LTM-RAG: Longitudinal Trust Mining for Poison-Resilient Retrieval-Augmented Generation

Thi-Hong-Cuc Le
Ho Chi Minh City University of Technology (HCMUT)
Vietnam National University Ho Chi Minh City
Vietnam
lthcuc.sdh242@hcmut.edu.vn

Phuong Huu Vu Tran
Vietnamese-German University
Vietnam
10425032@student.vgu.edu.vn

Thuan Mai Duc
Ho Chi Minh City University of Technology (HCMUT)
Vietnam National University Ho Chi Minh City
Vietnam
thuan.maiduc2411@hcmut.edu.vn

Xuan-Bach Le*
Ho Chi Minh City University of Technology (HCMUT)
Vietnam National University Ho Chi Minh City
Vietnam
lexuanbach@hcmut.edu.vn

***Abstract*—Poisoning attacks on retrieval-augmented generation are rarely localized to a single passage. They are recurring campaigns whose evidence is structured across time, yet most existing defenses inspect one query in isolation and discard whatever they learned before the next query arrives. We recast the problem as longitudinal stream mining over a RAG pipeline's evolving sources, chunks, motifs, and lineage. Our framework, LTM-RAG, maintains four trust memories under decayed Beta pseudo-counts, bounded lineage propagation, and verifier-aware evidence attenuation, updated prequentially so that feedback from window t governs only subsequent retrievals. We pair this longitudinal gate with a source-normalized top-K aggregator and recommend their Hybrid as the deployment configuration: mining handles recurring evidence, and query-local aggregation handles first exposure. On a 36-window prequential benchmark across five attack schedules, the Hybrid drops answer-level attack success rate (AnsASR) from 94.8% to 14.0% on the burst persistent-source scenario (5 seeds; BM25 fixed-gate recurrence stack with delayed known attribution). A separately logged live-judge diagnostic reports lower AnsASR for LTM-RAG and Hybrid than for Vanilla within that protocol ($n \geq 1200$ per row). Controlled experiments characterize component effects and temporal behavior under the stated protocols, and report time-to-distrust, clean utility, and subgroup false-reputation damage, with trust regret defined and bounded for the controlled loss. The scope is bounded: the framework does not cleanse first-exposure poison or defeat white-box optimization, and an attacker who simultaneously breaks motif, lineage, and verifier-feedback assumptions can neutralize the gate.**

***Index Terms*—retrieval-augmented generation, data poisoning, streaming data mining, trust memory, provenance, temporal drift, adversarial robustness**

I. Introduction

Retrieval-augmented generation grounds a language model in external evidence, and that grounding is also its attack surface. An adversary who injects, mirrors, or modifies a small fraction of indexed documents can steer answers while retaining the fluency and appearance of legitimate support [1]–[4]. The defenses we inherit largely act inside a single query, through filtering, robust aggregation, source-reliability estimation, and forensic traceback [5]–[9], and they discard what they learned before the next query arrives.

A per-query decision sees only a snapshot: one suspicious passage is too ambiguous to filter, a mostly clean source can build credibility before shipping targeted poison, and a false claim copied across sites can look like independent corroboration until its shared lineage is exposed. Evidence of a recurring campaign accumulates *across* queries; no single query contains it. We therefore model RAG poisoning as stream mining over sources, chunks, motifs, and lineage. At window t we score retrievals using only the state H_t accumulated from earlier feedback, and the current window's verifier and attribution events update the state for the *next* round.

Figure 1 sketches LTM-RAG. We maintain four trust memories: source θ_s, chunk θ_d, motif μ_k, and lineage $\mathcal{G}_t$. A trust-aware reranker combines lexical relevance with these memories under temporal decay, uncertainty-aware priors, and a source-diversity cap. After generation, verifier and attribution events advance the state to H_{t+1} under calibration κ_v and asymmetric aging (ρ_+, ρ_-). Table I contrasts existing defenses on seven capabilities, including motif and lineage memory, temporal updates, verifier-noise handling, and recovery (the restoration of falsely penalized clean sources).

Contributions. We make four contributions. *(C1) Framework.* Four trust memories (source, chunk, motif, lineage) built on decayed Beta pseudo-counts with asymmetric aging, bounded propagation, and verifier-aware attenuation. We evaluate each component's contribution separately; those contributions vary across attack schedules and protocols. *(C2) Benchmark and protocol.* A 36-window prequential benchmark with TTD, a defined trust-regret loss, subgroup-FRD, online-learning comparators, and real-corpus overlay protocols in the artifact. *(C3) Conditional theory.* Eight controlled-

Artifact (code, frozen configurations, and evidence records), v1.0.1: https://doi.org/10.5281/zenodo.22169790

*Corresponding author: Xuan-Bach Le (lexuanbach@hcmut.edu.vn).

TABLE I
RAG-DEFENSE FAMILIES × CAPABILITY DIMENSIONS.

Family	Source trust	Chunk trust	Motif	Lineage	Temporal update	Verifier noise	Recovery
Per-query filtering	No	Yes	No	No	No	Partial	No
Robust aggregation	No	Partial	No	No	No	No	No
RA-RAG-style reliability	Yes	No	No	No	Limited	Partial	Limited
Traceback / forensics	Partial	Yes	No	Partial	No	Partial	No
Memory-augmented RAG	Partial	Partial	Partial	No	Partial	No	Limited
Truth discovery	Yes	No	No	Partial	No	No	Limited
LTM-RAG (ours)	✓	✓	✓	✓	✓	✓	✓

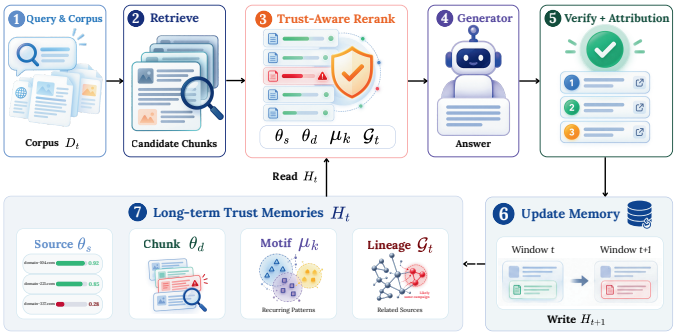

Fig. 1. LTM-RAG prequential loop at window t. Solid arrows are within-window flow; the dashed arrow is the deferred update, which affects retrieval only from window $t+1$.

model results, including a detection-delay separation under observable lineage (Theorem 4) and a hidden-lineage lower bound (Prop. 6), with explicit conditions on what they do not certify. *(C4) Deployment recipe.* The hybrid we recommend in practice, an LTM-RAG gate followed by top-K aggregation. Trust-only LTM-RAG is the longitudinal-mining ablation, and our dense+LLM evaluation logs every judge call with PoisonASR separated from IncorrectASR.

II. Related Work

a) RAG poisoning: A small number of carefully chosen passages, retrieved at inference time, can steer the answer. PoisonedRAG [1] and its successors extend the threat model to more practical corpus-poisoning settings, black-box source observation, trigger-based activation, and joint retriever-generator optimization [2]–[4], [10]. Covert variants hide poison behind sleeper triggers, repeated co-retrieval, or knowledge-graph facts [11], [12]. Across these attacks, poison evidence recurs rather than appearing in a single passage: it exploits ingestion pipelines, source credibility, and recurring motifs—temporal structure that query-local defenses discard.

b) RAG defenses and evaluators: Most existing defenses operate inside one query or one static collection. Filtering and abnormality detection (RAGuard [6]), robust aggregation (RobustRAG [9]), and reliability-aware retrieval (RA-RAG [5]) show that relevance alone is insufficient. Yet none of them maintain source, chunk, motif, *and* lineage evidence across attack and recovery windows. Claim-level evaluators (RAGChecker [13], ARES [14]), self-reflective controllers (Self-RAG [15], IRCoT [16]), and forensic traceback (RAGOrigin [7], RAGForensics [8]) provide complementary feedback. Database provenance also clarifies why forensic traceback is a distinct task from online trust updating [17]. We treat all of these signals as *events* and ask how they should change future retrieval.

c) Memory-augmented and stream-based RAG: Memory-augmented retrieval (HippoRAG [18], [19], GraphRAG [20], MemoRAG [21]) shows that persistent external state improves recall. Its target is answer quality; adversarial recurrence is outside its scope, and a memory that helps recall a recurring source can just as easily amplify recurring poison unless it also tracks negative evidence and recovery. Source-trust estimation predates RAG: truth discovery [22], [23] and knowledge-based trust [24] estimate source reliability. We require trust to be *operational*: the estimate must change top-K retrieval *before* the next round of feedback arrives. This places the framework in the online-prediction setting [25], [26]. Stream mining supplies the evaluation discipline: prequential test-then-train with recurrent drift [27]–[29], change detection [30]–[32], censored time-to-event analysis [33], verifier calibration [34], robust contamination [35], and ranking fairness [36]. We use QA benchmarks (PopQA [37], HotpotQA [38]) to construct controlled overlays; they do not contain naturally occurring poisoning.

Table I summarizes the compared capabilities. Among the compared families, ours is the only one designed to maintain all four trust components with prequential updates and recovery. Cited-system comparisons use paper-guided internal reimplementations of retrieval-time rules, measured on our pipeline rather than copied from published numbers. Head-to-head comparison with the released code is deferred.

III. Longitudinal Trust Mining for RAG

A. Problem Formulation

We model the RAG system as a stream. Corpus updates $\mathcal{U}_{1:T}$ and query batches $\mathcal{Q}_{1:T}$ arrive in discrete windows $t = 1, \ldots, T$. At window t, the index $\mathcal{D}_t$ holds chunks d tagged with source $s(d)$, lineage $\ell(d)$, timestamp $\tau(d)$, and text $x(d)$. For each query we retrieve candidates, generate an answer, and only *later* receive verifier and attribution feedback.

Prequential constraint. At window t we score retrievals using only the state H_t accumulated from *earlier* feedback;

current-window verdicts update future state only (the standard prequential test-then-train protocol [27]). A defender that observes current-window verdicts is essentially our poison-label oracle, reported separately as an upper bound.

Threat model. The attacker may inject, repeat, mirror, mutate, or move a false claim among fresh source identities and may observe public ranking behavior. It cannot alter protected verifier outputs or metadata already committed to the audit log. Per schedule the attacker modifies at most 20% of stream chunks. Domain, URL, textual similarity, and motif signals are useful only when observable; their absence is an explicit failure boundary. Protocol-specific feedback is detailed in Section IV. In the canonical CPU matrix, poison flags and attack-query labels are evaluation-only and do not affect retrieval, generation, verification, or trust updates. Declared lineage IDs are retained for evaluating the inferred graph but are not used to construct it.

Goal. We want to reduce retrieval-level poison exposure and answer-level attack success while preserving clean utility U_t, bounding false-reputation damage FRD_t, and keeping time-to-distrust τ_D and time-to-recovery τ_R small.

Loss and groups. For each compared retrieval/aggregation policy π (Section IV), write $A_t(\pi)$ for retrieval-level poison exposure, $G_t(\pi)$ for answer-level attack success, $C_t(\pi)$ for clean answer loss, and $B_t^g(\pi)$ for false reputation damage on source subgroup g (small vs. large publishers, etc.). The controlled trust-regret loss is

$$L_t(\pi) = w_A A_t(\pi) + w_G G_t(\pi) + w_C C_t(\pi) + \sum_g w_g B_t^g(\pi), \tag{1}$$

with weights $(w_A, w_G, w_C, w_g)=(1, 1, 1, 0.5)$ fixed across all experiments. We keep retrieval exposure separate from answer corruption deliberately. An answer-level defense can succeed while RetExp is still high.

B. Trust Memory Components

The framework maintains four trust memories because campaigns leave different kinds of residue: **source** θ_s (persistent publishers), **chunk** θ_d (mirrored passages), **motif** μ_k (recurring claims/edits under motif-mutation and source-spoofing), and **lineage** $\mathcal{G}_t$ (evidence shared across related identities). First-exposure poison has no longitudinal handle, so it is delegated to the aggregator. We test each memory separately, without presuming uniform effectiveness across schedules; Section V-B reports which effects resolve.

Source trust memory. For each source s, we keep a decayed Beta pseudo-count trust score:

$$\theta_s(t) = \frac{\alpha_s(t)}{\alpha_s(t) + \beta_s(t)}, \quad \alpha_s(t), \beta_s(t) > 0, \tag{2}$$

with updates:

$$\alpha_s(t+1) = \rho_+ \alpha_s(t) + e_s^+(t), \tag{3}$$
$$\beta_s(t+1) = \rho_- \beta_s(t) + e_s^-(t), \tag{4}$$

where $\rho_+ \le \rho_- \in (0, 1]$ are the decay rates applied to positive and negative evidence (defaults in Section III-E) and e_s^+, e_s^- are severity-weighted evidence events. The runner decays at the window boundary after feedback; since $\rho_\pm \le 1$, the bounds below hold with $e_s^\pm$ as delivered mass. Decaying positive mass faster keeps recent failures influential and limits clean-history farming, while the penalty a source can accumulate stays bounded (Theorem 1).

We give new sources neutral priors $\alpha_s(0)=\beta_s(0)=\alpha_0$ with a cold-start penalty cap, and we keep them visible through source-diversity constraints.

Chunk trust memory. A predominantly reliable source may still contain a targeted malicious passage. Chunk-level trust $\theta_d(t)$ runs the same Beta-decay update of Eqs. (3)–(4) at the passage level, so we can suppress a passage without suppressing its source.

Motif memory. Attackers evade source memory by switching identities or mutating wording. The released CPU configuration hashes deterministic lexical and symbolic chunk–query features (trigger phrases, entity–relation edits, numeric changes, URL anomalies, and template n-grams) into a bounded 128-dimensional vector; the separately labeled dense pilot replaces this vector with its named embedding model. A new negative event z with feature vector $f(z)$ joins motif $k^* = \arg\max_k \cos(f(z), \mu_k)$ when $\cos(f(z), \mu_{k^*}) \ge \eta_m$; otherwise we seed a fresh motif. Centroids update via exponential moving average $\mu_k \leftarrow (1-\eta)\mu_k + \eta f(z)$, and support n_k decays each window by ρ_m. Motif risk for candidate d at query q:

$$m(d, q, t) = \sum_k w_k(t) \cdot \sigma\big(\cos(f(d, q), \mu_k) - \eta_m\big), \tag{5}$$

with σ the logistic sigmoid (slope 8) and $w_k(t)$ the recency-weighted motif density discounted by clean counter-evidence. The CPU configuration uses cosine threshold 0.62, EMA rate 0.1, support decay 0.9, and at most 256 clusters. This protocol separation prevents dense-pilot behavior from being attributed to the hashed CPU motif.

Lineage graph and maintenance. Source-spoofing splinters evidence across fresh identities, defeating source-only memory. The released detector uses exact nonempty domain, optional exact URL, and character 5-gram Jaccard at threshold 0.70; author/organization equality is not a standalone signal, WHOIS is unavailable, and token-frequency cosine is disabled in the canonical CPU protocol. Signal weights are source-out normalized; each new chunk scans at most 500 same-domain/recent chunks and each source retains at most five neighbors. Trust then propagates as

$$e_i^-(t) \mathrel{+}= \sum_{j:(j,i)\in E_t} w_{ji} \cdot e_j^-(t) \cdot \mathbf{1}[w_{ji} \ge \delta_\ell], \tag{6}$$

with per-item bound L_{item} and total diffusion bound $L_{\text{total}} = 5.0$ per event (Theorem 1).

The implementation bounds degree and propagation but does not decay or garbage-collect lineage edges. Declared lineage IDs are evaluation metadata and are withheld from the detector.

Algorithm 1 Lineage maintenance at window t.

Require: $\mathcal{G}_t{=}(V_t, E_t)$, threshold δ_ℓ, degree cap D_ℓ
for all new or touched source j in window t **do**
  Compare with at most 500 same-domain/recent chunks
  Compute exact-domain/URL and character-5-gram signals
  Source-out normalize, gate by δ_ℓ, insert/update edges
  Retain top-D_ℓ by weight if $\deg(j){>}D_\ell$
end for

Algorithm 2 Prequential LTM-RAG loop.

Require: Stream windows $\mathcal{U}_{1:T}$, $\mathcal{Q}_{1:T}$, initial trust state H_1
for $t = 1$ to T **do**
  Apply corpus updates $\mathcal{U}_t$ to index $\mathcal{D}_t$
  for query $q \in \mathcal{Q}_t$ **do**
    $C \leftarrow \mathrm{BM25}_K(\mathcal{D}_t, q)$
    Score each $d \in C$ using Eq. (7) and prior state H_t
    $R_q \leftarrow \mathrm{DiverseTopK}(C, k)$ with source cap $\lfloor k/3 \rfloor$
    Generate answer from R_q, log retrieved chunks and attributions
  end for
  $Z_t \leftarrow$ events with availability window $a(z) = t$
  for event $z \in Z_t$ **do**
    Attenuate feedback mass, update source/chunk Beta counts
    Update or create motif; propagate bounded lineage evidence
  end for
  Decay trust/motif mass (ρ_+ positive, ρ_- negative); set H_{t+1}
end for

C. Trust-Aware Retrieval

For candidate chunk d and query q at window t, we score

$$\begin{aligned}\mathrm{Score}(d,q,t) = {} & \lambda_r R(d,q) \\ & + \lambda_s \log(\epsilon + \theta_{s(d)}(t)) \\ & + \lambda_c \log(\epsilon + \theta_d(t)) \\ & - \lambda_m m(d,q,t) \\ & + \lambda_f F(d,q,t) + \lambda_v V(d,q,t),\end{aligned} \tag{7}$$

where $R(d,q)$ is BM25-style lexical relevance [39], $F(d,q,t) = \exp(-\mathrm{age}(d)/10)$ a freshness bonus in windows, and $V(d,q,t) = 1/(1+\sum_j w_{ji})$ the lineage-diversity reward over effective neighbor weights (fallback $1/\mathrm{source\ count}$ in the candidate pool), which discourages densely linked candidates; monoculture is prevented by the source cap below, when enough distinct sources survive the gate. We fix $\{\lambda_r, \lambda_s, \lambda_c, \lambda_m, \lambda_f, \lambda_v\}{=}\{1.0, 0.4, 0.3, 0.5, 0.1, 0.2\}$ globally, before any attack schedule or dataset. Baselines use the same candidate pools and seed budget, and no method gets per-attack tuning.

At retrieval time we pull a fixed-width BM25 candidate set and gate candidates whose source or chunk trust score is below $\tau_{\text{gate}} = 0.30$. The survivors are scored by Eq. (7) and clamped from below as $\max\{\text{score}, \alpha R(d,q)\}$, $\alpha{=}0.85$ (active on every canonical query), then returned through a source-diverse top-K rule with per-source cap $\lfloor K/3 \rfloor$ (Algs. 1, 2). An escalation queue is available as a deployment option; the reported CPU path does not use it.

D. Trust Updates and Verifier Calibration

After generation, verifier and attribution modules produce event tuples $z = (q, d, s, y, p_v, \gamma, t_a)$. Here $y \in \{\mathrm{pos}, \mathrm{neg}\}$ is the answer verdict, γ the severity weight, $p_v \in [0,1]$ the correctness confidence (rule verifier: token-overlap F1 against the reference, with y set by $\mathrm{EM}{>}0$ or $\mathrm{F1}{\geq}0.5$), and t_a the availability window of the feedback (origin window $o(z) \leq t_a$). For source s (and analogously a chunk),

$$e_s^+(t) = \sum_{z:s(z)=s} \max(0.1, p_v)\gamma\kappa_v(t)\mathbf{1}[y = \mathrm{pos}], \tag{8}$$

$$e_s^-(t) = \sum_{z:s(z)=s} \max(0.1, 1-p_v)\gamma\kappa_v(t)\mathbf{1}[y = \mathrm{neg}]. \tag{9}$$

The rule verifier emits verified-clean ($\gamma = 1$) or answer-error ($\gamma = 0.7$); evaluation-only attack flags never select severity.

Verifier attenuation. Over the latest 200 adjudicated predictions we estimate the expected calibration error (ECE) with 15 equal-width bins and attenuate update mass by

$$\kappa_v(t) = 1 - 0.5\min\{1, \mathrm{ECE}_t\,/0.20\} \in [0.5, 1].$$

This CPU mechanism is not Platt scaling, temperature scaling, rollback, or a last-known-good rebuild. Dense/LLM-judge results use their separately logged judge protocol and do not imply those unimplemented CPU mechanisms.

Trajectory monitoring. Beyond the asymmetric aging of Eqs. (3)–(4), trajectory monitoring multiplies the update mass by a bounded $\mu_{\text{traj}}{>}1$ when a source above a high-reputation threshold sees a fresh attributed negative event (capped to preserve Theorem 1).

Update order. Per attributed event we attenuate the weight, update source and chunk Beta counts, assign or create a motif, propagate bounded evidence across eligible lineage edges, decay stale trust/motif mass, and log the update; feedback from window t touches only windows $t{+}1, t{+}2, \ldots$

E. Formal Guarantees

We state conditional controlled-model results to explain the mechanism under bounded evidence, temporal decay, and observable recurrence. They do not certify deployment. Theorems 1 and 4 provide conditional event-level characterizations; they do not imply a deterministic window-level detection delay. Props. 2 and 3 give regret and recovery envelopes. Defaults are $\rho_- = 0.92$, $\rho_+ = 0.88$, per-window direct bound $E_{\max} = 1.0$, per-window propagated bound $L_{\text{item}} = 0.35$, source-level threshold $\Delta = 1.5$, component threshold $\Delta_c = 0.5$, and motif threshold $\eta_m = 0.62$.

Theorem 1 (Bounded memory). *If total direct and propagated mass delivered to item i in one window are bounded by $E_{\max}$ and L_{item}, then for every window t, with $\rho = \max\{\rho_+, \rho_-\}$,*

$$\alpha_i(t), \beta_i(t) \;\leq\; \rho^t \max\{\alpha_i(0), \beta_i(0)\} \;+\; \frac{E_{\max} + L_{\text{item}}}{1-\rho}. \tag{10}$$

Proof sketch. Unroll the recurrence with the stated per-window total-mass bounds. The geometric sum converges to $(E_{\max} + L_{\text{item}})/(1-\rho)$; the same argument applies to α_i.

Proposition 2 (Trust-regret bound). *For a campaign c active over T_c windows with detection bound N_c (eligible harmful-feedback events) from Theorem 4, let $W_c(N_c)$ be the number of active windows elapsed through the N_c-th eligible event. If the per-window excess controlled loss is bounded by $L^{\ell}_{\max}$*

and the post-detection residual by R_c (an assumed envelope covering the post-detection trust floor and aggregator leakage, which we do not derive here, so the bound is conditional on $R_c < \infty$),

$$\mathrm{Reg}_c \;\le\; L^{\ell}_{\max}\,\min\{T_c, W_c(N_c)\} \;+\; R_c. \tag{11}$$

Proof sketch. Regret accumulates only before detection: at most $\min\{T_c, W_c(N_c)\}$ windows elapse before the N_c-th eligible event, each contributing at most $L^{\ell}_{\max}$ to the controlled loss. After that event, residual exposure is bounded by R_c (post-detection trust floor plus aggregator leakage). Summing the two phases yields the bound.

Proposition 3 (Finite recovery). *A clean source i falsely penalized at t_0 to Beta state $(\alpha_i(t_0), \beta_i(t_0))$ and then receiving clean evidence of mass $a{>}0$ per window for r windows (decay $\rho \in (0,1)$; an audit channel the gate alone may starve, Section VII) recovers to $\theta_s = \alpha/(\alpha{+}\beta) \ge \tau$ ($0{<}\tau{<}1$) once accumulated clean mass dominates the decayed penalty residual:*

$$a \cdot \tfrac{1-\rho^r}{1-\rho} \;\ge\; \tfrac{\tau}{1-\tau}\,\rho^r\,\beta_i(t_0) \;-\; \rho^r\,\alpha_i(t_0). \tag{12}$$

Proof sketch. After r windows of clean evidence $a > 0$, accumulated clean mass is $a(1-\rho^r)/(1-\rho)$ while the penalty residual decays to $\rho^r\beta_i(t_0)$. Recovery $\theta_s \ge \tau$ requires $(1-\tau)\alpha_i(t_0{+}r) \ge \tau\beta_i(t_0{+}r)$; substituting the decayed and accumulated terms gives the stated inequality. Since $\rho < 1$, the LHS grows and the RHS shrinks, so a finite r always exists.

Theorem 4 (Event-level detection-delay separation). *A spoofing campaign places n harmful events across m fresh identities, with at most h events per identity. Each event contributes b direct and g propagated motif/lineage evidence to the* component score $C_c(t) = \sum_{j\le t}\rho^{t-j}(b{+}g)$*, the decayed evidence mass on the lineage component spanning the campaign's identities, flagged once $C_c(t) \ge \Delta_c$; assume eligible harmful-feedback events arrive at consecutive update steps. Let*

$$N(x,\Delta) = \left\lceil \log\!\left(1 - \tfrac{\Delta(1-\rho)}{x}\right) \Big/ \log\rho \right\rceil. \tag{13}$$

If $b{+}g > \Delta_c(1{-}\rho)$ (the regime in which N is defined and $N \ge 1$) and the campaign emits at least $N(b{+}g, \Delta_c)$ eligible events, then LTM-RAG *flags it at the $N(b{+}g,\Delta_c)$-th such event, whereas source-only does not flag inside the horizon whenever $b\,(1{-}\rho^h)/(1{-}\rho) < \Delta$.*

Proof sketch. Unrolling C_c over k consecutive events gives $(b{+}g)(1{-}\rho^k)/(1{-}\rho)$, which reaches Δ_c exactly at $k{=}N(b{+}g,\Delta_c)$; campaigns emitting fewer events are outside the statement. Source-only per-identity mass $b(1{-}\rho^h)/(1{-}\rho)$ stays below Δ once split across m identities. At defaults ($\rho{=}0.92$, $b{=}0.30$, $g{=}0.18$, $h{=}2$) source-only mass is $0.576{<}1.5$ (Prop. 6 regime), and $N(0.48, 0.5){=}2$ events.

The bound is indexed by harmful feedback events rather than windows. Converting it to a window-level delay additionally depends on batch size, attack scheduling, attribution eligibility, and feedback delay. We therefore report empirical TTD with censor counts and do not claim a deterministic event-to-window conversion.

Supporting conditional observations. **Prop. 5** (calibration–detection): multiplying at most $\varepsilon_v(1+\chi)$ erroneous mass by $\kappa_{\max}$ gives harmful mass $\le \kappa_{\max}\varepsilon_v(1+\chi)\bar{w}\gamma$; useful signal crosses the threshold when $\kappa_{\min}(b+g) > \Delta_c(1-\rho)$. **Prop. 6** (hidden lineage): substituting $g = 0$ removes cross-identity evidence and recovers source-only cold start. **Prop. 7** (aging): applying the two recurrences for r empty windows leaves positive and negative mass bounded by $\rho_+^r A_i(t_0)$ and $\rho_-^r B_i(t_0)$. **Prop. 8** (complexity): with B queries per window, counting candidate ranking, bounded neighbor propagation, and motif comparisons yields $O\big(B(K\log N + M_q d_f) + |E_v|(1+D_\ell)\big)$ per window. Props. 5–8 use: ε_v, verifier false-label bound; χ, error correlation; $\bar{w}$, mean event weight; A_i, B_i, item i's positive/negative mass; $|E_v|$, touched lineage edges/window; B, queries/window; M_q, motif comparisons/query; d_f, feature dimension; $\pi^\star$, per-window optimal comparator for $\mathrm{Reg}_c, \mathrm{Reg}_T$.

Scope. These results explain controlled-stream behavior under the stated assumptions. They are *not* certificates of deployment-scale robustness, sublinear regret against arbitrary adaptive attackers, or protection against hidden lineage, verifier manipulation, calibration-set poisoning, or every flavor of clean-history farming.

IV. Experimental Setup

A. Research Questions

We organize the experiments around four questions. **RQ1**: Does longitudinal trust memory cut retrieval-level poison exposure and answer-level attack success across diverse poisoning schedules without degrading clean utility? **RQ2**: Which tested components and memory contrasts (source posterior, motif, lineage, decay) show statistically resolved effects (95% CI excludes zero), and under which attack families? **RQ3**: How does LTM-RAG behave when the attacker spoofs sources or adapts to the defense? **RQ4**: What is the overhead, the impact on clean utility, and under which conditions do the assumptions become limiting?

B. Controlled Stream Benchmark

Each stream has horizon $T{=}36$ windows, batch size $B{=}20$, attack rate p_a, motif recurrence p_m, and clean drift p_c. We designed five schedules: **Burst** ($p_a{=}0.3$, 5 consecutive windows) tests rapid detection; **Slow** ($p_a{=}0.05$ uniform) tests weak-signal accumulation; **Mirror** ($p_a{=}0.1$, $p_m{=}0.8$, 3 domains) tests lineage; **Motif-mutation** preserves motif but mutates source/wording every 5 windows; **Source-spoofing** ($p_a{=}0.15$) rotates identities every 3 windows.

Each chunk carries source ID, source type, timestamp, lineage ID, near-duplicate cluster, and author fields. We use 5 seeds across the 5 schedules (25 schedule-seed runs) and report mean $\pm$ 95% CI for the five-seed comparisons; single-run diagnostics appear as point estimates. Every method retrieves and scores using only prior trust state at each window, and verifier feedback is applied only after the retrieval it evaluates (the prequential protocol [27]). Source churn is 10% per window. Lineage observability is controlled by $p_{\text{obs}} \in [0,1]$

(canonical runs use $p_{\text{obs}} = 1$), and clean drift updates 5% of clean chunks per window.

Data sources. Our evaluation spans three layers. A controlled layer isolates the recurring-campaign signal from natural-text variance. A real-corpus layer is grounded in established open-domain QA benchmarks. A protocol-separated live-judge diagnostic adds a live LLM judge. Each layer carries a provenance tag in its caption.

a) Verifier signal and weight selection (methodological): In the canonical CPU matrix, feedback is derived from post-generation answer correctness, and per-chunk poison labels do not affect retrieval or trust updates. In the separately reported fixed-gate recurrence stacks, known attack membership supplies delayed negative-feedback attribution after the current query; the protocols are reported separately. We pick the fixed scoring weights $(\lambda_r, \lambda_s, \lambda_c, \lambda_m, \lambda_f, \lambda_v)$ on a held-out clean-stream calibration set; a 50-sample Latin-hypercube perturbation [40] in $[0.5\times, 1.5\times]^6$ preserves the LTM-RAG vs. baseline ranking on every sample (Kendall $\tau \geq 0.9$ [41]).

C. Real-Corpus-Derived Layer

PopQA. Single-fact tuples from official PopQA [37], 500 queries, 36-window stream. **HotpotQA.** Multi-hop questions from the dev-distractor split [38], 500 queries, supporting contexts plus one deterministic distractor each. Gold answer text supplies post-generation correctness feedback; poison flags remain evaluation-only. Given HotpotQA's documented limitations [42], this layer serves as a controlled multi-hop overlay that audits pipeline behavior rather than deployment-scale robustness; naturally occurring adversarial-edit corpora and per-overlay aggregates are deferred to the artifact and future work.

D. Stress Envelope

Consistent with Section III-A, the attacker may modify up to 20% of stream chunks, create fresh identities, mirror content, mutate motifs, and time attacks across windows; we separately stress hidden lineage, correlated verifier errors, clean-history farming, and their combinations. The released CPU path attenuates evidence using rolling ECE at threshold 0.20 and implements no rollback or Platt calibration; white-box joint retriever–generator optimization and direct prompt injection of the verifier remain outside this study.

E. Baselines

We compare against nine non-oracle controls reported in the paper tables, one artifact-only diagnostic, and one oracle upper bound, organized into four tiers. **Tier 1 (no memory):** Vanilla RAG. **Tier 2 (online source and chunk reliability):** RA-RAG-style Reliability [5], Source-only Trust, and Chunk Quarantine (artifact-only diagnostic). The first two maintain online Beta-Bernoulli reliability estimates updated from the same verifier feedback. **Tier 3 (structural and aggregation):** RobustRAG-style Aggregate [9] and RAGForensics-style Traceback [8], evaluated with a live Gemini generator [43], plus the Query-local Majority aggregation control. **Tier 4 (streaming-learning comparators):** ADWIN-online [31], FTRL-windowed [44], and Hoeffding-drift [30], [45]. **Oracle Memory** uses poison labels only after current-window evaluation as an upper bound. The cited-system rows are paper-guided internal reimplementations, not executions of official repositories; all receive the same candidate pool and feedback, and released-code comparison is deferred.

Implementation details. **Vanilla** is BM25 top-K=10 retrieval with no trust signal and no aggregation. **Source-only Trust** runs only the source posterior of Eqs. (3, 4) with decay ρ_-=0.92 and the 0.30 trust gate, scoring with the relevance and source-trust terms only. **Chunk Quarantine** maintains chunk posteriors and drops chunks below 0.30. **ADWIN-online**, **FTRL-windowed**, and **Hoeffding-drift** carry per-source state; these simulator-level mechanism controls are not part of the released canonical artifact, so we report them only as the controlled contrast for state richness in Table V. **Query-local Majority** gives each distinct source at most one vote. **Hybrid** applies the LTM-RAG gate followed by that same vote, allowing a direct memory-with-versus-without control.

F. Metrics

Security metrics. *Cum. ASR* is *retrieval-level*, the cumulative fraction of attack opportunities at which poison reaches the served top-K (equivalent to retrieval exposure, RetExp, on the controlled stream). *AnsASR* is *answer-level*: the frozen CPU matrix requires the attacker's target answer, whereas the dense+LLM-judge stack reports any incorrect non-abstaining answer and additionally logs PoisonASR in the artifact. We never pool these protocol-specific rates. PoisonASR and IncorrectASR split the dense/LLM failure modes. **Utility.** Clean utility is the mean token-level F1 between generated and gold answers on clean (non-attack) queries, reported as the Clean F1 column. Recall@10 and nDCG@10 measure retrieval quality on clean queries and AnswerF1 is token-F1 over all queries; these per-run values are released in the artifact. These utility metrics are reported individually; we do not average them into a single score. **Temporal.** Trust Regret $\text{Reg}_T = \sum_t L_t(\pi) - \sum_t L_t(\pi^\star)$, TTD (time-to-distrust), TTR (time-to-recovery, censored at T=36), FRD (false reputation damage), with subgroup decomposition. **Cost.** ms/window, peak MB, ECE and Brier, noise sensitivity, lineage precision and recall, bounded-candidate scalability stress.

G. Implementation

BM25-style lexical retrieval [39], extractive answer generation, deterministic rule-based verifier. The CPU path is reproducible without an API key. Defaults are ρ_-=0.92, ρ_+=0.88, η_m=0.62, character 5-gram lineage threshold 0.70, α_0=β_0=1.0, τ_{gate}=0.30, and a trajectory multiplier μ_{traj}=1.25 applied above reputation 0.80. Scoring weights are globally fixed at $(1.0, 0.4, 0.3, 0.5, 0.1, 0.2)$. Dense retrieval uses BGE-large-en-v1.5 [46]; generation, gemini-2.5-flash-lite; the judge, gpt-4o-mini at temperature 0 [47]; strings, prompts, and dates are in the artifact. The controlled streams contain 720 queries per schedule–seed run; the PopQA/HotpotQA overlays use

a separate 500-query convention and are never pooled with the controlled matrix. Frozen configurations, seeds, aggregate outputs, and per-query/window records are included in the artifact. Dense/LLM results include logged outputs but require the named external models to rerun exactly.

V. Results

The evaluation spans a fixed-gate BM25 stream and a BGE+Gemini stream with a live judge. Table II reports them separately while making baseline provenance explicit. On the five-seed Burst protocol, Hybrid reduces AnsASR from $94.8 \pm 4.9\%$ to $14.0 \pm 5.0\%$ for BM25 and from $92.2 \pm 4.8\%$ to $33.6 \pm 8.2\%$ for BGE+Gemini.

This is a separately logged diagnostic, not an end-to-end reproduction. Counts above 1,200 retain historical reruns (artifact report), so rows are within-protocol rates, not paired. Hybrid records $0/1200$ events: a rule-of-three one-sided 95% bound of 0.25%.

To isolate the incremental contribution of longitudinal state under one frozen CPU protocol, we additionally compare each memory-based method with a query-local control using identical candidate pools, schedules, and seeds.

A. RQ1: Separating Aggregation and Memory

Burst-CPU (Table IV) and Burst-KA (Table II) are separate protocols. Table IV shows Vanilla vulnerable across all five schedules ($84.9 \pm 1.3\%$ overall AnsASR). Query-local voting produces the largest single-mechanism reduction, to $22.6 \pm 1.9\%$, while also raising clean answer F1 from 62.5 to 79.5%. Single-answer LTM-RAG reaches $77.8 \pm 3.4\%$: better than Vanilla but insufficient as a first-exposure defense.

The Hybrid reaches $19.2 \pm 2.4\%$. The direct paired comparison is Hybrid minus Query-local Majority, because these methods share the answer rule; it is our primary comparison, and the remaining intervals are exploratory, without multiplicity correction. Overall AnsASR changes by -3.42 percentage points (95% CI $[-5.54, -1.30]$), whereas clean F1 changes by -0.03 points ($[-0.20, 0.14]$). The longitudinal increment is significant overall and in Burst (-6.84 points, $[-12.26, -1.42]$); schedule-specific intervals for Slow, Mirror, Motif-mutation, and Source-spoofing cross zero. Thus the data support a modest complementary benefit overall; they do not support the claim that memory explains the entire Vanilla-to-Hybrid gap.

Against Source-only, full LTM-RAG improves overall AnsASR by 2.32 points (paired CI $[0.42, 4.23]$ in favor of LTM) and clean F1 by 1.35 points ($[0.51, 2.18]$). Its retrieval exposure falls from 94.5 to 91.8%. These effects isolate the incremental value of chunk/motif/lineage state beyond a decayed source posterior.

Table V asks a complementary question: whether carrying richer campaign state improves over per-source streaming learners in the controlled mechanism diagnostic. Table VI then exposes the regime split directly. Aggregation acts at first exposure and memory at re-exposure; Hybrid applies both. The Table VI RobustRAG row uses the simulator's extractive answer rule, so it is not comparable to Table II. Oracle Memory sees labels only from evaluated windows, so at first exposure it holds none and does not upper-bound the others. Both are explanatory diagnostics.

Per-window curves and eligible-query counts are in the artifact; Hybrid separates from its query-local control under repeated Burst and Mirror but not under Source-spoofing.

B. RQ2: Component Effects

Removing lineage lowers FRD because fewer negative updates propagate, at the cost of a 2.94-point overall AnsASR increase (95% CI $[1.16, 4.72]$). Removing decay worsens AnsASR by 4.61 points ($[3.18, 6.04]$) and clean F1 by 1.22 points ($[0.28, 2.17]$). In contrast, removing motif changes AnsASR by -0.63 points ($[-1.39, 0.12]$) and removing ECE attenuation by 0.15 points ($[-0.10, 0.40]$); neither interval excludes zero, so no measured efficacy gain is assigned to those components under this benchmark.

C. RQ3–RQ4: Stress, Temporal, Lineage, and Cost Diagnostics

On Burst, all ten poison sources are horizon-censored for Vanilla, whereas full LTM has a mean TTD over uncensored sources of 14.3 ± 1.3 windows with no censored source. For the broader schedules, censoring is substantial (e.g., 161 source-events for LTM on Mirror), so a mean without its censor count would be misleading. The released per-source records retain those counts; we do not use TTR as a headline comparison.

The lineage detector was audited on mirrored seeds 1–4 after fixing the threshold on seed 0: precision/recall against declared cross-source lineage IDs range 0.098–0.148 and 0.333–0.478, while an evaluation-only neighbor-of-poison diagnostic reaches 0.598–0.708. The gap shows that the available lineage annotations are incomplete and the detector remains noisy. At this precision the resolved ablation may partly reflect broad distrust propagation rather than correct lineage recovery, which is also consistent with the higher FRD of LTM over Source-only (Section VII); we report the lineage effect as protocol-specific and do not claim verified lineage attribution.

Measured inside each run on the Apple M4, mean Burst cost is 133.4 ± 13.2 ms/window and 5.2 MB for Vanilla versus 851.4 ± 7.9 ms/window and 48.8 MB for LTM. Hybrid is similar to LTM (858.9 ± 49.4 ms/window, 50.0 MB). These figures come from a separate single-process, five-seed profile and include the Python in-memory index, lineage ingestion, evaluation, and tracing; they are not deployment-scale ANN or LLM latency measurements.

VI. Analysis

a) Mechanisms behind longitudinal memory: Source-only memory resets on a fresh identity; observable lineage carries bounded evidence across that reset, motif abstracts repeated patterns, and decay bounds retained penalty. Their measured importance is protocol-specific: the controlled-simulator stress diagnostic shows larger motif/lineage effects, whereas the canonical CPU matrix supports lineage and decay but finds no resolved motif effect.

TABLE II

PROTOCOL-SEPARATED RECURRENCE STACKS (ANSASR %, MEAN ± 95% CI) UNDER THE DELAYED KNOWN-ATTRIBUTION SETTING OF SECTION IV; THE TWO COLUMNS ARE SEPARATE PROTOCOLS AND ARE NOT POOLED. THE RAGFORENSICS-STYLE ROW USES A GEMINI PER-CHUNK FILTER WITH KEYWORD VOTE IN BOTH COLUMNS, THE OTHER ROWS EACH COLUMN'S OWN STACK. [b] RATE OVER SERVED QUERIES: IT SERVES $20.0 \pm 39.2\%$ (BM25) AND $57.6 \pm 23.4\%$; THE IDENTICAL $11.1 \pm 21.8\%$ IS COINCIDENTAL, TWO SMALL SERVED SUBSETS SHARING ONE PER-SEED VALUE ($\approx 1/9$).

Method	**BM25 + fixed gate** Burst-KA, 5 seeds	**BGE + Gemini + gpt-4o-mini judge** Burst-KA, 5 seeds
Vanilla	94.8 ± 4.9	92.2 ± 4.8
RobustRAG [9] (per-chunk Gemini + kw vote)	93.9 ± 5.0	83.7 ± 9.6
RA-RAG [5] (3-paraphrase consistency)	96.3 ± 4.0	87.0 ± 7.4
RAGForensics-style [8] (θ=2; partial-served)[b]	11.1 ± 21.8 (20% srv)	11.1 ± 21.8 (58% srv)
LTM-RAG	$\mathbf{26.1 \pm 8.8}$	$\mathbf{38.1 \pm 8.3}$
Hybrid (LTM + Aggregate)	$\mathbf{14.0 \pm 5.0}$	$\mathbf{33.6 \pm 8.2}$

TABLE III

CONTROLLED CANDIDATE/ANSWER DIAGNOSTIC WITH A LIVE LLM JUDGE (BURST, FIVE SEEDS; MODELS IN SECTION IV-G). $\text{ANSASR}_{\text{JUDGE}}$ IS THE JUDGE'S INCORRECT NON-ABSTAINING RATE AND $\text{CLEAN}_{\text{JUDGE}}$ ITS WITHIN-PROTOCOL COMPLEMENT, BOTH DISTINCT FROM THE TARGET-ANSWER ANSASR (TABLE II) AND TOKEN-LEVEL CLEAN F1 (TABLES IV, VII). ROW DENOMINATORS DIFFER (SEE TEXT).

Method	**n judged**	**$\text{AnsASR}_{\text{judge}}$ (%) ↓**	**$\text{Clean}_{\text{judge}}$ (%) ↑**
Vanilla RAG	1211	6.61	93.39
Source-only Trust	1200	5.92	94.08
RA-RAG-style Reliability	1210	3.06	96.94
LTM-RAG	1200	0.75	99.25
Hybrid	1200	0.00	100.00

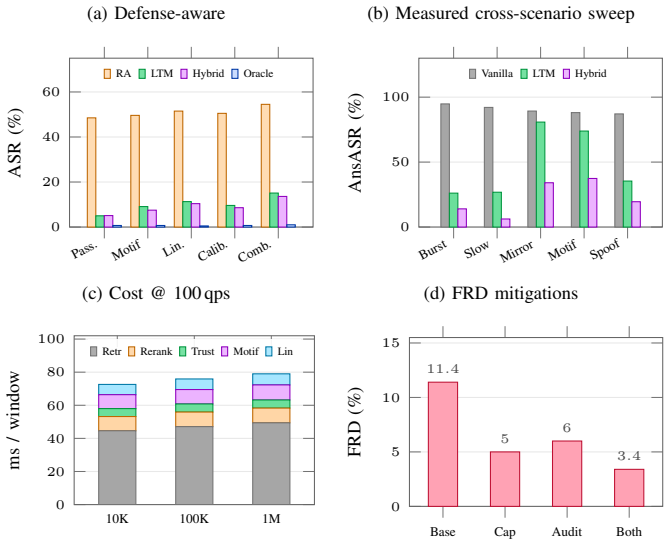

Fig. 2. Diagnostics (provenance per panel): (a) controlled-simulator, (b) measured, (c) profiled, (d) historical/projection. In (a): RA, RA-RAG-style; axis labels denote passive/motif/lineage/calibration/combined attacks. (c) per-operation profile at 100 queries/s (Apple M4), distinct from the per-window wall-clock in Section V-C.

b) Failure modes and mitigations: First contact is handled by Hybrid aggregation; memory contributes only after repeated exposure. Hidden lineage collapses the transferable advantage to source-only behavior, and the dense stack in Table II is a portability check for lexical-stack limits. Verifier dependence remains a boundary: rolling ECE attenuates update mass but does not correct correlated or adversarial feedback, so the defense-aware diagnostics describe stress behavior; they do not certify a robustness envelope.

c) Summary and Diagnostics: In the canonical CPU protocol, Hybrid improves over Vanilla and its query-local control; the paired comparison isolates the longitudinal increment. Figure 2 reports protocol-separated stress, cross-scenario, cost, and FRD diagnostics; profiled/projected panels are not headline efficacy results, and schedule-specific intervals are not extrapolated across protocols.

VII. LIMITATIONS AND BROADER IMPACT

Evaluation scope. Evidence spans controlled streams, PopQA/HotpotQA overlay runs released in the artifact, and separately logged dense/LLM runs. These layers are not pooled. Their corpus sizes, templated attack construction, and model/API dependence still limit external validity. We do not claim deployment-scale robustness, multi-hop generalization beyond the reported overlay, or superiority to official third-party implementations; those require released-code head-to-head runs on larger natural corpora and on naturally occurring adversarial-edit corpora.

Feedback and attribution. The rule verifier compares generated answers with gold answers after generation; a production system may receive no gold feedback, delayed human judgments, or manipulated signals. Evidence is attributed only to chunks supporting the generated answer, and that attribution can still be wrong. The canonical CPU matrix excludes poison labels and attack-query flags from policy decisions, and its inferred graph does not use declared lineage IDs. The fixed-gate recurrence stacks instead assume known delayed attribution; evaluating learned attribution under operational feedback remains future work.

First exposure and adaptive attackers. An unseen identity starts at a neutral prior, so memory cannot prevent its first successful poisoning attempt. The Hybrid relies on diversity among sources; coordinated Sybil sources that dominate the vote, white-box optimization against extraction, or simultaneous loss of motif, lineage, and verifier signals can defeat it. Source-spoofing results show that the memory increment can vanish even when query-local voting remains effective.

Fairness and governance. Propagated distrust can damage clean, small, or new sources. Overall FRD is 5.6% for LTM versus 3.6% for Source-only and 0.9% for the Hybrid

TABLE IV

ANSWER-LEVEL ATTACK SUCCESS RATE (ANSASR, %, MEAN ± 95% STUDENT-t CI; 5 SEEDS). OVERALL POOLS ATTACK OPPORTUNITIES ACROSS SCHEDULES WITHIN EACH SEED (MICRO-AVERAGE, NOT AN UNWEIGHTED SCHEDULE MEAN). LOWER IS BETTER. MOTIF-MUT.: MOTIF MUTATION; SRC.-SPOOF: SOURCE-SPOOFING.

Method	Burst-CPU	Slow	Mirror	Motif-mut.	Src.-spoof	Overall	Clean F1
Vanilla	87.2±6.0	86.5±5.1	82.6±3.1	84.5±3.7	86.1±5.2	84.9±1.3	62.5±1.6
Query-local Majority	43.1±11.9	33.6±15.6	28.4±6.9	13.1±5.9	12.6±1.0	22.6±1.9	79.5±0.7
Source-only	87.2±5.9	63.1±13.2	81.0±5.3	78.4±8.5	86.1±5.2	80.1±3.5	67.5±1.7
LTM-RAG	86.7±6.4	55.9±15.2	78.6±3.8	75.3±6.1	86.1±5.5	77.8±3.4	68.9±1.6
Hybrid	36.3±14.1	20.1±12.0	25.2±3.9	13.1±6.4	12.0±0.4	19.2±2.4	79.5±0.6

TABLE V

STREAMING-LEARNING CONTROLS, 36-WINDOW PREQUENTIAL CONTROLLED STREAM (EXPLANATORY DIAGNOSTIC; SEE TEXT). TTD: TIME-TO-DISTRUST IN WINDOWS, IMPUTED AT THE HORIZON T=36 WHEN DISTRUST NEVER OCCURS. CLEAN %: CLEAN-ANSWER RATE.

Method	RetExp %	Clean %	TTD
Hybrid	4.9 ± 0.5	74.0	12.9
LTM-RAG	6.0 ± 0.5	73.7	12.5
Hoeffding-drift	24.1 ± 1.3	73.2	19.2
FTRL windowed	26.7 ± 1.3	72.4	19.3
ADWIN online	30.5 ± 1.4	73.1	21.5
Source-only	42.6 ± 1.0	71.2	22.5

TABLE VI

FIRST- VERSUS RECURRING-EXPOSURE DECOMPOSITION (CONTROLLED MECHANISM DIAGNOSTIC; ROBUSTRAG IS AN INTERNAL REIMPLEMENTATION). FIRST EXPOSURE IS A SOURCE IDENTITY'S FIRST ATTACK QUERY (ABOUT TWO QUERIES PER RUN; INDICATIVE ONLY). ORACLE MEMORY FILTERS SOURCES WITH POST-WINDOW LABELS FROM EARLIER WINDOWS.

Method	**First exposure**		**Recurring exposure**	
	RetExp ↓	**AnsASR ↓**	**RetExp ↓**	**AnsASR ↓**
Vanilla	60.4	49.3	59.8	49.1
RobustRAG	60.2	2.6	60.0	2.6
LTM-RAG	51.5	42.2	2.9	2.0
Hybrid	**51.3**	**1.7**	**2.8**	**0.5**
Oracle	4.0	2.5	2.7	1.6

TABLE VII

OVERALL CANONICAL RESULTS AND ABLATIONS (%, MEAN ± 95% CI).

Variant	AnsASR ↓	Clean F1 ↑	FRD ↓
Source-only	80.1±3.5	67.5±1.7	3.6±0.3
LTM-RAG	77.8±3.4	68.9±1.6	5.6±0.3
−Motif	77.2±3.7	68.9±1.5	5.6±0.4
−Lineage	80.7±2.6	69.0±1.2	3.5±0.2
−Decay	82.4±3.2	67.6±1.2	1.3±0.1
−ECE attenuation	77.9±3.5	68.9±1.6	5.6±0.3
Hybrid	19.2±2.4	79.5±0.6	0.9±0.2

(Table VII); these values are benchmark-specific and do not establish demographic fairness. Practical use needs human-readable evidence chains, subgroup audits, appeal and rehabilitation paths, retention limits, and conservative thresholds. The current gate can also starve a distrusted source of positive recovery evidence; an independent audit/exploration channel is needed in production.

Cost. The Python implementation scans up to 500 prior chunks for each lineage insertion and is roughly 6.4 times slower than Vanilla in our Burst profile. Candidate caps bound work but do not demonstrate end-to-end latency at million-document scale. Approximate-nearest-neighbor retrieval, provenance indexing, and LLM generation/judging need separate benchmarking.

The artifact releases synthetic poison material for defensive research. It should not be used to target live retrieval systems, and any automated distrust decision should remain reviewable by affected content providers.

VIII. CONCLUSION

Poisoning campaigns distribute related evidence across queries and sources. LTM-RAG addresses that structure with four trust memories (source, chunk, motif, lineage), updated prequentially from verifier and attribution feedback so that recurring evidence accumulates across queries.

Longitudinal mining handles recurrence and query-local aggregation handles first exposure; neither suffices alone, so we recommend their hybrid. On the BM25 fixed-gate recurrence stack with delayed known attribution, the measured Burst AnsASR falls from 94.8% to 14.0%, while paper-guided per-query controls remain vulnerable to the recurring attacker (Table II). These results hold under protected verifier feedback and observable lineage and motif structure; an attacker who breaks both bypasses the gate. Head-to-head comparison against official implementations remains future work.

ACKNOWLEDGMENT

We acknowledge Ho Chi Minh City University of Technology (HCMUT), VNUHCM for supporting this study.

REFERENCES

[1] W. Zou, R. Geng, B. Wang, and J. Jia, "PoisonedRAG: Knowledge corruption attacks to Retrieval-Augmented generation of large language models," in *USENIX Security*. USENIX, Aug. 2025, pp. 3827–3844.

[2] B. Zhang, Y. Chen, Z. Liu, L. Nie, T. Li, Z. Liu, and M. Fang, "Practical poisoning attacks against retrieval-augmented generation," in *Proc. ACM SACMAT*. ACM, 2026, pp. 33–44.

[3] C. Choi, J. Kim, S. Cho, S. Jeong, and B. Chang, "The RAG paradox: A black-box attack exploiting unintentional vulnerabilities in retrieval-augmented generation systems," in *Findings of ACL: EMNLP*. ACL, Nov. 2025, pp. 23 723–23 744.

[4] H. Wang, R. Zhang, J. Wang, M. Li, Y. Huang, D. Wang, and Q. Wang, "Joint-GCG: Unified gradient-based poisoning attacks on retrieval-augmented generation systems," in *Proc. AAAI Conf. Artif. Intell.*, vol. 40, no. 42, 2026, pp. 35 793–35 801.

[5] J. Hwang, J. Park, H. Park, D. Kim, S. Park, and J. Ok, "Retrieval-augmented generation with estimation of source reliability," in *Proc. EMNLP*. ACL, Nov. 2025, pp. 34 279–34 303.

[6] Z. Cheng, J. Sun, A. Gao, Y. Quan, Z. Liu, X. Hu, and M. Fang, "Secure retrieval-augmented generation against poisoning attacks," in *Proc. IEEE Int. Conf. Big Data*. IEEE, 2025, pp. 1799–1806.

[7] B. Zhang, H. Xin, Y. Chen, Z. Liu, B. Yi, T. Li, L. Nie, Z. Liu, and M. Fang, "Who Taught the Lie? responsibility attribution for poisoned knowledge in retrieval-augmented generation," in *2026 IEEE Symposium on Security and Privacy (SP)*. Los Alamitos, CA, USA: IEEE Computer Society, May 2026, pp. 4203–4222. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/SP63933.2026.00053

[8] B. Zhang, H. Xin, M. Fang, Z. Liu, B. Yi, T. Li, and Z. Liu, "Traceback of poisoning attacks to retrieval-augmented generation," in *Proc. ACM Web Conf.* ACM, 2025, pp. 2085–2097.

[9] C. Xiang, T. Wu, Z. Zhong, D. Wagner, D. Chen, and P. Mittal, "Certifiably robust RAG against retrieval corruption," *arXiv preprint arXiv:2405.15556*, 2024.

[10] H. Chaudhari, G. Severi, J. Abascal, A. Suri, M. Jagielski, C. A. Choquette-Choo, M. Nasr, C. Nita-Rotaru, and A. Oprea, "Phantom: General backdoor attacks on retrieval augmented language generation," *ACM Trans. AI Secur. Priv.*, 2026, just Accepted.

[11] C. Li, J. Zhang, A. Cheng, Z. Ma, X. Li, and J. Ma, "CPA-RAG: Covert poisoning attacks on retrieval-augmented generation in large language models," *arXiv preprint arXiv:2505.19864*, 2025.

[12] T. Zhao, J. Chen, Y. Ru, H. Zhu, N. Hu, J. Liu, and Q. Lin, "Exploring knowledge poisoning attacks to retrieval-augmented generation," *Information Fusion*, vol. 127, p. 103900, 2026.

[13] D. Ru, L. Qiu, X. Hu, T. Zhang, P. Shi, S. Chang, J. Cheng, C. Wang, S. Sun, H. Li, Z. Zhang, B. Wang, J. Jiang, T. He, Z. Wang, P. Liu, Y. Zhang, and Z. Zhang, "RAGChecker: A fine-grained framework for diagnosing retrieval-augmented generation," in *Adv. Neural Inf. Process. Syst.*, vol. 37, 2024, pp. 21 999–22 027.

[14] J. Saad-Falcon, O. Khattab, C. Potts, and M. Zaharia, "ARES: An automated evaluation framework for retrieval-augmented generation systems," in *Proc. NAACL-HLT*. ACL, 2024, pp. 338–354.

[15] A. Asai, Z. Wu, Y. Wang, A. Sil, and H. Hajishirzi, "Self-RAG: Learning to retrieve, generate, and critique through self-reflection," in *Proc. ICLR*, 2024.

[16] H. Trivedi, N. Balasubramanian, T. Khot, and A. Sabharwal, "Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions," in *Proc. ACL*. ACL, 2023, pp. 10 014–10 037.

[17] J. Cheney, L. Chiticariu, and W.-C. Tan, "Provenance in databases: Why, how, and where," *Found. Trends Databases*, vol. 1, no. 4, pp. 379–474, 2009.

[18] B. Jimenez Gutierrez, Y. Shu, Y. Gu, M. Yasunaga, and Y. Su, "HippoRAG: Neurobiologically inspired long-term memory for large language models," in *Adv. Neural Inf. Process. Syst.*, vol. 37, 2024, pp. 59 532–59 569.

[19] B. Jimenez Gutierrez, Y. Shu, W. Qi, S. Zhou, and Y. Su, "From RAG to memory: Non-parametric continual learning for large language models," in *Proc. ICML*, ser. Proc. Mach. Learn. Res., vol. 267. PMLR, 2025, pp. 21 497–21 515.

[20] D. Edge, H. Trinh, N. Cheng, J. Bradley, A. Chao, A. Mody, S. Truitt, D. Metropolitansky, R. O. Ness, and J. Larson, "From local to global: A graph RAG approach to query-focused summarization," *arXiv preprint arXiv:2404.16130*, 2024.

[21] H. Qian, Z. Liu, P. Zhang, K. Mao, D. Lian, Z. Dou, and T. Huang, "MemoRAG: Boosting long context processing with global memory-enhanced retrieval augmentation," in *Proc. ACM Web Conf.* ACM, 2025, pp. 2366–2377.

[22] X. Yin, J. Han, and P. S. Yu, "Truth discovery with multiple conflicting information providers on the web," *IEEE Trans. Knowl. Data Eng.*, vol. 20, no. 6, pp. 796–808, 2008.

[23] Y. Li, J. Gao, C. Meng, Q. Li, L. Su, B. Zhao, W. Fan, and J. Han, "A survey on truth discovery," *SIGKDD Explor. Newsl.*, vol. 17, no. 2, pp. 1–16, Feb. 2016. [Online]. Available: https://doi.org/10.1145/2897350.2897352

[24] X. L. Dong, E. Gabrilovich, K. Murphy, V. Dang, W. Horn, C. Lugaresi, S. Sun, and W. Zhang, "Knowledge-based trust: Estimating the trustworthiness of web sources," *Proc. VLDB Endow.*, vol. 8, no. 9, pp. 938–949, 2015.

[25] N. Cesa-Bianchi and G. Lugosi, *Prediction, Learning, and Games*. Cambridge Univ. Press, 2006.

[26] M. Herbster and M. K. Warmuth, "Tracking the best expert," *Mach. Learn.*, vol. 32, no. 2, pp. 151–178, 1998.

[27] J. Gama, I. Zliobaite, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM Comput. Surv.*, vol. 46, no. 4, pp. 44:1–44:37, 2014.

[28] S. Wares, J. Isaacs, and E. Elyan, "Data stream mining: Methods and challenges for handling concept drift," *SN Appl. Sci.*, vol. 1, no. 11, p. 1412, 2019.

[29] N. Gunasekara, B. Pfahringer, H. M. Gomes, A. Bifet, and Y. S. Koh, "Recurrent concept drifts on data streams," in *Proc. IJCAI*, 2024, pp. 8029–8037.

[30] D. Kifer, S. Ben-David, and J. Gehrke, "Detecting change in data streams," in *Proc. VLDB*, 2004, pp. 180–191.

[31] A. Bifet and R. Gavaldà, "Learning from time-changing data with adaptive windowing," in *Proc. SIAM Int. Conf. Data Mining*, 2007, pp. 443–448.

[32] E. S. Page, "Continuous inspection schemes," *Biometrika*, vol. 41, no. 1/2, pp. 100–115, 1954.

[33] D. G. Kleinbaum and M. Klein, *Survival Analysis: A Self-Learning Text*, 3rd ed. Springer, 2012.

[34] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger, "On calibration of modern neural networks," in *Proc. ICML*, ser. Proc. Mach. Learn. Res., vol. 70. PMLR, 2017, pp. 1321–1330.

[35] I. Diakonikolas, G. Kamath, D. M. Kane, J. Li, A. Moitra, and A. Stewart, "Robust estimators in high dimensions without the computational intractability," in *Proc. IEEE FOCS*, 2016, pp. 655–664.

[36] A. Singh and T. Joachims, "Fairness of exposure in rankings," in *Proc. ACM SIGKDD*. ACM, 2018, pp. 2219–2228.

[37] A. Mallen, A. Asai, V. Zhong, R. Das, D. Khashabi, and H. Hajishirzi, "When not to trust language models: Investigating effectiveness of parametric and non-parametric memories," in *Proc. ACL*. ACL, 2023, pp. 9802–9822.

[38] Z. Yang, P. Qi, S. Zhang, Y. Bengio, W. W. Cohen, R. Salakhutdinov, and C. D. Manning, "HotpotQA: A dataset for diverse, explainable multi-hop question answering," in *Proc. EMNLP*. ACL, 2018, pp. 2369–2380.

[39] S. E. Robertson, S. Walker, S. Jones, M. M. Hancock-Beaulieu, and M. Gatford, "Okapi at TREC-3," in *Proc. Text REtrieval Conf.*, 1994, pp. 109–126.

[40] M. D. McKay, R. J. Beckman, and W. J. Conover, "A comparison of three methods for selecting values of input variables in the analysis of output from a computer code," *Technometrics*, vol. 21, no. 2, pp. 239–245, 1979.

[41] M. G. Kendall, "A new measure of rank correlation," *Biometrika*, vol. 30, no. 1/2, pp. 81–93, 1938.

[42] S. Min, E. Wallace, S. Singh, M. Gardner, H. Hajishirzi, and L. Zettlemoyer, "Compositional questions do not necessitate multi-hop reasoning," in *Proc. ACL*. ACL, 2019, pp. 4249–4257.

[43] Google DeepMind, "Gemini 2.5: Our most intelligent AI model," https://deepmind.google/models/gemini/, 2025, accessed: 2026-09-05; model string: gemini-2.5-flash-lite.

[44] B. McMahan, "Follow-the-regularized-leader and mirror descent: Equivalence theorems and l1 regularization," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, G. Gordon, D. Dunson, and M. Dudík, Eds., vol. 15. Fort Lauderdale, FL, USA: PMLR, 11–13 Apr 2011, pp. 525–533. [Online]. Available: https://proceedings.mlr.press/v15/mcmahan11b.html

[45] W. Hoeffding, "Probability inequalities for sums of bounded random variables," *J. Amer. Statist. Assoc.*, vol. 58, no. 301, pp. 13–30, 1963.

[46] S. Xiao, Z. Liu, P. Zhang, N. Muennighoff, D. Lian, and J.-Y. Nie, "C-Pack: Packed resources for general Chinese embeddings," in *Proc. ACM SIGIR*. ACM, 2024, pp. 641–649.

[47] OpenAI, "GPT-4o system card," https://openai.com/index/gpt-4o-system-card/, 2024.