International Journal of Research and Development in Engineering Sciences
ISSN: 2582 - 4201(Online) , http://www.ijrdes.com/
Regular Issue, Vol. 8, Issue. 6 (November – December) , 2026 , Pages: 1 - 8
Received: 18 February 2026 ; *Accepted:* 29 October 2026 ; *Published:* 06 November 2026
https://doi.org/10.63328/IJRDES-V8RI6P1
https://www.youtube.com/watch?v=9mLE2qC9WN4
https://archive.org/details/jsp-ijrdes-v8ri6p1
Original Paper : Applied Research – Experimental Research Paper

Anchoring Trust in Agentic AI: A Cost and Security Evaluation of Tamper-Evident Audit Logs

Hema Dega

Department of Department of Electrical Engineering , New York Institute of Technology , New York, USA

** Corresponding Author : Hema Dega ; hemadega@gmail.com*

Abstract: If an AI agent's actions are logged, the log is only useful if nobody can quietly change it afterwards. A hash chain makes edits visible, but a determined insider who holds the signing key can rebuild the whole chain, and someone who simply cuts off the end of the log leaves a chain that still verifies. The standard remedy is to send periodic signed summaries, called anchors, to an outside witness. This paper works out what that costs and what it buys. We implemented a signed hash-chained log with Merkle batches, inclusion and consistency proofs, and an external witness, using real Ed25519 signatures and SHA-256, and attacked it with 300 real rewrites, edits and truncations for each of seven anchoring settings. We prove that the window of records an attacker can alter undetected is exactly the set of records since the last anchor (or last signed checkpoint, if the attacker has no key), and the attacks bore this out in every case. Anchoring cost and window length are tied by a simple invariant: the fraction of time spent anchoring times the window in seconds equals the cost of one anchor. Signing every record cost 54 µs against 8.4 for chain-only logging, but signing every 100 records was within measurement noise of chain-only. A full audit of 100,000 records took about 313 ms, while one Merkle inclusion proof for a million records was 20 hashes and verified in 24 µs. Without a witness, auditors who gossip can still catch a fork, and we give an exact formula for how long that takes. Finally, anchoring right after high-risk records beats uniform anchoring exactly when the high-risk records carry more than their share of the risk, and we give the break-even ratio. Timings come from one machine and the witness latency is modelled.

Keywords: Audit Logs, Merkle Trees, External Anchoring, Transparency Logs, Agent Accountability.

1. Introduction

When an agent can send email, move money or change a production system, the first thing a careful team wants is a record of what it did. The second thing, usually after the first incident, is a way to be sure the record has not been touched. An audit log that its operator can rewrite is evidence of whatever the operator would like it to say. Tamper evidence has a long history. Haber and Stornetta showed how to chain time-stamped documents so that changing one invalidates the rest [1], and Schneier and Kelsey described how to protect the log on an untrusted machine using keys that evolve [2]. Merkle trees [3] let one verify that a single entry is in the log without reading all of it. Crosby and Wallach showed how to build efficient tamper-evident logs from them [4], and Certificate Transparency made the idea routine on the web [5].

The prototype in the first paper of this series, on evidence-based delegation, found that a hash chain with a signed head detects every in-place edit, deletion and reordering we tried, and misses truncation of the tail un-less an external checkpoint is kept. This paper takes that observation seriously and asks three practical questions. How much does it cost to sign, batch and anchor a log? Exactly what can an attacker still do between anchors? And what should an operator do who has no trusted outside witness, or who wants to spend the anchoring budget where it matters most? We make five contributions.

- A working implementation with real signatures, Merkle batches, and tested inclusion and consistency proofs, with measured costs for appending, auditing and proving (Sections 5 and 6).
- A precise statement of what an attacker can change undetected, as a function of how often the log is signed and anchored, verified against 6,300 real attacks (Sections 4 and 7). A cost–window invariant for

Jack Sparrow Publishers © 2026, IJRDES , All Rights Reserved
www.jacksparrowpublishers.com

ISSN: 2582 - 4201(Online) , http://www.ijrdes.com/ , IJRDES, Vol. 8, Issue 6 , 2026 , https://doi.org/10.63328/IJRDES-V8RI6P1

anchoring (Proposition 2). An exact formula for how quickly auditors who gossip catch a fork when there is no witness (Proposition 4).

- A rule for when to anchor right after high-risk records instead of at regular intervals (Proposition 5).

Timings come from a single sandbox machine, the witness is simulated in process, and the latency to a real witness is a modelled parameter. We say clearly where we measured and where we assumed.

2. Background and Research Gap

Chains And Keys.: A hash chain links each record to the hash of the previous one. By itself it proves only that the records from some point on are consistent with each other. Schneier and Kelsey added message authentication keyed per entry, and key evolution so that a later compromise cannot forge earlier entries [2]. Our experiments assume the attacker has the current signing key, which key evolution would not prevent from rewriting the unanchored suffix.

Merkle Trees: A Merkle tree commits to a list of entries with one root hash. An inclusion proof of ⌈log2 n⌉ hashes shows that an entry is in a tree of n entries, and a consistency proof shows that an earlier tree is a prefix of a later one [4, 5]. We implement both in the form used by Certificate Transparency.

Anchoring and Witnesses: Publishing a signed summary to a party the log operator cannot control fixes the log's history up to that point. Time-stamping services work this way [1], and transparency logs add independent monitors and gossip so that the operator cannot show different histories to different clients [5]. We study anchoring as a design parameter, with measured costs.

Signatures. We use Ed25519 [6], which is fast and has short signatures. The measurements below include its real cost.

Where this fits: Nothing here is new cryptography. What we add is a measured account of the costs and a clean statement of what remains attackable at each setting, in the context of logging agent actions, where the log is the evidence on which later review depends.

3. Design and Threat Model

3.1. The log

Each record is a small JSON object with an index and a payload (the agent's action and decision). A leaf hash is H(0x00 record), and a running chain value is updated as

$$\text{leaf}_i = H(0\text{x}00 \parallel \text{record}_i), \; c_{i+1} = H(c_i \parallel \text{leaf}_i)$$

with SHA-256. Every B records the logger signs a checkpoint (n, cn) with its key. Every A records it publishes an anchor (n, cn, σ) to a witness, an append-only store that the logger cannot edit. Records within a batch can also be arranged in a Merkle tree, so that single entries can be proved without revealing the rest. Figure 1 shows the parts and Figure 2 the steps for each append.

3.2. Auditors and attackers

An auditor holds the logger's public key and can read the log and the witness. A full audit recomputes the chain, checks every checkpoint signature, and compares the chain value at each anchor with the witness. We consider three attackers.

Stolen Key: Holds the current signing key and write access. Can rewrite any record and re-sign everything, and can truncate the log.

Write Access : only Can edit stored records and recompute hashes, but cannot sign.

Forking Operator: Shows different versions of the log to different auditors. This is the threat when no trusted witness exists. We assume SHA-256 is collision resistant, Ed25519 signatures are unforgeable without the key, and the witness is append-only and honest. These are the usual assumptions, and the third is the one an operator has to arrange in practice.

4. Experimental Setup

Let the log have n records, a checkpoint every B and an anchor every A. $c = B\left\lfloor \frac{n}{B} \right\rfloor$ be the number of records covered by the latest checkpoint and $a = A\left\lfloor \frac{n}{A} \right\rfloor$ the number covered by the latest anchor. Index records from 0.

Proposition 1 (Undetectable windows). Assume a complete audit as above. (i) An attacker with write access but no key who changes record j is detected iff j < max(a, c).
(ii) An attacker with the key who rewrites record j and re-signs is detected iff j < a. (iii) An attacker with the key who removes the last m records is detected iff n − m < a. With no witness (a = 0), (ii) and (iii) are never detected.

Proof: Changing record j changes the chain value cn′ for every n′ > j. An artifact that commits to the log up to n′ therefore exposes the change if n′ > j. The artifacts are checkpoints, which are signed and need the key to forge, and anchors, which sit at the witness and cannot be altered.

(a). Without the key, the attacker cannot produce a valid signature for any changed value, so every

Jack Sparrow Publishers © 2026, IJRDES , All Rights Reserved
www.jacksparrowpublishers.com

ISSN: 2582 - 4201(Online) , http://www.ijrdes.com/ , IJRDES, Vol. 8, Issue 6 , 2026 , https://doi.org/10.63328/IJRDES-V8RI6P1

checkpoint with $n' > j$ fails verification. These exist iff $j < c$. Anchors with $n' > j$ also expose the change, and exist iff $j < a$. So detection holds iff $j < \max(a, c)$.

(b). With the key, checkpoints can be re-signed, so only anchors remain, and they exist for $n' > j$ iff $j < a$.

(c). Truncation to $n - m$ records is exposed iff some anchor refers to a count larger than $n - m$, that is, iff $n - m < a$. Without a witness there are no anchors.

Corollary 1 : (Window size). If the attack time is uniform relative to the anchoring schedule, the number of records an attacker with the key can rewrite undetected has mean $(A-1)/2$ $\mathbb{E}[\text{unanchored records}] = \frac{A-1}{2}$, and maximum $A - 1$. For an attack on a uniformly random record in a log of n records, the probability of going undetected is $P(\text{undetected}) = \frac{n-a}{n} \approx \frac{A}{2n}$. This is a worst case in the sense that the attacker chooses where to cut. A smart attacker will wait until just after an anchor to maximise the window. The key point is that the exposed window depends only on the anchor interval A, and not on the signing interval B, when the attacker has the key.

Proposition 2 (Cost–window invariant). Let records arrive at rate r per second $\text{anchor calls per second} = \frac{r}{A}$ and let each anchor take time c_a (signing plus the call to the witness). The anchoring calls per second are r/A, the largest undetectable window is $W = \frac{A-1}{r} \approx \frac{A}{r}$ seconds, and the fraction of one thread's time spent anchoring is $f = \frac{c_a r}{A}$

Therefore

$$f \cdot W \approx c_a \qquad (1)$$

independent of r.

Proof: $f = (r/A)\, c_a$ and $W = A/r$ (up to the -1), so $f W = c_a$.

The invariant says that halving the window doubles the anchoring cost, whatever the traffic. With $c_a = 50$ ms, a one-second window costs 5% of a thread, and a 100-ms window costs half of one.

We implemented the log in Python 3 using the cryptography library for Ed25519 and hashlib for SHA-256. Records are about 150 bytes of JSON. Merkle trees follow the Certificate Transparency construction [5]. Inclusion and consistency proofs were checked for every tree size up to 257, for every prefix length, including rejection of forged proofs. Timings come from one core of the sandbox machine with no tuning. Run-to-run variation on this machine is a few microseconds per record, which we report. The witness is an in-process object, so its network latency is a modelled parameter in Section 8. For attacks we log 1,000 to 2,000 records with $B = 50$ and a witness anchoring every $A \in \{10, 50, 100, 250, 500, 1000\}$, plus a case with no witness. For each setting we run 300 attacks of each type on freshly generated logs, with the attack time and record position drawn at random. The attacker rebuilds the log with the real code (new chain, new signatures if it has the key) and a full audit decides whether it is detected.

Proposition 3 (Proof sizes). In a Merkle tree of n and the expected number of epochs until detection is 1/(1–

$q0) \approx (g-1)/\, 2pk(g-k)$ for small p.

Proof: An auditor holding X compares across views if it gossips (probability p) and picks one of the g–k holders of Y (probability $(g - k)/(g - 1)$). Choices are independent across auditors, so the probability that none of the k X-holders and none of the $g - k$ Y-holders makes a cross-view comparison is the product in Eq. (2). Detection time is geometric with success probability $1 - q0$. For small p, $q0 \approx \exp - 2pk(g-k)/(g-1)$.

$$q_0 = \left(1 - \frac{p(g-k)}{g-1}\right)^k \left(1 - \frac{pk}{g-1}\right)^{g-k} \qquad (2)$$

Proposition 5 (Risk-triggered anchoring). Records are of a high-risk class with probability q and risk rh, and otherwise of risk rℓ. Fix an average anchoring rate ϱ per record. Uniform anchoring places an anchor every 1/ϱ records. Risk-triggered anchoring anchors right after every high-risk record $\mathbb{E}[\text{random-anchor suffix length}] = \frac{1-\rho}{\rho} = \frac{1}{\rho} - 1$ plus a background rate ϱ − q on other records. $\bar{r} = (1-q)r_\ell + q r_h$ Let the exposure be the expected total risk of the unanchored suffix at a random attack time $E_t < E_u \Leftrightarrow 2r_\ell < \bar{r}$. Then leaves, an inclusion proof has at most ⌈log2 n⌉ hashes

$$E_u = \frac{1/\rho - 1}{2}\bar{r}, \qquad E_t = \left(\frac{1}{\rho} - 1\right) r_\ell \qquad (3)$$

and a consistency proof between sizes $m < n$ has at most $\text{inclusion proof hashes} \le \lceil \log_2 n \rceil$ hashes [5]. A full audit of a chain is Θ(n). We do not prove this; it is standard. Section 6 measures it. where $\bar{r} = (1 - q)r_\ell + q\, r_h$, so risk-triggered anchoring has the lower exposure iff $\frac{r_h}{r_\ell} > \frac{1+q}{q}$ (4)

Jack Sparrow Publishers © 2026, IJRDES , All Rights Reserved
www.jacksparrowpublishers.com

ISSN: 2582 - 4201(Online) , http://www.ijrdes.com/ , IJRDES, Vol. 8, Issue 6 , 2026 , https://doi.org/10.63328/IJRDES-V8RI6P1

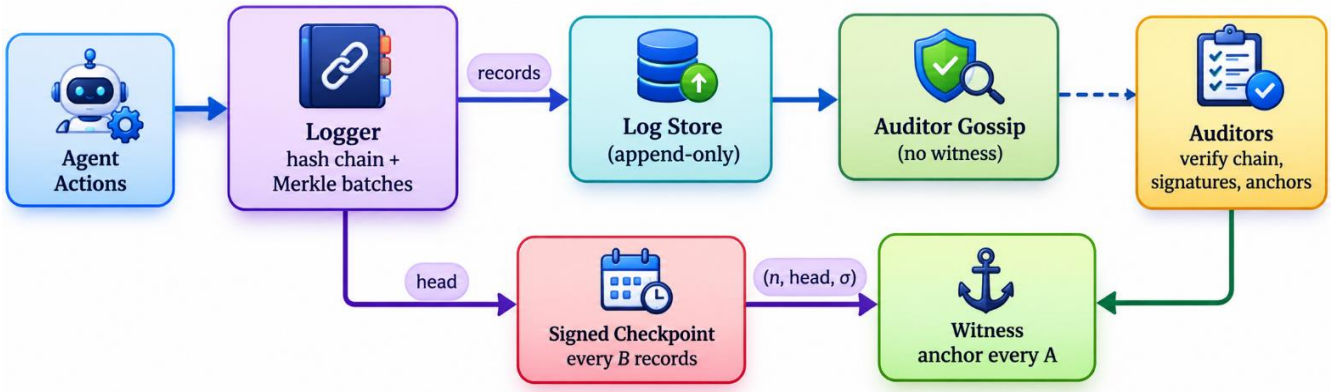


Figure. 1 System overview. The logger chains records and signs a checkpoint every B records. A witness holds anchors published every A records. Auditors verify the chain, signatures and anchors, and, if there is no witness, compare heads with each other.

5. Cost of Logging and Auding

5.1. Appending

Table 1 and Figure 4 give the cost per record. Signing is the expensive primitive: one Ed25519 signature took 37 µs, against 1.1 µs for hashing a 150-byte record. Signing every record cost 54 µs, about 6.5 times the chain-only cost of 8.4 µs, and limits one thread to about 18,481 records per second.

Signing a checkpoint every 10 records brought the cost down to about 12 µs per record, and by 100 records the extra cost was below the noise of our measurements, with throughput of about 80,962 records per second. We cannot rank the three cheapest settings.

$$\mathbb{E}[\text{epochs}] = \frac{1}{1-q_0} \approx \frac{g-1}{2pk(g-k)} \quad \text{for small } p$$

Their medians differ by less than the spread across repeated runs of the same setting (up to about 7 µs).

$$q_0 \approx \exp\left(-\frac{2pk(g-k)}{g-1}\right) \quad \text{for small } p$$

The prediction "chain cost plus signature cost divided by B" matches the measured values well for B = 1 and B = 10 and is, within noise, in line with the rest. The practical conclusion is that batching signatures removes the signing cost, and what remains is hashing and serialisation.

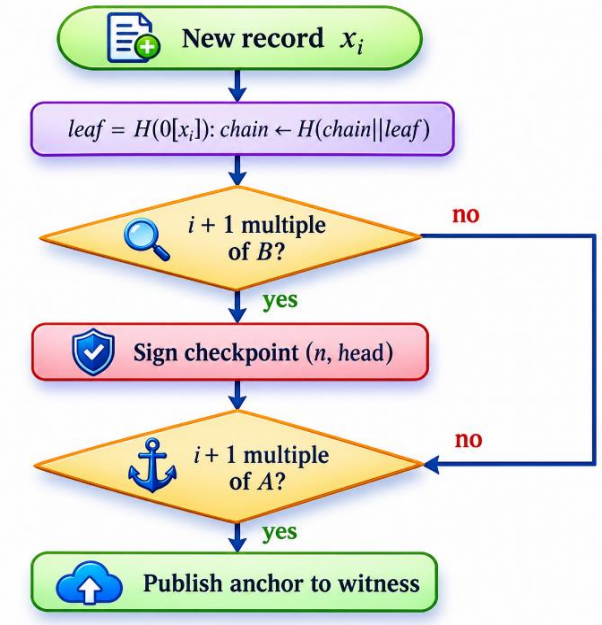


Figure 2: Steps for each appended record.

Table. 1 Append cost per record (median of five runs of 10,000 records, in random order). "Spread" is the minimum and maximum across the runs. "Predicted" is the chain-only cost plus signature cost divided by B.

Strategy	Median (μs)	Spread	Predicted	Records/s
chain only	8.4	7.1–9.0	8.4	119,296
sign each record	54.1	48.5–72.0	44.9	18,481
sign every 10	12.1	11.1–18.6	12.0	82,771
sign every 100	12.4	7.6–14.6	8.7	80,962
sign every 1000	11.6	7.3–14.1	8.4	86,423

Jack Sparrow Publishers © 2026, IJRDES , All Rights Reserved
www.jacksparrowpublishers.com

ISSN: 2582 - 4201(Online) , http://www.ijrdes.com/ , IJRDES, Vol. 8, Issue 6 , 2026 , https://doi.org/10.63328/IJRDES-V8RI6P1

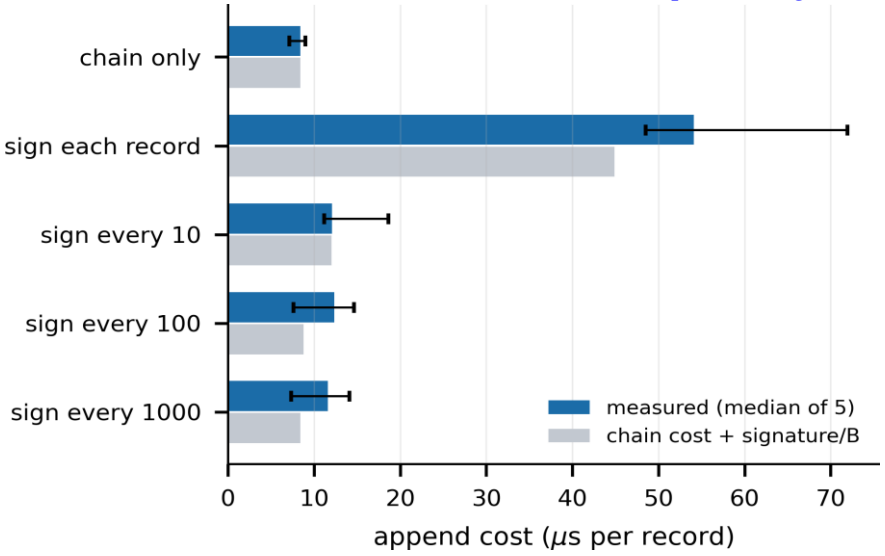


Figure. 4 Append cost per record for different signing strategies. Error bars show the spread across five runs. Only signing every record is clearly more expensive.

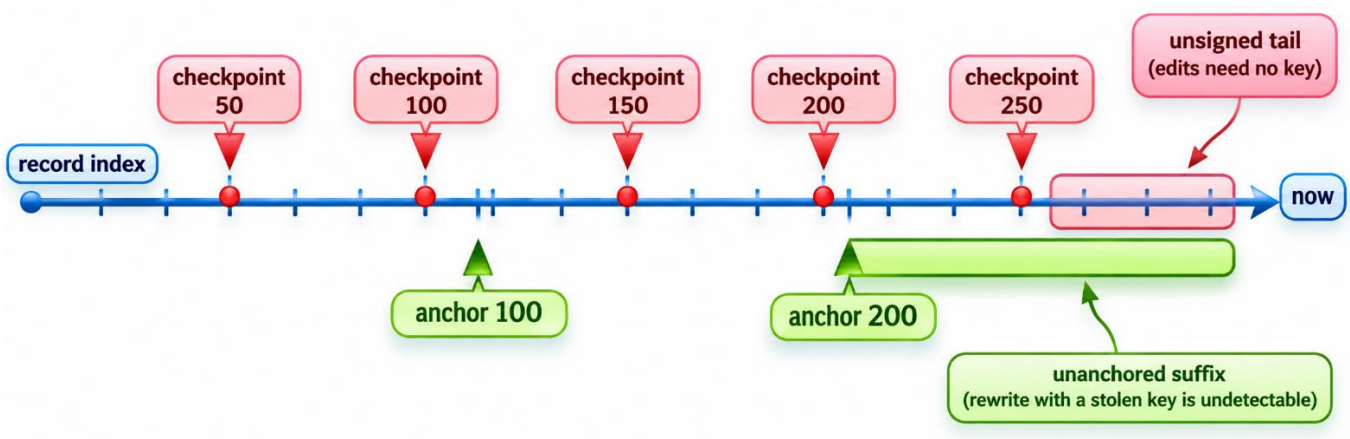


Figure. 3 The two windows. With B = 50 and A = 100 and a log of 273 records, the last 23 records are covered by no signature (red), so anyone with write access can edit them. The last 73 records are covered by no anchor (green), so an attacker with the key can rewrite them without leaving a trace.

Table. 2 Full audit time in milliseconds (checkpoint every 100 records, and signature on every record).

Records	Every 100	Every record
1,000	3.1	121
3,000	9.3	365
10,000	30.9	1348
30,000	97.7	3954
100,000	312.9	–

Table. 3 Merkle tree costs. Build time in seconds; inclusion proof length in hashes and verification in μs; consistency proof for $m = n/2$.

n	Build (s)	Incl. len	Incl. (μs)	Cons. len	Cons. (μs)
100	0.00	6.9	8.3	7	10
1,000	0.00	10.0	11.3	9	16
10,000	0.01	13.7	15.7	12	19
100,000	0.11	16.9	18.6	14	24
1,000,000	1.22	20.0	24.3	–	–

Jack Sparrow Publishers © 2026, IJRDES , All Rights Reserved
www.jacksparrowpublishers.com

ISSN: 2582 - 4201(Online) , http://www.ijrdes.com/ , IJRDES, Vol. 8, Issue 6 , 2026 , https://doi.org/10.63328/IJRDES-V8RI6P1

Merkle proofs avoid the linear cost. A single inclusion proof for a tree of a million records has 20 hashes and verified in 24 µs, about 12,856 times faster than the full audit of one-tenth as many records. Building the tree of a million leaves took 1.2 s. A consistency proof between a tree of 50,000 and one of 100,000 records had 14 hashes and verified in 24 µs (Table 3, Figure 5b). An auditor who tracks the log can therefore check that it only grows, without ever downloading it. Our attack experiments use full audits, which is the conservative choice.

5.2. What the Attacks Catch

Table 4 and Figure 6 give detection rates for 300 real attacks of each type at each setting, next to the prediction from Proposition 1. They agree exactly in every cell. This needs one caveat. The prediction is evaluated on the same random draws as the attack, so agreement shows that the real implementation detects an attack if and only if the proposition says it should, and it is a correctness test of the proposition and of our code, not an independent statistical estimate.

Stolen key: Without a witness, rewriting a record with a stolen key is never detected (0% detection), and nei-ther is truncation, consistent with our earlier prototype. With a witness, detection depends on A. Rewriting a random record was detected in 100% of attacks at A = 10, 97% at A = 100 and 70% at A = 1000. Tail truncation is much harder to catch because the attacker removes the most recent records, which are exactly the unanchored ones: detection was 99% at A = 10, 84% at A = 100 and only 15% at A = 1000.

Write access without the key: Signed checkpoints alone catch nearly all such edits, 97% even with no wit-ness. The undetected edits are those in the last partial batch, after the final checkpoint, which anyone with write access can alter and which the logger will later sign as if it were genuine. With B = 50 that is the last 0 to 49. records. A shorter checkpoint interval or an anchor right after each batch closes it.

Table 4: Detection rate in 300 real attacks (theory in parentheses; B = 50). Rewrite and truncation need the key.

Anchor every A	Key rewrite	No-key edit	Truncation
10	100% (100%)	100% (100%)	99% (99%)
50	99% (99%)	99% (99%)	90% (90%)
100	97% (97%)	98% (98%)	84% (84%)
250	91% (91%)	99% (99%)	55% (55%)
500	80% (80%)	98% (98%)	30% (30%)
1000	70% (70%)	99% (99%)	15% (15%)
none	0% (0%)	97% (97%)	0% (0%)

Table 5: Anchoring at 100 records per second. Time is the share of one thread spent anchoring, for an assumed witness latency of 50 ms and of 10 ms.

A	Window (s)	Calls/hour	50 ms	10 ms
1	0.01	360,000	500.4%	100.37%
10	0.1	36,000	50.0%	10.04%
100	1	3,600	5.0%	1.00%
1,000	10	360	0.5%	0.10%
10,000	100	36	0.1%	0.01%

5.3. The Cost–Window Trade-off

Anchoring needs a call to the witness, whose latency is outside our measurements. We therefore treat it as a parameter and use the measured signing cost of 37 µs plus 10, 50 or 200 ms. Table 5 applies Proposition 2 at 100 records per second. Anchoring every record means 100 calls per second and a window of 10 ms, which needs more than a whole thread at 50 ms per call (500%), so it would have to run asynchronously on many connections. Anchoring every 100 records means one call per second, a window of one second and about 5% of a thread at 50 ms. At 10,000 records per anchor the window is 100 seconds and the cost is negligible. Figure 7 shows the same trade-off. The straight lines on the log–log plot are the invariant of Eq. (1).

5.4. No Witness: Detecting a Fork by Gos-sip

If there is no trusted witness, the remaining defence against a forking operator is for auditors to compare notes. We simulated 50 auditors with k of them shown one version and the rest another, each gossiping with

Jack Sparrow Publishers © 2026, IJRDES , All Rights Reserved
www.jacksparrowpublishers.com

ISSN: 2582 - 4201(Online) , http://www.ijrdes.com/ , IJRDES, Vol. 8, Issue 6 , 2026 , https://doi.org/10.63328/IJRDES-V8RI6P1

probability p per epoch with one random peer (Figure 8). The simulated mean epochs to detection match Eq. (2) within about 2% across all twelve settings. A lone victim (k = 1) is the hardest case. With p = 0.01 it takes about 50.4 epochs to catch the fork, with p = 0.1 about 5.5. If a tenth of the auditors see the fork (k = 5) it takes 11.3 and 1.6 epochs. The cost of detection is therefore gossip traffic: halving the expected delay requires doubling p. Gossip is cheaper than a witness in infrastructure but weaker in guarantee. It catches forks only after auditors have actually compared, and it assumes honest auditors

5.5. Where to Spend the Anchoring Budget

Anchors can be placed after the records that matter most. Proposition 5 says this pays only if the triggering class carries enough risk. We simulated streams of 200,000 records with background anchor rate 0.001 and compared uniform and risk-triggered anchoring at the same total number of anchors (Figure 9a). The formulas of Eq. (3) matched the simulation within 7.3%. The result is a threshold, not a free win. With 7% high-risk records and a risk ratio of 100, uniform anchoring exposes about 52 units of risk and risk-triggered anchoring about 13, a ratio of 0.26. With a ratio of 20 the two are close (15 against 13, ratio 0.87), and with a ratio of 10 targeting is worse (1.25 times as much exposure). The break-even ratio (1 + q)/q is 15.3 for q = 0.07 and 101 for q = 0.01 (Figure 9b). Rare, very dangerous events are worth anchoring immediately. Moderately risky ones are not, because anchoring after every one of them leaves long unanchored stretches of ordinary records between them.

5.6. Discussion and Limitations

The witness is a trust assumption. Everything above assumes the witness is append-only and the at-tacker cannot alter it. In practice this means a third party, a public transparency log, or several independent witnesses. If the witness is compromised, anchors lose their meaning. Multiple witnesses with a quorum would help, and we did not evaluate that.

Measurements are indicative: The timings are from one machine, in Python, with noise of a few microsec-onds per record. We did not test concurrency, durable storage, network witnesses or key hardware. Real signing in a hardware module or a remote key service could cost much more than 37 µs, which would favour larger batches. The witness latency is an assumed parameter.

The audit is complete in our model: Proposition 1 assumes the auditor checks every checkpoint and anchor, and that it sees the log it should see. An auditor who checks rarely gives the attacker more time. If audits happen every P records, a tampered record is found about P/2 records after it falls into the anchored region, which we did not simulate.

What a log cannot prove: The log shows that records were not changed after they were written. It does not show that the agent logged everything, or that the logged payload is true. An agent or logger that omits or falsifies records before they enter the log defeats it. That needs controls on the path into the log, such as the mediation gateway of the earlier papers in this series.

Privacy: Logging every action can expose personal data. Hash-chained logs resist deletion, which conflicts with erasure rights. One approach is to log hashes of payloads and keep the payloads in a separate, delectable store, at the price of losing the ability to prove content after deletion.

Gossip Assumptions: The gossip result assumes independent random pairing and honest auditors, and counts epochs and not time. The attacker may also try to learn who talks to whom.

6. Conclusion and Future Scope

Hash chains and signatures make a log tamper-evident only for the part of it that an outside party has pinned down. We found that signing in batches removes the signing cost, that anchoring converts directly between money and window length (the invariant $f \cdot W = ca$), and that what an attacker with the key can still do is exactly bounded by the last anchor, as shown by 6,300 real attacks. Truncation is the weak point: it is the attack that anchoring catches least often, because it removes precisely the newest records. Without a witness, gossip detects forks in a time we can state exactly. Spending the budget after the riskiest records pays only above a clear risk ratio. The next steps are to test real witnesses and multi-witness quorums, to measure the full path with a hardware-backed key, and to simulate auditors who check at intervals.

References

[1]. S. Haber, W. S. Stornetta. How to time-stamp a digital docu-ment. Journal of Cryptology, 3(2):99–111, 1991.

[2]. B. Schneier, J. Kelsey. Secure audit logs to support com-puter forensics. ACM Trans. Information and System Security, 2(2):159–176, 1999.

[3]. R. C. Merkle. A digital signature based on a conventional encryption function. In CRYPTO, 1987.

[4]. S. A. Crosby, D. S. Wallach. Efficient data structures for tamper-evident logging. In USENIX Security, 2009.

[5]. B. Laurie, A. Langley, E. Kasper. Certificate Transparency. RFC 6962, 2013.

[6]. D. J. Bernstein, N. Duif, T. Lange, P. Schwabe, B.-Y. Yang. High-speed high-security signatures. J. Cryptographic Engi-neering, 2(2):77–89, 2012.

A. Parameters

Record about 150 bytes of JSON. Hash SHA-256 with domain separation (0x00 for leaves, 0x01 for inner

Jack Sparrow Publishers © 2026, IJRDES , All Rights Reserved
www.jacksparrowpublishers.com

ISSN: 2582 - 4201(Online) , http://www.ijrdes.com/ , IJRDES, Vol. 8, Issue 6 , 2026 , https://doi.org/10.63328/IJRDES-V8RI6P1

nodes). Signatures Ed25519. Append timing: 10,000 records, five runs in random order, median reported. Audit timing: up to 100,000 records, checkpoint every 100 or every record. Merkle timing: trees up to 106 leaves; 300 random inclusion proofs and 50 consistency checks per size. Attacks: logs of 1,000–2,000 records, B = 50, 300 attacks of each type per anchor interval, random positions and times. Gossip: 50 auditors, one random peer per gossip, mean over 4,000 runs. Risk-triggered anchoring: 200,000 records, background rate 0.001. Code and figures are in the supplementary archive. All figures were generated from code at 300 dpi and printed at their stored width.

How To Cite:

Hema Dega , " Anchoring Trust in Agentic AI: A Cost and Security Evaluation of Tamper-Evident Audit Logs " , International Journal of Research and Development in Engineering Sciences (IJRDES), Volume 8, Issue 6 , pp 1 - 8 , November 2026.
CrossRef DOI: https://doi.org/10.63328/ijrdes-v8ri6p1

Declaration

Conflicts of Interest: The authors declare no conflict of interest.

Author Contribution: All authors wrote the main manuscript text and also consent to the submission.

Ethical approval: Not applicable.

Consent to Participate: All authors consent to participate.

Funding: Not applicable, and No funding was received

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Personal Statement: We declare with our best of knowledge that this research work is purely Original Work and No third party material used in this article drafting. If any such kind material found in further online publication, we are responsible only for any judicial and copyright issues.

Generative AI Statement: The authors declare that generative AI was not used in the preparation or creation of this manuscript. The alternative text (alt text) accompanying the figures in this article was generated by Frontiers with the assistance of artificial intelligence. Reasonable efforts have been made to ensure the accuracy and appropriateness of the generated alt text, including review by the authors wherever possible. If you identify any inaccuracies or issues, please contact the editorial team.

Publisher's Note: The statements, opinions, and claims presented in this article are exclusively those of the authors and do not necessarily reflect the views of their affiliated institutions, the publisher, editors, or reviewers. Any products discussed or evaluated, as well as any claims made by their manufacturers, are not warranted, approved, or endorsed by the publisher.

Acknowledgements

We thank everyone who inspired our work

Jack Sparrow Publishers © 2026, IJRDES , All Rights Reserved
www.jacksparrowpublishers.com