for LLM agents. arXiv:2610.19158v1 [cs.AI], 13 Oct. 2026. https://arxiv.org/abs/2610.19158

[11] T. Yun, D. Kim, G. Kim, J. Seo, M. Hur, and M. Min. Correct answers, unsupported findings: Evidence binding in forensic reconstruction of LLM agent logs. arXiv:2610.09581, Oct. 2026. https://arxiv.org/abs/2610.09581

[12] M. Rafiei and W. M. P. van der Aalst. Privacy-preserving data publishing in process mining. In *Business Process Management Forum (BPM Forum 2020)*, LNBIP, Springer, pp. 122–138, 2020. https://doi.org/10.1007/978-3-030-58638-6_8

[13] G. Elkoumy, A. Pankova, and M. Dumas. Differentially private release of event logs for process mining. *Information Systems*, 115:102161, 2023. https://doi.org/10.1016/j.is.2022.102161

[14] M. Mohammady, L. Wang, Y. Hong, H. Louafi, M. Pourzandi, and M. Debbabi. Preserving both privacy and utility in network trace anonymization. In *Proc. ACM CCS*, pp. 459–474, 2018. https://doi.org/10.1145/3243734.3243809

[15] S. E. Coull, C. V. Wright, F. Monrose, M. P. Collins, and M. K. Reiter. Playing devil's advocate: Inferring sensitive information from anonymized network traces. In *Proc. Network and Distributed System Security Symposium (NDSS)*, 2007.

[16] M. S. Boesgaard and M. Larsen. Format-preserving encryption creates a privacy attack surface for re-identification. Cryptology ePrint Archive, Paper 2026/993, 2026. https://eprint.iacr.org/2026/993

[17] ENISA. Pseudonymisation techniques and best practices: Recommendations on shaping technology according to data protection and privacy provisions. European Union Agency for Cybersecurity, Nov. 2019. https://doi.org/10.2824/247711

[18] European Data Protection Board. Guidelines 01/2025 on pseudonymisation. Adopted 16 Jan. 2025, version for public consultation, paras. 115–121. https://www.edpb.europa.eu/system/files/2025-01/edpb_guidelines_202501_pseudonymisation_en.pdf

[19] ISO/IEC. ISO/IEC TS 27560:2023 Privacy technologies — Consent record information structure. Edition 1, Aug. 2023. https://www.iso.org/standard/80392.html

[20] Presidio project. Presidio Anonymizer documentation (operators *replace* and *hash*). Accessed 12 Nov. 2026. https://presidio.dataprivacystack.org/anonymizer/

[21] OpenTelemetry. Semantic conventions for generative AI: execute tool span (status: Development). Accessed 12 Nov. 2026. https://github.com/open-telemetry/semantic-conventions-genai/blob/main/docs/gen-ai/execute-tool.md

[22] OCSF. Open Cybersecurity Schema Framework 1.9.0, *delegation* object. Accessed 12 Nov. 2026. https://schema.ocsf.io/objects/delegation

[23] W3C Data Privacy Vocabularies and Controls Community Group. Consent records and receipts as per ISO/IEC TS 27560:2023 using DPV. Final Community Group Report, 15 Feb. 2026. https://w3c-cg.github.io/dpv/guides/consent-27560