MEMMARK: State-Evolution Attribution Watermarking for Agent Long-Term Memory Systems

Haobo Zhang[1,*] **Xutao Mao**[2,*]
Guangyuan Dong[3] **Ziwei Li**[4] **Xuanbo Su**[5]
Kaijie Chen[6] **Jing Yang**[7] **Zheng Lin**[8,†]
[1]Zhejiang University of Technology, [2]Independent Researcher
[3]National University of Singapore, [4]King Abdullah University of Science and Technology
[5]Bairong, [6]Tongji University, [7]Universiti Malaya, [8]University of Hong Kong
zhanghaobo@zjut.edu.cn
[*]Equal contribution. [†]Corresponding author.

Abstract

Memory-backed agents need provenance that can survive leaked or migrated snapshots, where logs, visible outputs, and trusted metadata may be absent. We propose MEMMARK[1], a state-evolution attribution watermark that embeds an owner-controlled signal into latent memory-write decisions. At each internal LLM call, MEMMARK samples among admissible candidates using keyed, distribution-preserving selection, and records cryptographic commitments with signed session anchors and reveal evidence. This makes attribution depend on reproducible backend behavior rather than mutable provenance fields. Across A-MEM and GRAPHITI on LoCoMo, with three LLM backbones, MEMMARK preserves memory utility: Overall F1 retains 99.6% of the unwatermarked baseline, while BLEU-1 changes by +0.2%. It also provides usable carrier capacity, with 1.16, 1.14, and 1.26 bits of mean entropy for update-target, link-target, and semantic-realization decisions. In the snapshot-only R3 setting, MEMMARK recovers the full 40-bit payload from final snapshots, while wrong-key verification remains near chance. Under nine memory-lifecycle attacks, verification distinguishes tampering, evidence deletion, and partial payload recovery. These results show that robust snapshot-only attribution is feasible for long-term agent memory without surviving traces, trusted metadata, or utility-degrading.

1 Introduction

LLM agents are increasingly moving from single-session responders to persistent actors whose decisions depend on state that survives across interactions. In such systems, the memory layer is no longer a passive cache: it becomes part of the security boundary (Park et al., 2023; Packer et al., 2024; Zhong et al., 2024; Chen et al., 2024; Wei et al., 2025; Lin et al., 2026). Recent systems such as A-MEM, GRAPHITI, Mem0, MemOS, Memory-R1, and MemMachine (Xu et al., 2026a; Rasmussen et al., 2025; Chhikara et al., 2025; Li et al., 2025; Yan et al., 2025; Wang et al., 2026b) maintain long-lived state through extraction, updates, consolidation, linking, and deletion. Benchmarks likewise evaluate memory through long-horizon recall, knowledge updates, temporal reasoning, and incremental multi-turn state maintenance (Maharana et al., 2024; Wu et al., 2025; Hu et al., 2025). As memory writing becomes an explicit object of system design (Zhang et al., 2025, 2026a; Sun et al., 2025), a natural response is to attach provenance fields, such as source anchors, versions, or lifecycle traces (Zhu et al., 2026; Li et al., 2025). These fields are useful when the writer and storage layer are trusted, but they are much less useful when the memory snapshot itself may have been rewritten.

We study a post-compromise forensic setting in which the verifier may not have access to a trusted write-time trace. Prompts, tool calls, memory-write requests, and backend logs may be absent, incomplete, or controlled by the same actor who controls the memory store; the only durable artifact may be the final memory snapshot. This setting is not a replacement for trusted logging in normal operation, but the fallback case that remains when such logs are lost, withheld, or suspected to be corrupted. An attacker can rewrite ownership fields, erase identifiers, fabricate provenance chains, or edit backend-native histories such as A-MEM evolution logs and GRAPHITI fact-invalidation traces. This threat model is motivated by evidence that agent memory can be poisoned, stealthily modified, or persistently compromised (Chen et al., 2024; Wei et al., 2025; Srivastava and He, 2025; Lin et al., 2026), and by recent surveys that identify agent memory attacks as a distinct safety risk for large-model-powered agents (Ma et al., 2026). Thus, field-based provenance has a circular failure mode: the same untrusted snapshot contains both the contested memory and the mutable fields that certify

[1]https://github.com/zhb0119/MemMark

Findings of the Association for Computational Linguistics: EMNLP 2026, pages 8333–8349
October 24-29, 2026 ©2026 Association for Computational Linguistics

it.

Watermarking offers a natural way to make attribution survive untrusted handling. Token-level LLM watermarks (Kirchenbauer et al., 2023; Dathathri et al., 2024; Pan et al., 2024; Mao et al., 2025), RAG or structured-data watermarks (Lv et al., 2025; Jovanović et al., 2025; Liu et al., 2025; Chen et al., 2026; Hristov et al., 2023; Peng et al., 2025), and agent-level behavioral watermarks (Huang et al., 2025, 2026a; Meng et al., 2026; Wang et al., 2026a) provide important attribution mechanisms. However, their signals are placed in generated text, protected corpora or graphs, visible tool use, or action trajectories. Long-term memory forensics exposes a different evidence channel: the original prompts, tool calls, and execution trace may be unavailable, while the surviving artifact is only the final memory snapshot. The question is therefore whether memory evolution itself can leave reproducible evidence of who wrote it.

We propose MEMMARK, a watermark for long-term memory evolution under adversarial snapshot control. Instead of storing attribution in mutable metadata, MEMMARK embeds it into latent but utility-preserving state-transition choices: which existing item to update, which related item to link, or which semantically equivalent realization to store. A keyed sampler selects among admissible candidates while preserving the backend's preference distribution, so attribution is carried by the backend's own evolution behavior rather than by a self-reported field. To make this evidence usable after the external execution trace is lost, MEMMARK records an in-snapshot audit path whose validity is checked by replay against the keyed sampler and the admissible transition set. MEMMARK supports three verification regimes: R1 with a complete external log, R2 with a partial log, and R3 with only the final memory snapshot. In this way, MEMMARK moves provenance from editable claims to a reproducible behavioral trace.

Contributions.

1. **Snapshot-only attribution with full payload recovery.** We introduce an audit design and R1/R2/R3 verification hierarchy that supports attribution even when only the final memory snapshot is available. In R3, MEMMARK recovers the full 40-bit payload, compared with no recovery for signed-metadata-only and 15% wrong-key recovery.
2. **Utility-preserving memory-evolution watermarking.** We introduce MEMMARK, a backend-invariant watermarking abstraction that embeds attribution into latent memory-evolution decisions rather than editable provenance fields. Across six model–backend settings, MEMMARK retains 99.6% of the unwatermarked Overall F1 and improved by 0.2% of BLEU.
3. **Cross-backend capacity and attack diagnostics.** Across A-MEM and GRAPHITI, MEMMARK exposes mean carrier entropies of 1.16, 1.14, and 1.26 bits for update-target, link-target, and semantic-realization decisions, and remains diagnostic under nine memory-lifecycle attacks at strengths 0.1, 0.3, and 0.5.

2 Related Work

2.1 Watermarking for Text, Data, and Agent Behavior

Existing watermarks protect evidence channels that may be absent in memory forensics. Text watermarks embed signals in generated tokens (Kirchenbauer et al., 2023; Dathathri et al., 2024; Pan et al., 2024; Mao et al., 2025; Gao et al., 2026b; Xu et al., 2026b); data-level methods protect retrieval corpora, multimodal stores, or graphs (Lv et al., 2025; Jovanović et al., 2025; Liu et al., 2025; Chen et al., 2026; Hristov et al., 2023; Peng et al., 2025); and agent watermarks target visible planning, tool use, or trajectory data (Huang et al., 2025, 2026a; Meng et al., 2026; Wang et al., 2026a). MEMMARK keeps the behavioral intuition but moves the carrier to memory-evolution decisions that remain recoverable from the backend snapshot itself.

2.2 Long-Term Agent Memory and Adversarial Provenance

Persistent-memory agents and backends such as A-MEM, GRAPHITI, MemOS, Memory-R1, and MemMachine maintain note networks, temporal graphs, or structured memory objects (Park et al., 2023; Packer et al., 2024; Zhong et al., 2024; Xu et al., 2026a; Rasmussen et al., 2025; Li et al., 2025; Yan et al., 2025; Wang et al., 2026b); benchmarks evaluate their long-horizon recall, updates, temporal reasoning, and incremental multi-turn memory (Maharana et al., 2024; Wu et al., 2025; Hu et al., 2025). Work on admission, update, and curation treats memory writing as an explicit decision layer (Zhang et al., 2025, 2026a; Sun et al., 2025), which is exactly the layer MEMMARK watermarks.

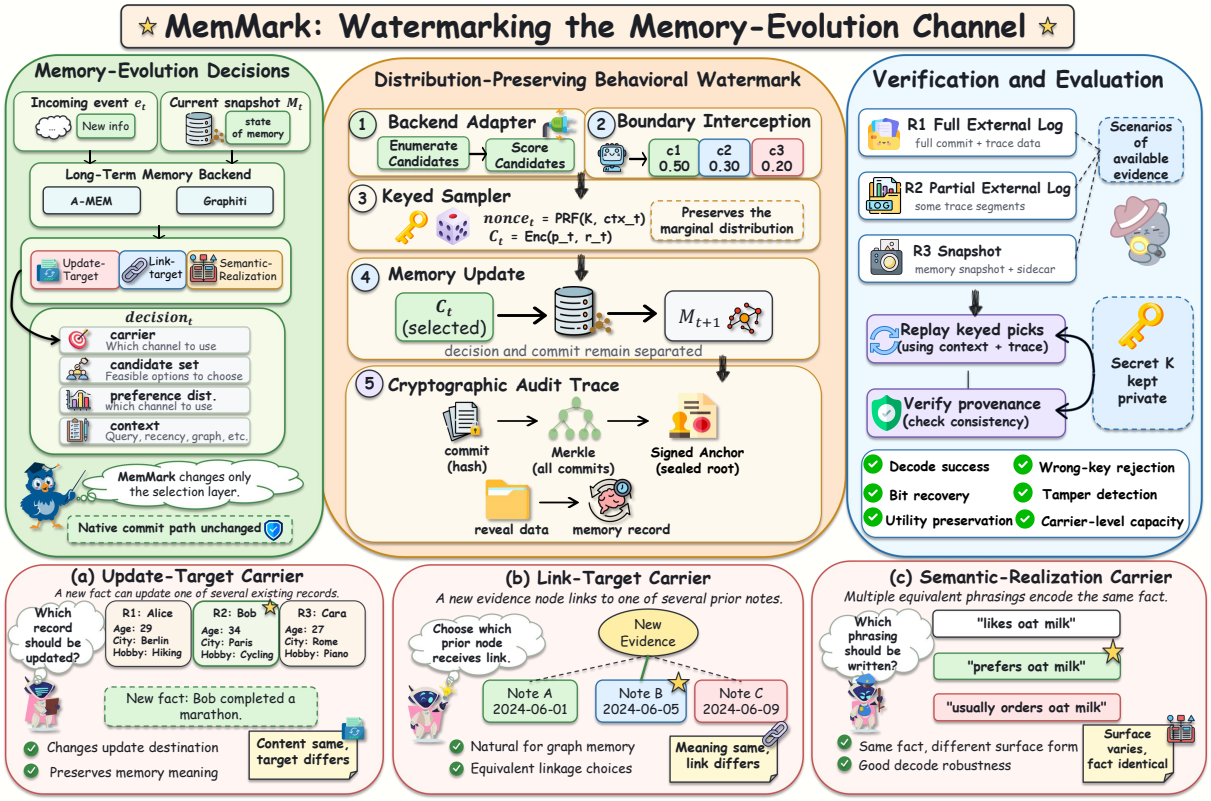

Figure 1: **End-to-end MEMMARK pipeline.** A memory write exposes carrier-specific choices over update targets, link targets, and semantic realizations. MEMMARK enumerates and scores feasible candidates, intercepts the LLM-call boundary, and uses a secret-keyed, distribution-preserving sampler to select c^* while leaving the native commit path unchanged. Each selected decision is bound to a commitment, per-session Merkle log, and signed anchor; reveal data is stored with the memory record. The same evidence supports R1 full-log, R2 partial-log, and R3 snapshot-only attribution by replaying keyed picks and checking provenance consistency.

Provenance and security work motivate the adversarial setting. TIERMEM and MemOS attach source anchors, versioning, or lifecycle metadata (Zhu et al., 2026; Li et al., 2025), while poisoning and memory-security studies show that memory can be maliciously or persistently compromised (Chen et al., 2024; Wei et al., 2025; Srivastava and He, 2025; Lin et al., 2026; Gao et al., 2026a; Yu et al., 2025; Zhang et al., 2026b; Huang et al., 2026b). Metadata helps under trusted storage, but not when the attacker controls the snapshot containing both memory and claimed provenance. MEMMARK instead combines replayable keyed choices with commitments, Merkle trees, and transparency logs (Pedersen, 1991; Merkle, 1987; Laurie et al., 2013).

3 Problem Formulation

3.1 Preliminaries and Notation

We consider a long-term memory backend over T turns. At turn $t \in \{1, \ldots, T\}$, the backend maps an incoming event e_t and current snapshot M_t to M_{t+1}. This transition is not monolithic: it contains latent state-evolution choices that are semantically admissible but not uniquely determined.

We formalize each such choice as a memory-evolution decision

$$\text{decision}_t = \langle \text{carrier}_t,\ \text{candset}_t,\ \text{probdist}_t,\ \text{ctx}_t \rangle \tag{1}$$

where carrier_t is the carrier type, $\mathcal{C}_t = \{c_t^1, \ldots, c_t^{k_t}\}$ is the candidate set, $\pi_t \in \Delta(\mathcal{C}_t)$ is the backend preference distribution, and ctx_t is reconstructible context.

We assume the backend induces a latent preference distribution $\pi_t^\star$ over $\mathcal{C}_t$, and that the baseline backend would commit $c_t \sim \pi_t^\star$. We elicit an explicit estimate π_t from self-reported model weights and define distribution preservation with respect to this estimate:

$$\hat{c}_t \sim \pi_t \quad \text{for all } t \in \{1, \ldots, T\}. \tag{2}$$

Decision–commit separation. MEMMARK observes and modifies only $(\mathcal{C}_t, \pi_t, \mathrm{ctx}_t)$; candidate generation and the commit step $\mathrm{APPLY_SELECTED}(M_t, \hat{c}_t) \rightarrow M_{t+1}$ remain on the native backend path. Thus, the watermark changes only the selected candidate, not the backend write operation itself.

3.2 Memory-Evolution Channel and Distribution-Preserving Coding

We model each decision stage as a time-varying discrete channel. To embed a provenance payload, MEMMARK replaces native sampling with a distribution-preserving encoder:

$$\hat{c}_t \leftarrow \mathsf{Enc}(\pi_t, r_t) \in \mathcal{C}_t, \tag{3}$$

where r_t is per-decision randomness reproducible by the embedder and verifier.

Keyed pseudorandomness. The watermark secret K and context derive a nonce

$$nonce_t \leftarrow \mathrm{PRF}(\mathsf{K},\ \mathrm{ctx}_t), \tag{4}$$

which seeds the sampling stream consumed by Enc. Since ctx_t is reconstructible from surviving evidence, the verifier can replay the keyed pick.

3.3 Threat Model

The watermark is embedded at write time but verified later, possibly after backend maintenance and partial or full loss of the external audit log.

Structural attacks on the backend. Memory stores may be poisoned or compacted (Chen et al., 2024; Wei et al., 2025; Srivastava and He, 2025). Such operations can remove, merge, or rewrite records, but cannot forge a commitment that opens against the anchored Merkle root without K. We therefore model the dominant effect as a surviving decision set $\mathcal{I} \subseteq \{1, \ldots, T\}$.

Verification regimes. We consider three deployment regimes:

- **R1 (full external log):** the verifier holds all per-decision commitments and the complete Merkle tree.
- **R2 (partial external log):** only a subset $\mathcal{I}$ of commitments survives, due to truncation, retention limits, or partial loss.
- **R3 (snapshot only):** the external log is unavailable; verification relies on the memory snapshot plus an in-record sidecar carrying reveal data and the anchored header.

Carrier carrier	A-MEM	GRAPHITI
`update_target`	note id	fact-edge id
`link_target`	keyword cluster	entity attach point
`semantic_realize.`	note description	edge label

Table 1: **Carriers Taxonomy**. Each carrier is a non-trivial-candidate-set evolve decision realized in backend-specific form; the watermark sampler reads only $(\mathcal{C}, \pi, \mathrm{ctx})$ and is therefore backend-invariant.

3.4 Objectives

We formalize two objectives under the hard constraint $\hat{c}_t \sim \pi_t$.

Utility preservation. Watermarking should preserve downstream memory quality. For baseline and watermarked snapshots M_T and $\hat{M}_T$, we require

$$\left| \mathbb{E}[\mathcal{U}(\hat{M}_T)] - \mathbb{E}[\mathcal{U}(M_T)] \right| \leq \varepsilon_{\mathcal{U}}, \tag{5}$$

where $\mathcal{U}(\cdot)$ is any downstream utility metric. This motivates carriers whose candidates are semantically equivalent.

Robust attribution. The verifier should recover provenance whenever enough decisions survive. In R1 and R2, given $\mathcal{I}$, we require

$$\Pr\big[\mathsf{Verify}\big(\{\hat{c}_t, \pi_t, \mathrm{ctx}_t\}_{t \in \mathcal{I}};\ K\big) = 1\big] \geq 1 - \delta(|\mathcal{I}|), \tag{6}$$

with failure probability $\delta(|\mathcal{I}|)$ decreasing as $|\mathcal{I}|$ grows. R3 uses the in-record sidecar instead of external commitments.

4 MemMark

4.1 Overview

Figure 1 gives the roadmap. MEMMARK exposes carrier-specific choices in a backend, turns them into a discrete candidate distribution at the LLM-call boundary, selects c^* with a secret-keyed sampler, and stores the evidence needed for R1–R3 verification. We follow the same structure below: carriers and adapter hooks (Section 4.2), distribution-preserving sampling (Section 4.3), and the cryptographic audit trace (Section 4.4).

4.2 Carrier Taxonomy and Backend Adapter

Table 1 lists our three carriers. The update-target carrier changes which existing object is modified; the link-target carrier changes which prior object

is connected to new evidence; and the semantic-realization carrier changes the surface form encoding the same fact. Each captures backend decision freedom while satisfying the semantic-equivalence requirement of Section 3.4.

Connecting a backend to MEMMARK requires three adapter hooks: ENUMERATE_CANDIDATES$(M_t, e_t, \text{carrier}) \rightarrow \mathcal{C}$, SCORE_CANDIDATES$(\mathcal{C}, \text{ctx}) \rightarrow \pi$, and APPLY_SELECTED$(M_t, c^*) \rightarrow M_{t+1}$. They expose the choice space, score candidates, and commit the selected candidate. The same abstraction also yields the carrier-level entropy and payload-allocation statistics used in RQ2. Since the adapter is the only backend-specific code path, one watermarking layer can operate over structurally distinct systems such as A-MEM and GRAPHITI.

4.3 Distribution-Preserving Behavioral Watermark

Decision interception and elicitation. Each internal LLM call provides an intervention boundary: the backend would often accept several nearby alternatives without changing the semantic role of the write. MEMMARK extends the prompt to request K plausible answers with self-reported preference weights. Parsing this wrapper response yields candidates $\{d_t^1, \ldots, d_t^K\}$ and a normalized distribution π_t, reducing open-vocabulary generation to a discrete choice that can be keyed, audited, and replayed. The backend receives only the final selected response.

Keyed distribution-preserving sampling. MEMMARK feeds $(\mathcal{C}_t, \pi_t, \text{ctx}_t)$ to a distribution-preserving binning sampler keyed by the watermark secret K and the context-bound nonce $nonce_t$ from Section 3.2. The sampler returns $\hat{c}_t$ and the number of embedded bits while preserving the marginal:

$$\Pr[\hat{c}_t = c \mid \pi_t, \mathsf{K}] = \pi_t(c), \quad \forall c \in \mathcal{C}_t. \tag{7}$$

Properties. The sampler provides the following guarantees; proofs are deferred to Appendix B.

Lemma 1 (Strict distribution preservation)**.** *For any valid π_t, the keyed pick $\hat{c}_t$ satisfies* $\Pr[\hat{c}_t = c^i \mid \mathsf{K}, \text{ctx}_t]_{\text{marg}} = \pi_t(c^i)$.

Lemma 2 (Cascade composition)**.** *Keyed picks across multiple calls of a single event are independent under distinct context-derived nonces, so expected embedding capacity grows approximately additively in the number of valid decisions.*

Lemma 3 (Backend invariance)**.** *Once $(\mathcal{C}_t, \pi_t, \text{ctx}_t)$ is fixed, the keyed pick depends only on the candidate distribution and secret key, not on the backend that produced the candidates.*

4.4 Cryptographic Audit Trace

To support verification after execution, MEMMARK records a cryptographic trace for the regimes in Section 3.3.

Per-decision commitments. Each sampled decision produces a commitment

$$\text{cm}_t = H\big(\,\text{ctx}_t \,\|\, H(\mathcal{C}_t) \,\|\, H(\pi_t) \,\|\, \hat{c}_t \,\|\, \text{bits}_t \,\|\, nonce_t\,\big),$$

with H collision-resistant and inputs canonically serialized.

Per-session Merkle tree and signed anchor. Per-decision commitments form a Merkle tree whose root is sealed in

$$\text{hdr} = (\,\text{agent_id},\ \text{user_id},\ \text{session_id},\ T,\ \text{root}_T,\ \text{sig}_{\mathsf{K}}(\text{root}_T)),$$

making the trace tamper-evident.

Per-leaf inclusion proofs. Each record carries its Merkle inclusion path at seal time, so a surviving leaf can be checked directly against the anchored root without rebuilding the log. Thus, structural attacks do not collapse verification into an all-or-nothing rebuilt-root test.

5 Experiments

We evaluate utility preservation (RQ1), capacity (RQ2), snapshot-only verification (RQ3), robustness to memory-lifecycle attacks (RQ4), and memory integrity (RQ5).

5.1 Setup

Backends & Models. We test two long-term memory backends: A-MEM (Xu et al., 2026a), a note-centric system with dynamic linking and memory evolution, and GRAPHITI (Rasmussen et al., 2025), a temporal graph memory with evolving entities and relations. We use Deepseek-V4-pro (DeepSeek-AI, 2026), Qwen3.6-flash (Qwen Team, 2026), and GLM-5 (GLM-5-Team et al., 2026). Candidate enumeration uses K=4 and T_{enum}=0.7 by default (Appendix E), candidate scoring uses T_{score}=0.0, and JSON mode is enabled throughout.

Benchmark. We use LoCoMo (Maharana et al., 2024), which contains ten multi-session dialogues with QA annotations spanning single-hop, multi-hop, temporal, commonsense, and adversarial.

Baselines. We compare MEMMARK with four controls: `no-watermark`, `random-replace`, `signed-metadata-only`, and KGMARK (Peng et al., 2025) on GRAPHITI, isolating native utility, unkeyed randomization, signed metadata without embedding, and the closest structured-memory watermark baseline.

Metrics. Appendix C defines all metrics. RQ1 reports LoCoMo F1, BLEU-1, and F1 deltas; RQ2 reports per-carrier entropy, payload share, and bits per decision; RQ3 reports R1/R2/R3 bit recovery and wrong-key bit recovery; RQ4 reports post-attack recovery, the verifier mode triggered by each attack, and wrong-key separation above the baseline under nine attacks[2]; and RQ5 reports carrier distribution, evidence-grounded retrieval recall, and write failures.

5.2 RQ1 — Utility Preservation

Table 2 compares MEMMARK with unwatermarked and ablated controls on LoCoMo. Across the six model–backend settings, MEMMARK preserves downstream utility: the average Overall F1 changes from 0.2816 for `no-watermark` to 0.2804 for MEMMARK, retaining 99.6% of the unwatermarked score, while average BLEU-1 changes from 0.3069 to 0.3077, retaining 100.2%. The absolute mean F1 drop is only 0.0012, and BLEU-1 increases slightly by 0.0008.

The effect differs by backend but remains small. On A-MEM, MEMMARK improves average Overall F1 from 0.3141 to 0.3255 and BLEU-1 from 0.3420 to 0.3529. On GRAPHITI, average Overall F1 decreases from 0.2490 to 0.2353, and BLEU-1 decreases from 0.2718 to 0.2624. Thus, the watermark does not introduce a systematic utility collapse: the attribution signal is embedded in ordinary memory-evolution decisions while keeping task performance close to the native systems.

The comparison with `signed-metadata-only` is especially useful: signing explicit provenance fields leaves utility largely unchanged, but does not create a recoverable state-evolution signal. MEMMARK remains in a comparable utility regime while moving the attribution evidence into the backend's own choices.

[2]Attack definitions are in Appendix G.

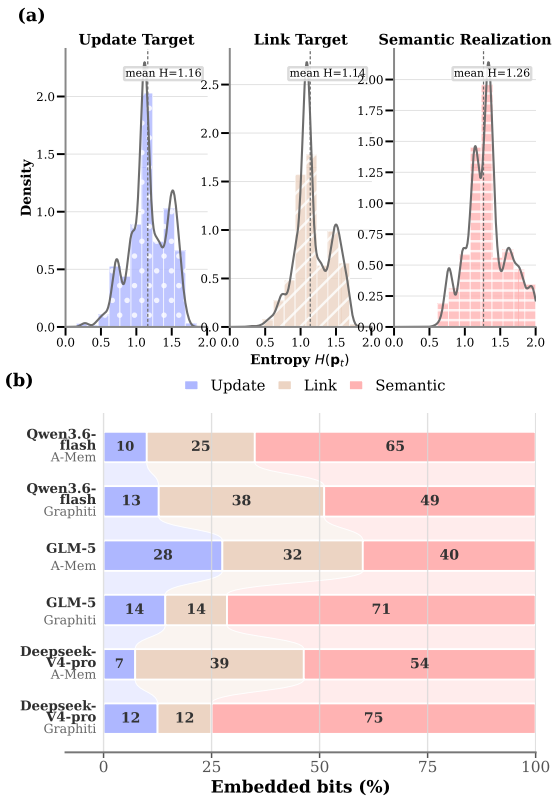

Figure 2: **RQ2 — Entropy and payload allocation across memory-evolution carriers.** (a) Per-carrier entropy over the six LLM–backend configurations; labels show means. (b) Per-carrier share of embedded payload bits, normalized within each configuration.

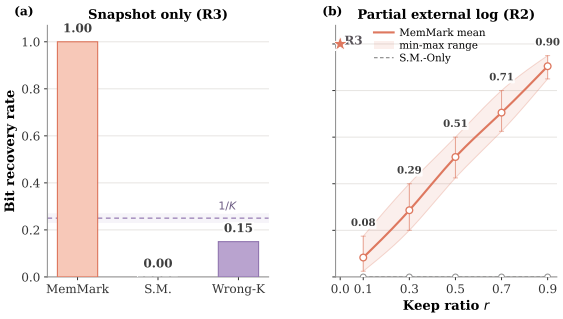

Figure 3: **R3 snapshot verification and R2 partial-log degradation.** Left: MEMMARK recovers all bits from the snapshot, while `signed-metadata-only` fails and the wrong-key control remains at chance level. Right: mean R2 recovery over six LLM–backend settings; shading and error bars show the minimum–maximum range.

5.3 RQ2 — Capacity

Figure 2 shows non-trivial capacity across all three carriers. The mean per-decision entropies are 1.16,

Model	Backend	Method	Single Hop (1)		Temporal (2)		Multi Hop (3)		Open Domain (4)		Adversarial (5)		**Overall**	
			F1	BLEU	F1	BLEU	F1	BLEU	F1	BLEU	F1	BLEU	F1	BLEU
Qwen3.6-flash	A-MEM	No-WM	0.1974	0.2373	0.3732	0.5360	0.1363	0.2333	0.4107	0.4265	0.1289	0.1234	0.2850	0.3322
		S.M.-Only	0.2127	0.2682	**0.4727**	**0.5676**	0.1884	0.3295	0.4115	0.4278	**0.2251**	**0.2072**	**0.3323**	**0.3696**
		Ran.	0.2305	**0.3364**	0.4309	0.5541	**0.2595**	**0.3761**	0.3479	0.3810	0.1980	0.1859	0.3033	0.3596
		MemMark	**0.2559**	0.3197	0.4143	0.5248	0.1840	0.2922	**0.4117**	**0.4434**	0.1243	0.1114	0.3044	0.3504
	Graphiti	No-WM	**0.1903**	**0.2736**	0.2867	0.3501	0.2905	0.3243	0.3543	0.3748	**0.1257**	**0.1277**	0.2572	0.2923
		S.M.-Only	0.1524	0.2131	0.3022	0.3604	0.3507	0.4615	0.3570	0.4023	0.1257	0.1277	**0.2589**	**0.3031**
		Ran.	0.1792	0.2587	0.3019	0.3658	0.3123	0.4231	**0.3614**	0.3839	0.0488	0.0426	0.2440	0.2823
		KGMARK	0.1538	0.2365	0.2878	0.3536	**0.3845**	**0.5556**	0.3589	**0.4088**	0.0887	0.0798	0.2505	0.3027
		MemMark	0.1389	0.2319	**0.4084**	**0.4865**	0.1865	0.3189	0.3364	0.3719	0.0701	0.0638	0.2453	0.2945
GLM-5	A-MEM	No-WM	0.2340	0.2868	0.3562	0.3908	0.2319	0.2452	**0.4677**	0.4592	0.0520	0.0438	0.2958	0.3067
		S.M.-Only	0.2342	0.3152	0.4144	**0.4689**	0.1927	0.2088	0.4463	**0.4621**	0.0408	0.0276	0.2939	0.3205
		Ran.	0.1875	0.2714	**0.4266**	0.4428	0.1838	0.2001	0.4516	0.4571	**0.0710**	**0.0643**	0.2971	0.3150
		MemMark	**0.3067**	**0.3491**	0.4139	0.4438	**0.2774**	**0.2803**	0.4489	0.4491	0.0670	0.0638	**0.3181**	**0.3300**
	Graphiti	No-WM	**0.1686**	**0.2731**	**0.2844**	0.3033	0.1474	0.1611	0.3487	0.3492	0.0914	0.0851	0.2338	0.2538
		S.M.-Only	0.1352	0.2203	0.2575	0.2713	0.1585	0.1519	0.3397	0.3633	0.0779	0.0673	0.2179	0.2395
		Ran.	0.1180	0.2154	0.2696	**0.3254**	0.1567	0.1628	0.3390	0.3501	0.0488	0.0426	0.2101	0.2390
		KGMARK	0.1473	0.2324	0.2436	0.2550	**0.2237**	**0.2643**	**0.3674**	**0.3773**	**0.1126**	**0.1064**	**0.2394**	**0.2599**
		MemMark	0.1060	0.1901	0.2518	0.2725	0.1199	0.1203	0.3532	0.3580	0.0968	0.0946	0.2188	0.2374
DeepSeek-V4-Pro	A-MEM	No-WM	0.3275	**0.3888**	0.4571	0.5476	0.2639	0.2912	**0.4899**	**0.4948**	0.1458	0.1254	0.3616	0.3870
		S.M.-Only	0.2768	0.3425	**0.4769**	**0.5743**	**0.3052**	**0.3911**	0.4873	0.4909	0.1632	0.1477	**0.3630**	**0.3950**
		Ran.	0.2105	0.2315	0.4359	0.5108	0.2335	0.3049	0.4605	0.4620	**0.2083**	**0.1883**	0.3413	0.3591
		MemMark	**0.3333**	0.3651	0.4279	0.4962	0.2597	0.3251	0.4679	0.4818	0.1667	0.1552	0.3541	0.3784
	Graphiti	No-WM	0.1276	0.1448	**0.3659**	**0.4108**	0.2616	0.2865	0.3436	**0.3588**	0.1249	0.1048	0.2560	**0.2694**
		S.M.-Only	0.1286	0.1769	0.2836	0.2998	0.2060	0.2105	0.3109	0.3162	**0.1626**	0.1323	0.2346	0.2404
		Ran.	**0.1653**	**0.1989**	0.3575	0.4050	0.1775	0.1835	**0.3585**	0.3555	0.1266	0.1039	**0.2606**	0.2689
		KGMARK	0.1304	0.1807	0.3213	0.3482	0.1815	0.2707	0.3470	0.3493	0.1432	**0.1361**	0.2484	0.2665
		MemMark	0.1104	0.1366	0.3353	0.3579	**0.2748**	**0.3334**	0.3229	0.3330	0.1277	0.1176	0.2418	0.2552

Table 2: **RQ1 — Utility preservation on LoCoMo**. Results are grouped by model and backend. `No-WM`, `S.M.-Only`, `Ran.`, and `KGMARK` denote unwatermarked execution, signed-metadata-only, random replacement, and KGMARK (Graphiti only); **MemMark** is the full method. Bold and underlined entries mark the best and second-best scores within each backend block. Averaged over the six model–backend settings, MEMMARK changes Overall F1 from 0.2816 to 0.2804, retaining 99.6% of the unwatermarked baseline, and changes BLEU-1 from 0.3069 to 0.3077, improved by 0.2%.

1.14, and 1.26 bits for update-target, link-target, and semantic-realization decisions, respectively. Thus, each carrier exposes roughly one bit of usable decision entropy per eligible write, and semantic realization is the highest-capacity carrier average.

Capacity follows backend structure rather than a fixed carrier recipe. For example, A-MEM distributes payload more evenly across carriers: with Qwen3.6-flash, the payload split is 10% update, 25% link, and 65% semantic; with GLM-5, it becomes 28%, 32%, and 40%. By contrast, GRAPHITI places more payload on semantic realization: semantic decisions carry 49%–75% of embedded bits across the three LLM backbones. The signal is therefore distributed over natural backend choices rather than forced into one artifact.

This distribution is important for portability. MEMMARK does not require two backends to expose the same internal objects; it only requires each backend to present admissible choices at write time. The carrier mix can change while the sampling and verification logic remains the same.

5.4 RQ3 — Snapshot-Only / Partial-Log Verification

RQ3 compares full-log verification (R1), partial logs with keep ratios $r \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$ (R2), and snapshot-only verification from in-record reveal evidence and the signed session anchor (R3).

Figure 3 shows that R3 matches full-log verification in the benign setting. MEMMARK recovers the full 40-bit payload from the final snapshot, giving a bit recovery rate of 1.00. In contrast, `signed-metadata-only` recovers 0.00, and the wrong-key control recovers only 0.15, near the $1/K$ chance level. Partial-log verification degrades smoothly with the amount of retained evidence. As the keep ratio increases from $r = 0.1$ to 0.9, mean R2 bit recovery rises from 0.08 to 0.29, 0.51, 0.71, and finally 0.90. This monotonic trend shows that verification is not an all-or-nothing artifact of the complete log: even partial evidence provides proportional attribution signal.

The gap to `signed-metadata-only` is the central RQ3 signal: metadata can authenticate a cooperative writer, but it does not bind the final memory state to latent evolution choices once the ex-

Family	Attack	Mode	Deepseek-V4-pro						Qwen3.6-flash						GLM-5					
			Rec			ΔWK			Rec			ΔWK			Rec			ΔWK		
			0.1	0.3	0.5	0.1	0.3	0.5	0.1	0.3	0.5	0.1	0.3	0.5	0.1	0.3	0.5	0.1	0.3	0.5
Content	Content-1	ComF	0.93	0.78	0.54	+0.73	+0.59	+0.34	0.95	0.60	0.45	+0.60	+0.25	+0.10	0.90	0.75	0.62	+0.75	+0.60	+0.47
	Content-2	ComF	0.83	0.68	0.54	+0.63	+0.49	+0.34	0.80	0.57	0.45	+0.45	+0.22	+0.10	0.82	0.68	0.35	+0.67	+0.53	+0.20
	Content-3	ComF	0.90	0.83	0.56	+0.71	+0.63	+0.37	0.90	0.72	0.50	+0.55	+0.38	+0.15	0.97	0.78	0.53	+0.82	+0.62	+0.38
	Content-4	ComF	0.98	0.83	0.63	+0.78	+0.63	+0.44	0.90	0.82	0.62	+0.55	+0.47	+0.28	0.85	0.50	0.25	+0.70	+0.35	+0.10
	Content-5	ComF	0.90	0.71	0.44	+0.71	+0.51	+0.24	0.90	0.53	0.35	+0.55	+0.18	+0.00	0.90	0.60	0.25	+0.75	+0.45	+0.10
Removal	Removal-1	Miss	1.00	1.00	1.00	+0.80	+0.80	+0.80	1.00	1.00	1.00	+0.65	+0.65	+0.65	1.00	1.00	1.00	+0.85	+0.85	+0.85
	Removal-2	Miss	1.00	1.00	1.00	+0.80	+0.80	+0.80	1.00	1.00	1.00	+0.65	+0.65	+0.65	1.00	1.00	1.00	+0.85	+0.85	+0.85
Synthesis	Synth-1	ComF	0.88	0.61	0.51	+0.68	+0.41	+0.32	0.90	0.68	0.40	+0.55	+0.33	+0.05	0.90	0.78	0.53	+0.75	+0.62	+0.38
	Synth-2	ComF	0.93	0.85	0.72	+0.74	+0.66	+0.52	0.93	0.80	0.71	+0.58	+0.45	+0.36	0.91	0.82	0.74	+0.76	+0.67	+0.59

Table 3: **RQ4 – Attack-specific recovery and wrong-key separation on A-MEM.** Each row reports the attack family, a compact row alias that maps to Table 9 (**Content-1–5** → attacks 1–5, **Removal-1–2** → attacks 6–7, and **Synth-1–2** → attacks 8–9), and the dominant verifier mode (`ComF` = `commitment_fail`; `Miss` = `missing_leaves`). For each model, **Rec** is post-attack bit recovery and $\Delta\mathrm{WK} = \mathrm{Rec} - \mathrm{WrongKey}$ compares recovery with that model's wrong-key R3 baseline at attack strengths $s \in \{0.1, 0.3, 0.5\}$. Positive ΔWK means the attacked snapshot retains more attribution signal than an incorrect key.

Backend	Method	Carrier Dist.	Ev. Rec.	Write Fail
A-MEM	`no-watermark`	284:90:477	0.597	4
	`signed-metadata-only`	251:135:462	0.560	3
	`random-replace`	168:205:471	0.504	2
	MEMMARK	237:143:465	0.533	3
GRAPHITI	`no-watermark`	288:481:834	0.236	0
	`signed-metadata-only`	274:422:917	0.198	4
	`random-replace`	264:384:920	0.176	1
	MEMMARK	278:386:938	0.230	5

Table 4: **RQ5 – Memory-integrity probes on LoCoMo.** All rows use the Qwen backbone. **Carrier Dist.** gives update:link:semantic counts; **Ev. Rec.** is evidence-grounded retrieval recall; **Write Fail** counts write-path failures.

ternal log is gone. In-record reveal evidence and the signed session anchor preserve enough replay material for snapshot-only attribution.

5.5 RQ4 — Robustness

We stress-test nine lifecycle attacks at strengths 0.1, 0.3, and 0.5, spanning content edits, removals, and synthesis-style mutations. We report post-attack recovery (Rec), the dominant verifier mode (Mode), and wrong-key separation ($\Delta\mathrm{WK} = \mathrm{Rec} - \mathrm{WrongKey}$).

Table 3 shows that non-removal attacks remain recoverable at mild strengths but diverge at strength 0.5. On A-MEM, content edits reduce average Rec from 0.896 to 0.472 as strength grows, while synthesis-style attacks are less destructive on average, ending at Rec 0.602. Additive poisoning is the mildest non-removal case at strength 0.5 (Rec 0.725, $\Delta\mathrm{WK} = +0.493$), whereas subgraph reanchoring is strongest, nearly reaching the wrong-key baseline (Rec 0.346, $\Delta\mathrm{WK} = +0.115$). Removal attacks behave differently: pruning and deduplication preserve surviving authenticated records, keeping Rec at 1.00 and producing `Miss`, while edits, compaction, and poisoning trigger `ComF`.

5.6 RQ5 — Memory Integrity

Finally, RQ5 checks whether watermarking materially changes the memory write path. Table 4 reports three LoCoMo probes: carrier allocation, evidence-grounded retrieval recall, and write-path failures. Overall, MEMMARK remains within the normal write regime: A-MEM retrieval recall retains 89.3% of the `no-watermark` baseline, with write failures changing from 4 to 3; GRAPHITI recall retains 97.5%, with only a small increase in write failures. Carrier use is also not concentrated in one decision type: with Qwen, semantic carriers account for 55.0% of carrier decisions on A-MEM and 58.6% on GRAPHITI, consistent with RQ2. These probes suggest that MEMMARK preserves snapshot verifiability without large observable write-path disruptions.

6 Conclusion

MEMMARK watermarks the state-evolution layer of long-term agent memory, binding provenance to backend write, update, linking, and retention choices. The results show that durable memory attribution can survive beyond visible actions, final text, and trusted metadata when it is tied to utility-preserving state choices and authenticated reveal evidence.

7 Limitations

MEMMARK is evaluated on two memory backends and one auditable benchmark, leaving broader deployment settings to future work. In particular, longer memory lifecycles, backend-specific compaction, migration, and periodic summarization may introduce new watermark carriers as well as new sources of drift. Future work should extend the adapter contract to cover these maintenance operations. The current attack study covers representative edits, deletions, and poisoning, but does not exhaust adaptive attempts to remove evidence while preserving utility. A natural next step is to study such attacks and build verification procedures that distinguish benign lifecycle changes from targeted tampering, making snapshot-only attribution more robust in production systems.

References

Tianyu Chen, Jian Lou, and Wenjie Wang. 2026. Safeguarding multimodal knowledge copyright in the RAG-as-a-service environment. In *The Fourteenth International Conference on Learning Representations*.

Zhaorun Chen, Zhen Xiang, Chaowei Xiao, Dawn Song, and Bo Li. 2024. Agentpoison: Red-teaming LLM agents via poisoning memory or knowledge bases. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Prateek Chhikara, Dev Khant, Saket Aryan, Taranjeet Singh, and Deshraj Yadav. 2025. Mem0: Building production-ready ai agents with scalable long-term memory. *arXiv preprint arXiv:2504.19413*.

Sumanth Dathathri, Abigail See, Sumedh Ghaisas, Po-Sen Huang, Rob McAdam, Johannes Welbl, Vandana Bachani, Alex Kaskasoli, Robert Stanforth, Tatiana Matejovicova, and 1 others. 2024. Scalable watermarking for identifying large language model outputs. *Nature*, 634(8035):818–823.

DeepSeek-AI. 2026. Deepseek-v4: Towards highly efficient million-token context intelligence.

Yudong Gao, Linghan Chen, Wenhan Wu, Mia Zhou, Jiyao Wang, Kaiyan Ji, Mingyu Guo, and Honglong Chen. 2026a. Bit-flip attacks on vision-language-action models: Action-decoding architecture shapes the vulnerability. *arXiv preprint arXiv:2608.15475*.

Yudong Gao, Qingyue Wang, Yuanyuan Yuan, Ruixuan Huang, Linghan Chen, Zimo Ji, and Shuai Wang. 2026b. Pathmark: Protecting intellectual property of mixture-of-expert llms via path watermarks. *arXiv preprint arXiv:2607.03688*.

GLM-5-Team, :, Aohan Zeng, Xin Lv, Zhenyu Hou, Zhengxiao Du, Qinkai Zheng, Bin Chen, Da Yin, Chendi Ge, Chenghua Huang, Chengxing Xie, Chenzheng Zhu, Congfeng Yin, Cunxiang Wang, Gengzheng Pan, Hao Zeng, Haoke Zhang, Haoran Wang, and 168 others. 2026. Glm-5: from vibe coding to agentic engineering. *Preprint*, arXiv:2602.15763.

Tsvetomir Hristov, Devriş İşler, Nikolaos Laoutaris, and Zekeriya Erkin. 2023. Graph database watermarking using pseudo-nodes. In *Proceedings of the Second ACM Data Economy Workshop*, pages 14–20.

Yuanzhe Hu, Yu Wang, and Julian McAuley. 2025. Evaluating memory in llm agents via incremental multi-turn interactions. *arXiv preprint arXiv:2507.05257*.

Kaibo Huang, Jin Tan, Yukun Wei, Wanling Li, Zipei Zhang, Hui Tian, Zhongliang Yang, and Linna Zhou. 2026a. Agentmark: Utility-preserving behavioral watermarking for agents. *arXiv preprint arXiv:2601.03294*.

Kaibo Huang, Zhongliang Yang, and Linna Zhou. 2025. Agent guide: A simple agent behavioral watermarking framework. *arXiv preprint arXiv:2504.05871*.

Ruiyang Huang, Chenxi Wang, Tinghe Zhang, Fengrui Liu, Jiayan Sun, Haocheng Wang, and Yifan Wu. 2026b. TopoSHIELD: Reshaping the flow of malice via spatio-temporal risk-aware topological evolution in multi-agent systems. In *Findings of the Association for Computational Linguistics: ACL 2026*, pages 8740–8768, San Diego, California, United States. Association for Computational Linguistics.

Nikola Jovanović, Robin Staab, Maximilian Baader, and Martin Vechev. 2025. Ward: Provable rag dataset inference via llm watermarks. In *International Conference on Learning Representations*, volume 2025, pages 93288–93314.

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

Ben Laurie, Adam Langley, and Emilia Kasper. 2013. Certificate transparency. *RFC 6962*.

Zhiyu Li, Shichao Song, Hanyu Wang, Simin Niu, Ding Chen, Jiawei Yang, Chenyang Xi, Huayi Lai, Jihao Zhao, Yezhaohui Wang, and 1 others. 2025. Memos: An operating system for memory-augmented generation (mag) in large language models. *arXiv preprint arXiv:2505.22101*.

Zehao Lin, Chunyu Li, and Kai Chen. 2026. A survey on the security of long-term memory in llm agents: Toward mnemonic sovereignty. *arXiv preprint arXiv:2604.16548*.

Yepeng Liu, Xuandong Zhao, Dawn Song, and Yuheng Bu. 2025. Dataset protection via watermarked canaries in retrieval-augmented llms. *arXiv preprint arXiv:2502.10673*.

Peizhuo Lv, Mengjie Sun, Hao Wang, Xiaofeng Wang, Shengzhi Zhang, Yuxuan Chen, Kai Chen, and Limin Sun. 2025. Rag-wm: An efficient black-box watermarking approach for retrieval-augmented generation of large language models. In *Proceedings of the 2025 ACM SIGSAC Conference on Computer and Communications Security*, pages 1709–1723.

Xingjun Ma, Yifeng Gao, Yixu Wang, Ruofan Wang, Xin Wang, and 1 others. 2026. Safety at scale: A comprehensive survey of large model and agent safety. *arXiv preprint arXiv:2502.05206*.

Adyasha Maharana, Dong-Ho Lee, Sergey Tulyakov, Mohit Bansal, Francesco Barbieri, and Yuwei Fang. 2024. Evaluating very long-term conversational memory of LLM agents. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13851–13870, Bangkok, Thailand. Association for Computational Linguistics.

Minjia Mao, Dongjun Wei, Zeyu Chen, Xiao Fang, and Michael Chau. 2025. Watermarking large language models: An unbiased and low-risk method. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7939–7960, Vienna, Austria. Association for Computational Linguistics.

Wenlong Meng, Chen Gong, Terry Yue Zhuo, Fan Zhang, Kecen Li, Zheng Liu, Zhou Yang, Chengkun Wei, and Wenzhi Chen. 2026. Watermarking llm agent trajectories. *arXiv preprint arXiv:2602.18700*.

Ralph C. Merkle. 1987. A digital signature based on a conventional encryption function. In *Advances in Cryptology – CRYPTO '87*, pages 369–378. Springer.

Charles Packer, Vivian Fang, Shishir G. Patil, Kevin Lin, Sarah Wooders, and Joseph E. Gonzalez. 2024. Memgpt: Towards llms as operating systems. In *Proceedings of the 41st International Conference on Machine Learning*.

Leyi Pan, Aiwei Liu, Zhiwei He, Zitian Gao, Xuandong Zhao, Yijian Lu, Binglin Zhou, Shuliang Liu, Xuming Hu, Lijie Wen, and 1 others. 2024. Marker-llm: An open-source toolkit for llm watermarking. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 61–71.

Joon Sung Park, Joseph C. O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. 2023. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology*. Association for Computing Machinery.

Torben Pryds Pedersen. 1991. Non-interactive and information-theoretic secure verifiable secret sharing. In *Advances in Cryptology – CRYPTO '91*, pages 129–140. Springer.

Hongrui Peng, Haolang Lu, Yuanlong Yu, WeiYe Fu, Kun Wang, and Guoshun Nan. 2025. KGMark: A diffusion watermark for knowledge graphs. In *Forty-second International Conference on Machine Learning*.

Qwen Team. 2026. Qwen3.6-35B-A3B: Agentic coding power, now open to all.

Preston Rasmussen, Pavlo Paliychuk, Travis Beauvais, Jack Ryan, and Daniel Chalef. 2025. Zep: a temporal knowledge graph architecture for agent memory. *arXiv preprint arXiv:2501.13956*.

Saksham Sahai Srivastava and Haoyu He. 2025. Memorygraft: Persistent compromise of llm agents via poisoned experience retrieval. *arXiv preprint arXiv:2512.16962*.

Haoran Sun, Zekun Zhang, and Shaoning Zeng. 2025. Preference-aware memory update for long-term llm agents. *arXiv preprint arXiv:2510.09720*.

Liwen Wang, Zongjie Li, Yuchong Xie, Shuai Wang, Dongdong She, Wei Wang, and Juergen Rahmel. 2026a. On protecting agentic systems' intellectual property via watermarking. *arXiv preprint arXiv:2602.08401*.

Shu Wang, Edwin Yu, Oscar Love, Tom Zhang, Tom Wong, Steve Scargall, and Charles Fan. 2026b. Memmachine: A ground-truth-preserving memory system for personalized ai agents. *arXiv preprint arXiv:2604.04853*.

Qianshan Wei, Tengchao Yang, Yaochen Wang, Xinfeng Li, Lijun Li, Zhenfei Yin, Yi Zhan, Thorsten Holz, Zhiqiang Lin, and XiaoFeng Wang. 2025. A-memguard: A proactive defense framework for llm-based agent memory. *arXiv preprint arXiv:2510.02373*.

Di Wu, Hongwei Wang, Wenhao Yu, Yuwei Zhang, Kai-Wei Chang, and Dong Yu. 2025. Longmemeval: Benchmarking chat assistants on long-term interactive memory. In *International Conference on Learning Representations*.

Wujiang Xu, Zujie Liang, Kai Mei, Hang Gao, Juntao Tan, and Yongfeng Zhang. 2026a. A-mem: Agentic memory for llm agents. *Advances in Neural Information Processing Systems*, 38:17577–17604.

Zhenhua Xu, Haobo Zhang, Zhebo Wang, Qichen Liu, Haitao Xu, Wenpeng Xing, and Meng Han. 2026b. Forgetmark: Stealthy fingerprint embedding via targeted unlearning in language models. In *ICASSP 2026-2026 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 13912–13916. IEEE.

Sikuan Yan, Xiufeng Yang, Zuchao Huang, Ercong Nie, Zifeng Ding, Zonggen Li, Xiaowen Ma, Jinhe Bi, Kristian Kersting, Jeff Z Pan, and 1 others. 2025. Memory-r1: Enhancing large language model agents to manage and utilize memories via reinforcement learning. *arXiv preprint arXiv:2508.19828*.

Jimiao Yu, Honglong Chen, Junjian Li, Linghan Chen, Yudong Gao, Weifeng Liu, and Lei Zhang. 2025. Black-box adversarial defense based on image decomposition and reconstruction. *IEEE Transactions on Multimedia*, 27:5909–5921.

Guilin Zhang, Wei Jiang, Xiejiashan Wang, Aisha Behr, Kai Zhao, Jeffrey Friedman, Xu Chu, and Amine Anoun. 2026a. Adaptive memory admission control for llm agents. *arXiv preprint arXiv:2603.04549*.

Haobo Zhang, Zhenhua Xu, Junxian Li, Shangfeng Sheng, Dezhang Kong, and Meng Han. 2026b. Attndiff: Attention-based differential fingerprinting for large language models. *Preprint*, arXiv:2604.05502.

Yuxiang Zhang, Jiangming Shu, Ye Ma, Xueyuan Lin, Shangxi Wu, and Jitao Sang. 2025. Memory as action: Autonomous context curation for long-horizon agentic tasks. *arXiv preprint arXiv:2510.12635*.

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye Wang, and Yanlin Wang. 2024. Memorybank: Enhancing large language models with long-term memory. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(17):19724–19731.

Qiming Zhu, Shunian Chen, Rui Yu, Zhehao Wu, and Benyou Wang. 2026. From lossy to verified: A provenance-aware tiered memory for agents. *arXiv preprint arXiv:2602.17913*.

A Experiment Details

A.1 Data Statistics

We evaluate on the public LoCoMo benchmark, using the same fixed set of ten long-term multi-session conversations for all main model–backend configurations. This fixed-conversation protocol ensures that comparisons across methods, memory backends, and LLM backbones are made on the same evaluation instances rather than on backend- or model-specific subsets.

LoCoMo contains 10 conversations. In the release, each conversation contains an average of 27.2 sessions, 21.6 turns per session, and 16,618.1 tokens per conversation. The average dialogue turn contains 29.8 tokens. LoCoMo also provides derived memory artifacts: observations average 19.2 tokens each, and session summaries average 132.4 tokens. The benchmark is multimodal as well, with an average of 91.2 images per conversation, although our experiments use LoCoMo for memory-oriented QA evaluation rather than multimodal generation.

For the QA benchmark, LoCoMo provides 1,986 total questions across five reasoning categories. These include 841 single-hop questions (42.3%), 282 multi-hop questions (14.2%), 321 temporal-reasoning questions (16.1%), 96 open-domain knowledge questions (4.8%), and 446 adversarial questions (22.4%). Single-hop questions require evidence from one session, multi-hop questions require synthesizing information across multiple sessions, temporal questions require reasoning over time cues, open-domain questions require combining dialogue information with general knowledge, and adversarial questions are designed to be unanswerable or misleading.

A.2 Artifact Attribution and Licenses

We use these artifacts under their stated access conditions and licenses. The A-MEM reproduction repository is released under the MIT License, and GRAPHITI is released under the Apache License 2.0. For LoCoMo, we use the dataset and code for research evaluation according to the terms provided with the released repository. For model backbones, we access the models through their public model releases or APIs and follow the corresponding model licenses, platform terms, and usage policies.

A.3 Consistency with Intended Use

Our use of the artifacts is consistent with their intended research purposes. LoCoMo is used only as an evaluation benchmark for long-term conversational memory. We do not use LoCoMo-derived data for deployment, user profiling, or non-research applications. A-MEM and GRAPHITI are used as memory-system substrates, matching their intended role as agent-memory frameworks. The LLM backbones are used as controlled model components for candidate generation, memory writing, and memory-based QA; they are not fine-tuned on LoCoMo, and we do not redistribute model weights beyond the access conditions of the original providers.

A.4 Potential Risks

MEMMARK introduces a provenance mechanism for long-term agent memory, but it also creates risks that must be considered in deployment. First, watermark verification depends on secret keys,

canonicalized reveal evidence, and cryptographic audit material. If keys are leaked, sidecars are mishandled, or canonical serialization is inconsistent across systems, verification may become unreliable. Second, successful watermark verification should not be interpreted as a guarantee that the memory content is true or safe. It only shows that the observed memory evolution is consistent with the keyed sampler and audit trace.

B Full Proofs

This appendix collects the technical material that supports the main paper's central claims but would be too detailed for the main narrative. We begin with proof sketches for the sampler properties, then define the metrics and baselines before reporting the experimental appendix results referenced in the main text.

B.1 Proof of Lemma 1

Let $\mathcal{C}_t = \{c^1, \ldots, c^k\}$ and write $p_i = \pi_t(c^i)$. The sampler uses the same integer-binning argument as AGENTMARK (Huang et al., 2026a). Choose an audit precision N and represent the serialized distribution by non-negative integer masses n_i with $\sum_i n_i = N$ and $n_i/N = p_i$. In practice π_t is the canonical, finite-precision distribution stored in the reveal record, so the equality is exact with respect to the audited distribution.

Partition the cyclic group $\mathbb{Z}_N$ into consecutive intervals I_i with $|I_i| = n_i$, one interval per candidate. For any payload position $x \in \mathbb{Z}_N$, the keyed sampler draws a shift $s = \mathrm{PRF}(\mathsf{K}, \mathrm{ctx}_t) \bmod N$ and selects the unique c^i such that $(x+s) \bmod N \in I_i$. In the ideal experiment s is uniform on $\mathbb{Z}_N$; under a secure PRF it is computationally indistinguishable from uniform to an observer without K. Therefore $(x + s) \bmod N$ is uniform regardless of the payload point x, and

$$\begin{aligned}\Pr[\hat{c}_t = c^i] &= \Pr[(x + s) \bmod N \in I_i] \\ &= \frac{|I_i|}{N} = p_i.\end{aligned}$$

Thus the keyed pick has exactly the same marginal distribution as self-reported sampling from π_t. The notation $\Pr[\cdot \mid \mathsf{K}, \mathrm{ctx}_t]_{\mathrm{marg}}$ in the lemma should be read as the standard watermarking marginal over the PRF-key experiment, or equivalently over the pseudorandom shift that the hidden key induces. After a concrete key and context are fixed, the selector is of course deterministic and replayable by the verifier.

B.2 Proof of Lemma 2

Consider one memory event that triggers m internal LLM calls. For call j, the adapter exposes $(\mathcal{C}_{t,j}, \pi_{t,j}, \mathrm{ctx}_{t,j})$, where $\mathrm{ctx}_{t,j}$ contains the round index, dialogue identifiers, prompt hash, and the previous commitment. These fields domain-separate the calls: except with negligible collision probability, $\mathrm{ctx}_{t,j} \neq \mathrm{ctx}_{t,\ell}$ for $j \neq \ell$.

The backend may choose later contexts adaptively after observing earlier selected candidates. PRF security covers such adaptively chosen distinct inputs, so the sequence $\mathrm{PRF}(\mathsf{K}, \mathrm{ctx}_{t,1}), \ldots, \mathrm{PRF}(\mathsf{K}, \mathrm{ctx}_{t,m})$ is computationally indistinguishable from independent uniform draws. By Lemma 1, each individual keyed pick preserves its own marginal distribution:

$$\Pr[\hat{c}_{t,j} = c \mid \mathcal{C}_{t,j}, \pi_{t,j}, \mathrm{ctx}_{t,j}] = \pi_{t,j}(c).$$

Independence of the PRF outputs then gives the product-form joint distribution conditioned on the exposed decision tuples. If $B_{t,j}$ is the number of payload bits embedded at decision j, total capacity over the cascade is $B_t = \sum_{j=1}^{m} B_{t,j}$ and hence $\mathbb{E}[B_t] = \sum_{j=1}^{m} \mathbb{E}[B_{t,j}]$ by linearity of expectation. Since every stage is marginally distribution preserving, there is no accumulating sampling bias across the cascade; additional calls add evidence and capacity, not systematic drift.

B.3 Proof of Lemma 3

Fix any backend $\mathcal{B}$ that implements the adapter interface and emits a valid decision tuple $(\mathcal{C}_t, \pi_t, \mathrm{ctx}_t)$. The sampler receives only this tuple and the watermark secret. It does not inspect whether the candidates are A-MEM notes, GRAPHITI entities, graph edges, or surface realizations. Applying Lemma 1 to the emitted tuple therefore yields

$$\Pr_{\mathcal{B}}[\hat{c}_t = c^i \mid \mathcal{C}_t, \pi_t, \mathrm{ctx}_t] = \pi_t(c^i),$$

with no term depending on $\mathcal{B}$ beyond the tuple itself. If two backends expose the same $(\mathcal{C}_t, \pi_t, \mathrm{ctx}_t)$, the keyed sampler induces the same marginal law and the same replay rule. This does not assert that all backends produce identical candidate distributions; rather, it states that once a backend has reduced its native write choice to the common adapter representation, the watermark layer supplies the same distribution-preservation guarantee. This is the invariant used by the cross-backend comparisons in the main text.

C Metrics

Table 5 consolidates the definitions of all metrics used in this paper, grouped by the research question they address: utility (RQ1), capacity (RQ2), snapshot-only verification (RQ3), tamper detection (RQ4), and memory integrity (RQ5). For each metric we list its reporting unit and, where relevant, the carrier-level decomposition over update target, link target, and semantic realization. We refer the reader to this table throughout Sections 5.2–5.6 for precise definitions.

D Details of Baselines

This appendix clarifies the baseline definitions used in §5.1 and Table 2.

No-WM. `No-WM` is the unwatermarked execution path. The memory backend, agent harness, and evaluation pipeline are identical to the MEMMARK runs; the only difference is that no attribution logic is applied during memory-state evolution. It is therefore the utility reference for RQ1 and RQ5.

S.M.-Only. `S.M.-Only` keeps the signed metadata and audit-sidecar machinery but removes keyed selection over admissible candidates. This control tests whether attribution can be explained by explicit metadata alone rather than by hidden state-evolution choices. In R3, this baseline can validate that a sidecar was signed, but it carries no keyed payload and therefore cannot recover writer-specific state-evolution bits.

Ran. `Ran.` replaces keyed sampling with random selection among the same admissible candidates. It therefore controls for the possibility that any observed utility change is caused by multi-candidate enumeration itself rather than by the secret-keyed selection rule. Since the choice is not reproducible from K, it is not expected to verify as MEMMARK evidence.

KGMARK. `KGMARK` (Peng et al., 2025) is included only for GRAPHITI, where a knowledge-graph baseline is structurally meaningful. It is omitted for A-MEM because A-MEM does not expose graph-native edge operations of the kind assumed by KGMARK. This baseline is intentionally narrower than MEMMARK: it tests a graph-specialized watermark against a backend-invariant state-evolution watermark.

E Hyperparameter Sensitivity

We perform a default-centered one-factor sensitivity analysis on Qwen3.6-flash with the A-MEM backend. Starting from the default $K{=}4, T_{\text{enum}}{=}0.7$, we vary $K \in \{2, 4, 8\}$ while holding $T_{\text{enum}}{=}0.7$, and vary $T_{\text{enum}} \in \{0.5, 0.7, 1.0\}$ while holding $K{=}4$. This isolates the two expected trade-offs: candidate count controls the capacity and wrong-key chance level, while enumeration temperature controls the diversity–utility balance.

Table 7 shows that the default setting is a stable middle point rather than a tuned extreme. Reducing K to 2 lowers the realized capacity and pushes wrong-key recovery toward the $1/2$ chance-level reference. Increasing K to 8 improves bits per decision and separates wrong-key recovery more strongly, but the utility delta becomes more negative, consistent with lower-quality candidates entering the admissible set. Temperature has a similar but softer effect: $T_{\text{enum}}{=}0.5$ is conservative and slightly improves utility, whereas $T_{\text{enum}}{=}1.0$ increases capacity at the cost of a larger F1 drop. R3 recovery remains complete across the sweep, indicating that these settings affect capacity and perturbation more than benign snapshot verifiability.

F Overall and Per-Conversation Experimental Results

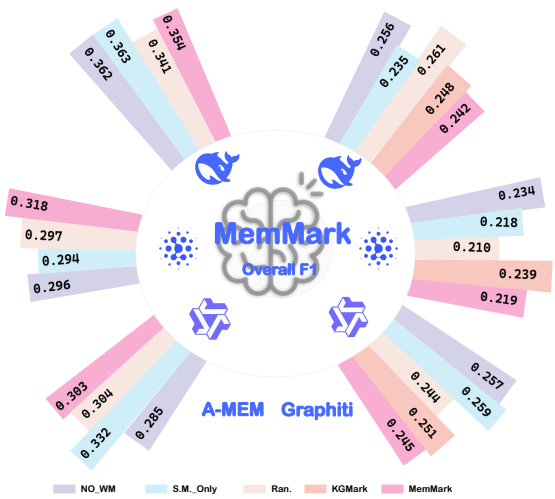

Figure 4: **Overall F1 comparison**. Overall F1 from Table 2 across three LLMs and two memory backends, comparing MEMMARK with baselines.

Axis (RQ)	Metric	Definition / Reporting Unit
Utility preservation (RQ1)	LoCoMo F1	Precision and recall computed over predicted versus gold answer items, combined as their harmonic mean.
	BLEU-1	Unigram precision between the generated answer and the reference answer, measuring lexical overlap at the token level.
	ΔF1 vs. no-watermark	Difference between the MEMMARK F1 and the corresponding `no-watermark` F1 under the same model, backend, and conversation.
Capacity (RQ2)	Per-carrier entropy $H(\pi)$	Shannon entropy of the self-reported candidate distribution, reported by carrier and averaged over the evaluated model–backend settings.
	Per-carrier payload share	Fraction of embedded payload bits carried by each memory-evolution carrier, normalized within each model–backend setting.
	bits/decision	Realized watermark capacity in the hyperparameter sweep: recovered payload bits divided by the number of watermarkable decisions.
Verification (RQ3) (in-record / snapshot-only)	R1/R2/R3 bit recovery	Fraction of embedded watermark bits correctly recovered from the available evidence under full-log R1, partial-log R2, and snapshot-only R3 verification.
	Wrong Key	Observed bit-level recovery when verification is run with an incorrect key; for the default $K{=}4$ setting, the chance-level reference is $1/K{=}0.25$.
Robustness (RQ4) (nine memory-lifecycle attacks)	Rec	Post-attack bit recovery over the surviving verifiable records, reported per attack type and strength.
	Mode	Dominant verifier signal triggered by the lifecycle attack: `ComF` for `commitment_fail` and `Miss` for `missing_leaves`.
	ΔWK	Difference between post-attack recovery and the corresponding R3 wrong-key baseline: $\mathrm{Rec} - \mathrm{WrongKey}$. Positive values indicate that the attacked snapshot still carries more attribution signal than an incorrect key.
Memory integrity (RQ5)	Carrier Dist.	Counts of watermark-carrying decisions by carrier, reported as update:link:semantic.
	Ev. Rec.	Fraction of QA-time gold evidence records successfully retrieved from memory.
	Write Fail	Number or fraction of attempted memory writes that fail validation, storage, or commitment construction, as specified by the table.

Table 5: Metric definitions across the five research questions.

Baseline	Selection rule	Evidence retained	Isolates	Backend
`No-WM`	Native backend execution; no keyed candidate selection.	No watermark payload, reveal record, or watermark-specific Merkle leaf.	Utility of the uninstrumented memory path.	A-MEM GRAPHITI
`S.M.-Only`	Samples from the backend distribution without embedding payload bits.	Signed metadata, in-record sidecar, commitments, and session anchor.	Whether authenticated metadata alone can support attribution.	A-MEM GRAPHITI
`Ran.`	Chooses randomly among the same admissible candidates, without the secret key.	Candidate enumeration and ordinary evaluation artifacts, but no keyed payload.	Perturbation from multi-candidate enumeration independent of watermarking.	A-MEM GRAPHITI
`KGMARK`	Applies the closest graph-watermark baseline to graph-native memory state.	KG-level watermark evidence rather than MEMMARK state-evolution evidence.	Comparison with a structured-memory watermark specialized to graphs.	GRAPHITI only

Table 6: **Baseline definitions**. The controls separate native memory quality, signed provenance metadata, unkeyed candidate randomization, and a graph-specific watermark baseline.

K	T_{enum}	bits/dec.	ΔF1	R3 Recov.	Wrong Key
2	0.7	0.0312	+0.004	1.000	0.492
4	0.5	0.0415	+0.009	1.000	0.221
4	0.7	0.0478	+0.006	1.000	0.205
4	1.0	0.0526	-0.014	1.000	0.236
8	0.7	0.0619	-0.019	1.000	0.119

Table 7: **Hyperparameter sensitivity around the default setting.** We vary one parameter at a time around K=4, T_{enum}=0.7 on Qwen3.6-flash with the A-MEM backend. Lower K reduces capacity and weakens wrong-key separation; higher K increases capacity but can perturb utility by admitting lower-quality candidates. Higher T_{enum} increases candidate diversity but also increases semantic drift.

Figure 4 provides an appendix visualization of the main experimental table, focusing on overall F1 across the three LLMs and the two backends. We then provide the per-conversation breakdown for Qwen3.6-flash on all 10 LoCoMo conversations. This table is included only in the appendix because its role is diagnostic rather than conceptual: it shows that the aggregate trends in the main text are not driven by a single conversation outlier.

The overall plot makes two points visible at once. First, MEMMARK tracks the main utility curve rather than collapsing it: across the three LLMs, the watermarked runs remain in the same regime as their unwatermarked counterparts. Second, the backend effect is larger than the watermark effect. A-MEM is generally more stable under the watermarking wrapper, while GRAPHITI shows a larger spread and a slightly stronger drop in the hardest settings. That is exactly the pattern expected from a memory system whose candidate space is backend-dependent but still compatible with the same sampler.

Table 8 shows the same story at the conversation level. The A-MEM deltas stay small, mostly within a few points, and are centered close to zero, with a mean gap of $+0.006$. GRAPHITI is more variable and more sensitive to conversation-specific structure, but the mean gap remains modest at -0.038 over the all-conversation Qwen3.6-flash diagnostic. In other words, the watermark does not introduce a single brittle failure mode; it behaves like a mild shift on top of ordinary conversational variance. The verification columns remain stable across conversations: R1 and R3 recover the payload completely, R2 at r=0.5 averages 0.556, and the wrong-key control averages 0.205, close to the $1/K$=0.25 chance reference. Thus, the capacity signal survives the same diversity that perturbs utility while remaining key-specific.

G Memory-lifecycle Attacks and Backend Diagnostics

We evaluate robustness (RQ4) under nine attacks that span the realistic lifecycle of an agent's memory store. The attacks fall into three operational families. *Content-tamper* attacks (#1–#5) mutate the contents of an audit record — its `probabilities` field, decision context `ctx_t`, selected candidate, or candidate payload — while leaving the leaf set of the Merkle tree intact; such attacks should be caught by per-record commitment verification and surface as a `commitment_fail` signal. *Leaf-removal* attacks (#6–#7) remove authenticated records and should surface as `missing_leaves`. *Synthesis/restructuring* attacks (#8–#9) collapse candidate sets or inject fabricated audit records, and should be caught by commitment verification when a leaf is altered or added. Together the nine attacks cover silent edits, semantics-preserving rewrites, fact supersession, knowledge-graph edge relabeling and subgraph reanchoring, pruning, deduplication, compaction, and poisoning. Each attack is parameterized by a strength level, and the operation, low-level verifier signal, and literature analogue are summarized in Table 9. The RQ4 robustness tables then report these attacks with Rec, Mode, and ΔWK to measure both recoverable attribution and separation from the wrong-key baseline.

G.1 Graphiti Backend Robustness

Table 10 repeats the RQ4 robustness breakdown for the GRAPHITI backend using the same Rec, Mode, and ΔWK metrics as the A-MEM table. The Mode column keeps the low-level diagnosis: in-place edits and synthesis-style rewrites surface as `ComF`, while pruning and deduplication surface as `Miss`. The ΔWK columns compare each attacked snapshot with the corresponding model's wrong-key R3 baseline, so the table reports both recovery and key-specific attribution separation.

The graph backend shows the same qualitative pattern as A-MEM but with backend-specific sensitivity. Content attacks have average Rec values of 0.911, 0.694, and 0.489 at strengths 0.1, 0.3, and 0.5, with average ΔWK values of $+0.629$, $+0.412$, and $+0.207$. Synthesis attacks recover 0.858, 0.707, and 0.583, with ΔWK values of

		A-MEM			GRAPHITI			Watermark Metrics			
Conv	QAs	F1 wm	F1 nwm	Δ	F1 wm	F1 nwm	Δ	R1	R2 $r{=}0.5$	R3	Wrong Key
0	199	0.304	0.285	+0.019	0.245	0.257	-0.012	1.000	0.600	1.000	0.175
1	105	0.365	0.393	-0.028	0.303	0.388	-0.084	1.000	0.475	1.000	0.212
2	193	0.389	0.393	-0.004	0.274	0.268	+0.006	1.000	0.610	1.000	0.234
3	260	0.270	0.249	+0.021	0.181	0.257	-0.076	1.000	0.575	1.000	0.198
4	242	0.362	0.348	+0.014	0.162	0.260	-0.097	1.000	0.700	1.000	0.241
5	158	0.248	0.283	-0.035	0.250	0.255	-0.005	1.000	0.500	1.000	0.194
6	190	0.372	0.384	-0.012	0.259	0.247	+0.012	1.000	0.525	1.000	0.145
7	239	0.331	0.303	+0.027	0.212	0.216	-0.004	1.000	0.475	1.000	0.265
8	196	0.300	0.276	+0.024	0.252	0.313	-0.061	1.000	0.600	1.000	0.205
9	204	0.333	0.300	+0.033	0.211	0.273	-0.061	1.000	0.500	1.000	0.179
Mean	198.6	0.327	0.322	+0.006	0.235	0.273	-0.038	1.000	0.556	1.000	0.205

Table 8: **Qwen3.6-flash per-conversation breakdown on all 10 LoCoMo samples**. F1 wm / nwm = MEMMARK vs `no-watermark` F1 (set formula). Watermark metrics use the A-MEM backend. Wrong Key reports observed bit-level recovery under an incorrect key; under the default $K{=}4$ setting, the chance-level reference is $1/K{=}0.25$. Conversation 0 is from the standalone Qwen markdown summaries; conversations 1–9 are from the full-result JSON files.

#	Attack	Operation on the audit record	Low-level signal	Inspiration / analogue
Content-tamper attacks — modify committed record bytes in place.				
1	`manual_edits`	Silently mutate the `probabilities` field of a record while leaving all other fields and the leaf set intact.	`commitment_fail`	MemoryGraft / generic record tampering.
2	`para._rewrite`	Append a `[PARAPHRASE]` marker to `ctx_t`, simulating a semantics-preserving rewrite of the context that produced the decision.	`commitment_fail`	RAG-WM paraphrase attack.
3	`supersession`	Replace `selected_candidate_id` with a sibling candidate from the same carrier, simulating a newer fact overwriting an older one.	`commitment_fail`	Graphiti native fact-invalidation chain.
4	`edge_relabel` (KG)	Append `[RELABEL]` to the selected candidate's `payload.text`, simulating relabeling of an entity–relation edge in the knowledge graph.	`commitment_fail`	KGMark edge perturbation.
5	`subgraph_reanch.` (KG)	Append `[REANCHOR]` to `ctx_t` and rotate the candidate list, simulating a change of root anchor for a subgraph.	`commitment_fail`	KGMark anchor swap.
Leaf-removal attacks — remove authenticated leaves from the trace.				
6	`pruning`	Delete a fraction of audit-record leaves uniformly at random, controlled by the attack strength.	`missing_leaves`	Memory-lifecycle pruning / KGMark subgraph removal.
7	`dedup`	Find records duplicated by selected payload text and remove secondary copies, retaining only the canonical leaf.	`missing_leaves`	Memory-lifecycle deduplication.
Synthesis/restructuring attacks — rewrite or add records without valid openings.				
8	`compaction`	Collapse the candidate set by removing one candidate, simulating multiple memories being merged into a single summary.	`commitment_fail`	Memory-lifecycle compaction.
9	`poisoning`	Inject fabricated audit records into the leaf set (additive, not deletive).	`commitment_fail`	A-MemGuard / KGMark node insertion.

Table 9: The nine memory-lifecycle attacks used in RQ4. They are grouped by lifecycle operation and list the concrete audit-record mutation, the expected low-level verifier signal, and the motivating analogue. Tables 3 and 10 then report Rec, Mode, and ΔWK for these attacks.

$+0.577$, $+0.426$, and $+0.301$. Compaction is the most damaging synthesis case at strength 0.5 (mean Rec 0.462, ΔWK $+0.180$), whereas poisoning remains more recoverable because it is additive

Family	Attack	Mode	Deepseek-V4-pro						Qwen3.6-flash						GLM-5					
			Rec			ΔWK			Rec			ΔWK			Rec			ΔWK		
			0.1	0.3	0.5	0.1	0.3	0.5	0.1	0.3	0.5	0.1	0.3	0.5	0.1	0.3	0.5	0.1	0.3	0.5
Content	Content-1	ComF	0.97	0.65	0.45	+0.68	+0.35	+0.15	0.82	0.65	0.55	+0.47	+0.30	+0.20	0.93	0.68	0.37	+0.73	+0.49	+0.17
	Content-2	ComF	0.85	0.78	0.50	+0.55	+0.48	+0.20	0.90	0.70	0.50	+0.55	+0.35	+0.15	0.98	0.76	0.51	+0.78	+0.56	+0.32
	Content-3	ComF	0.85	0.60	0.45	+0.55	+0.30	+0.15	0.90	0.62	0.47	+0.55	+0.28	+0.12	0.90	0.76	0.71	+0.71	+0.56	+0.51
	Content-4	ComF	0.93	0.57	0.40	+0.62	+0.27	+0.10	0.88	0.65	0.42	+0.53	+0.30	+0.08	0.90	0.68	0.41	+0.71	+0.49	+0.22
	Content-5	ComF	0.93	0.68	0.42	+0.62	+0.38	+0.12	0.95	0.90	0.60	+0.60	+0.55	+0.25	0.98	0.73	0.56	+0.78	+0.54	+0.37
Removal	Removal-1	Miss	1.00	1.00	1.00	+0.70	+0.70	+0.70	1.00	1.00	1.00	+0.65	+0.65	+0.65	1.00	1.00	1.00	+0.80	+0.80	+0.80
	Removal-2	Miss	1.00	1.00	1.00	+0.70	+0.70	+0.70	1.00	1.00	1.00	+0.65	+0.65	+0.65	1.00	1.00	1.00	+0.80	+0.80	+0.80
Synthesis	Synth-1	ComF	0.72	0.42	0.35	+0.42	+0.12	+0.05	0.93	0.75	0.45	+0.58	+0.40	+0.10	0.78	0.68	0.59	+0.59	+0.49	+0.39
	Synth-2	ComF	0.95	0.93	0.85	+0.65	+0.63	+0.55	0.93	0.67	0.56	+0.58	+0.32	+0.21	0.84	0.79	0.71	+0.64	+0.59	+0.51

Table 10: **RQ4 – Attack-specific recovery and wrong-key separation on GRAPHITI.** Each row reports the attack family, a compact row alias that maps to Table 9 (**Content-1–5** → attacks 1–5, **Removal-1–2** → attacks 6–7, and **Synth-1–2** → attacks 8–9), and the dominant verifier mode (`ComF` = `commitment_fail`; `Miss` = `missing_leaves`). For each model, **Rec** is post-attack bit recovery and $\Delta\mathrm{WK} = \mathrm{Rec} - \mathrm{WrongKey}$ compares recovery with that model's wrong-key R3 baseline at attack strengths $s \in \{0.1, 0.3, 0.5\}$. Positive ΔWK means the attacked snapshot retains more attribution signal than an incorrect key.

(mean Rec 0.705, $\Delta\mathrm{WK}$ $+0.423$).

Removal attacks preserve a recovery rate of 1.00 over the surviving records and retain a strong wrong-key margin ($\Delta\mathrm{WK} = +0.718$ on average), while their `Miss` mode exposes the deletion-style lifecycle mutation. This is the desired behavior: authenticated surviving records remain replayable, but the verifier still identifies that evidence has disappeared from the anchored trace.