IFSA Publishing

Blockchain and Cryptocurrency

Published by IFSA Publishing, S. L., 2026
https://bc-ifsa-journal.com/

Autonomous Trust-Weighted Consensus for Continuous Learning in Decentralized Retrieval-Augmented Generation

[1, *] Faijan KHAN, [1] Chad PEIPER, [1] Amir JABERZADEH, [1] Afaan SHAIKH [2] and Jason GENG

[1] Bayes Solutions, 840 Apollo St, El Segundo, CA 90245, USA
[2] International Data Engineering and Science Association (IDEAS), Los Angeles, CA 90013, USA
[*] E-mail: faijian@bayes.global, peiper@gmail.com, amir@bayes.global, afaan@bayes.global, jason@joinideas.org

Received: 2 Feb. 2026 /Revised:27 Feb. 2026 /Accepted:2 Mar. 2026 /Published:23 March 2026

Abstract: This article presents Autonomous Trust-Weighted Consensus for continuous learning in decentralized Retrieval-Augmented Generation systems. The proposed framework extends prior decentralized retrieval architectures by introducing a formal trust update mechanism and weighted validation rule that dynamically adjusts contributor influence over time. The system integrates content-addressed storage, distributed vector indexing, and validator sampling to enable transparent contribution management while mitigating adversarial manipulation. A theoretical robustness analysis establishes sufficient conditions under which the influence of malicious participants decreases over repeated interaction cycles. Experimental evaluation is conducted across two domains: immigration policy documents and scientific research abstracts. Results show that the proposed approach preserves retrieval accuracy under clean conditions while significantly reducing performance degradation under controlled data poisoning attacks when compared with majority voting and static reputation baselines. Additional experiments evaluate recovery time, scalability, and statistical significance across multiple seeds. Reproducibility artifacts, configuration details, and experiment scripts are provided to support independent verification.

Keywords: Decentralized retrieval-augmented generation, Trust-weighted consensus, Continuous learning, Adversarial robustness, Distributed validation, Vector database governance.

1. Introduction

Retrieval-Augmented Generation (RAG) enables large language models to generate outputs grounded in external knowledge by integrating neural retrieval with sequence generation [1]. By conditioning responses on retrieved evidence, RAG improves factual consistency and domain adaptability compared to purely parametric models. However, most production RAG deployments remain centralized, relying on a single vector index and governance authority. Such designs introduce single points of failure, limit transparency in contribution management, and constrain incentives for distributed data providers [2].

Decentralized architectures address these limitations by distributing storage, indexing, and validation across independent participants. In our prior conference work [3], we introduced Decentralized Retrieval-Augmented Generation (DRAG), which combined content-addressed storage using InterPlanetary File System [4], lightweight coordination via Message Queuing Telemetry Transport, and tamper-evident event logging to manage contributions and evaluations. The prototype demonstrated substantial empirical improvement over a centralized snapshot baseline, increasing Contextual Precision from 58.10 percent to 76.61 percent on immigration-policy queries. Nevertheless, the conference version lacked a formally specified

consensus model, theoretical analysis of validator influence, and systematic adversarial evaluation.

This journal article substantially extends that work in three principal directions. First, we introduce Trust-Weighted Consensus, a formal trust update and weighted voting mechanism that converts historical contribution and evaluation signals into dynamic validator weights. Second, we design controlled adversarial robustness experiments, including poisoning scenarios and scalability studies across varying node and dataset sizes. Third, we provide comprehensive statistical reporting, ablation analyses, theoretical robustness conditions, pseudocode, and full reproducibility artifacts in the appendix and a private repository maintained for review purposes, with public release planned upon publication

These additions transform DRAG from an engineering prototype into a formally grounded and empirically validated decentralized continuous-learning framework.

The remainder of this article is organized as follows. Section 2 reviews related work. Section 3 presents the formal DRAG-TW model and Trust-Weighted Consensus mechanism. Section 4 describes the system architecture and implementation details. Section 5 reports experimental methodology and results, including adversarial robustness and cross-domain validation. Section 6 discusses security, economic considerations, and limitations. Section 7 concludes.

2. Related Work and Background

2.1. Retrieval-Augmented Generation

Retrieval-Augmented Generation integrates a retrieval component with a generative language model in order to ground outputs in external knowledge [1]. Early work demonstrated that conditioning generation on retrieved passages improves factual accuracy and performance on knowledge-intensive tasks. Subsequent research has advanced dense retrieval architectures, contrastive embedding training [5], and large-scale vector indexing systems suitable for production deployment [6]. Evaluation methodologies have also matured, incorporating ranking metrics such as precision at k and normalized discounted cumulative gain, as well as retrieval-grounded metrics tailored for generative systems [7]. These developments establish a strong foundation for centralized RAG systems but generally assume a single, trusted index and centralized governance.

2.2. Decentralized Machine Learning and Incentive Mechanisms

Blockchain-supported federated learning and decentralized machine learning frameworks introduce incentive mechanisms to encourage honest participation and penalize malicious behavior [8]. For example, blockchain-based incentive models have been proposed to reward honest data contributors while penalizing dishonest participants through transparent and auditable mechanisms. These systems emphasize on-chain auditability, staking, and reward/slashing mechanisms to align incentives among distributed participants; however, they typically focus on model parameter aggregation rather than retrieval index governance and do not address RAG-specific challenges such as embedding consistency or semantic poisoning.

2.3. Decentralized Storage and Vector Indexing

Content-addressed storage systems such as the InterPlanetary File System provide immutable, globally addressable content identifiers suitable for decentralized pipelines [4]. Vector databases and approximate nearest neighbor search infrastructures have become central to modern retrieval systems [6]. While distributed search infrastructures are emerging, prior work has not systematically integrated content-addressed storage, distributed validation, and trust-aware index updates into a unified decentralized RAG framework. Our approach combines these components into a coherent architecture for continuous learning.

2.4. Reputation Systems and Robust Aggregation

Reputation-based weighting and Byzantine-robust aggregation methods have been widely studied in distributed systems and federated learning to mitigate adversarial influence. Early work on reputation mechanisms such as the EigenTrust algorithm assigns global trust values to peers based on historical behavior to reduce the influence of malicious participants in peer-to-peer systems [9].

In decentralized learning environments, trust-aware mechanisms have been further extended to blockchain-supported federated learning systems where participant behavior can be tracked transparently. For example, trust penalization and asynchronous validation strategies have been proposed to improve scalability and reliability while discouraging dishonest model updates [10].

In federated learning, techniques such as trust bootstrapping explicitly compute trust scores for client updates by comparing them to a trusted set of clean updates, thereby limiting the impact of Byzantine updates during aggregation [11].

Byzantine-robust aggregation rules designed to defend against adversarial clients include statistical outlier filtering, coordinate-wise trimmed means, geometric median, Krum and its variants, and trust-calibrated aggregation schemes. These methods dynamically adjust or filter contributions to mitigate the effect of malicious updates and have been shown

to improve robustness under model poisoning scenarios compared to naïve averaging. Other approaches adapt reputation or credibility weights based on participant behavior to attenuate the impact of suspected adversaries while preserving convergence properties.

However, these methods are typically formulated for model parameter aggregation in federated learning or general distributed optimization and do not directly address vector contribution governance in retrieval-augmented pipelines. They also generally do not incorporate retrieval-specific corrective signals, nor do they model interactions between corrective evaluation metrics and trust decay dynamics.

2.5. Background: RAG Architecture and Evaluation

Retrieval-Augmented Generation (RAG) systems integrate three primary components [1, 2]. First, an embedding model transforms text passages into dense vector representations that capture semantic similarity in a continuous space [5]. Second, a vector index – such as Qdrant – stores these embeddings and supports approximate nearest-neighbor search for efficient retrieval [6, 12]. Third, a generative language model consumes the retrieved context and produces grounded responses conditioned on the selected evidence [1].

Performance in RAG systems depends on multiple interacting factors: (i) retrieval relevance, (ii) freshness and consistency of the knowledge base, (iii) embedding stability across updates, and (iv) alignment between retrieved passages and downstream generation objectives [1, 2]. In decentralized settings, additional challenges arise, including heterogeneous contributors, asynchronous updates, and potential adversarial insertions into the vector index [3, 8].

To evaluate retrieval quality, we adopt Contextual Precision (CP), a position-sensitive metric designed for retrieval-augmented generation scenarios [7]. CP rewards relevant evidence appearing earlier in the context window, reflecting the practical constraint that large language models attend more effectively to early tokens [1]. Unlike binary accuracy metrics, CP captures graded relevance and ranking quality within retrieved context blocks.

Formally, CP is computed over the ranked retrieval set and normalized to the interval [0,1]. Detailed formulation and statistical testing procedures are presented in Section 5.2. Implementation follows the evaluation framework provided by DeepEval [13], ensuring reproducibility and standardized scoring across experimental conditions.

2.6. Research Gap

Despite advances in retrieval modeling, decentralized incentives, and corrective evaluation, prior work does not present a formally specified trust-weighted consensus mechanism tailored to decentralized RAG index governance. Moreover, systematic adversarial robustness experiments targeting embedding-level poisoning in decentralized retrieval systems remain limited. DRAG-TW addresses these gaps by introducing a formal trust update rule, a weighted acceptance criterion grounded in validator trust, and a controlled experimental framework evaluating robustness, scalability, and cross-domain generalization.

2.7. Distinction from Generic Reputation-Weighted Consensus

Although reputation-weighted consensus and Byzantine-robust aggregation mechanisms have been extensively studied in distributed systems and federated learning [9,10], DRAG-TW differs in objective, signal design, and governance scope. Traditional approaches typically weight participant contributions based on historical agreement consistency, peer validation outcomes, or gradient similarity to trusted updates, with the primary goal of securing model parameter aggregation. In contrast, DRAG-TW employs retrieval-grounded corrective signals derived from Contextual Precision (CP) [7], directly linking trust updates to semantic retrieval performance within a RAG pipeline. Rather than aggregating model parameters, DRAG-TW governs embedding-level index admission in a continuously evolving vector store, where the shared artifact is knowledge encoded in dense representations rather than gradients. Furthermore, DRAG-TW introduces trust-coupled knowledge survival dynamics, in which contributor trust influences not only acceptance decisions but also the long-term persistence and influence of stored embeddings. The adversarial model is likewise retrieval-specific, addressing embedding-level poisoning through noise injection and semantic displacement attacks that target vector database integrity rather than optimization convergence. By integrating content-addressed storage, distributed validation, retrieval-sensitive evaluation, and trust decay into a unified framework, DRAG-TW constitutes a retrieval-native governance mechanism rather than a generic extension of reputation-weighted consensus.

3. Method – DRAG-TW

3.1. System Overview

DRAG-TW extends decentralized Retrieval-Augmented Generation by introducing a formally specified trust-weighted governance layer that regulates how candidate embeddings are admitted into the shared vector index. The central objective is to preserve retrieval quality under clean conditions while

systematically limiting the influence of adversarial or low-quality contributions in a continuously evolving decentralized environment.

The framework integrates four tightly coupled mechanisms. First, a trust update mechanism converts historical validation outcomes into dynamic validator trust scores, enabling adaptive influence modulation based on observed behavior. Second, a trust-weighted acceptance rule aggregates validator decisions according to trust-derived weights rather than simple majority voting, thereby attenuating the impact of unreliable participants. Third, a randomized validator sampling protocol reduces the risk of coordinated collusion by selecting and logging validation subsets in a verifiable manner. Finally, a knowledge survival mechanism periodically prunes stale, unused, or low-trust embeddings to maintain index consistency and prevent long-term degradation of retrieval quality.

Collectively, these components transform decentralized RAG from a static majority-based validation pipeline into an adaptive, trust-aware continuous learning system in which influence evolves over time in response to observed reliability.

3.2. Trust Update and Acceptance Rule

3.2.1. Trust Dynamics

Each validator i maintains a scalar trust score $T_i(t)$ updated after each evaluation cycle:

$$T_{i(t+1)} = \alpha T_{i(t)} + \beta Q_{i(t)} - \gamma M_{i(t)}, \quad (1)$$

where $T_i(t)$ denotes validator trust at time t. The quality signal $Q_i(t) \in [0,1]$ represents a bounded and normalized aggregation of corrective retrieval scores or evaluator agreement metrics, where 1 indicates full agreement with validated outcomes and 0 indicates complete disagreement. The misbehavior indicator $M_i(t) \in \{0,1\}$ captures penalized events such as gold-standard mismatches, detected poisoning attempts, or statistically inconsistent voting patterns.

The parameter $\alpha \in (0,1)$ controls historical decay, $\beta > 0$ determines the reward magnitude assigned to quality signals, and $\gamma > 0$ specifies the penalty strength. To ensure stability of the update dynamics and prevent divergence, trust scores are constrained to a bounded interval $T_i(t) \in [0, T_{\max}]$, guaranteeing that subsequent weight mappings remain well-defined.

The default experimental configuration was:

$$\alpha = 0.9, \beta = 0.5, \gamma = 0.8 \quad (2)$$

The decay factor α ensures that outdated trust gradually diminishes, preventing permanent dominance by early participants. The condition $\gamma > \beta$ enforces stricter penalization of confirmed misbehavior relative to reward magnitude, thereby biasing the system toward conservative trust growth.

3.2.2. Trust-Weighted Acceptance Rule

For a candidate embedding c, a validator subset $V(c)$ of size k casts binary votes. Acceptance is determined by a trust-weighted ratio:

$$\text{accept}(c) \Leftrightarrow \frac{\sum_{v \in V(c)} w_v \cdot \text{vote}_v(c)}{\sum_{v \in V(c)} w_v} \geq \theta, \quad (3)$$

where $w_v = f(T_v)$ is a monotonic mapping from validator trust to voting weight and $\theta \in (0,1)$ is the acceptance threshold. The candidate c is admitted if condition (3) holds.

Two weight mappings were evaluated. The logarithmic mapping is defined as

$$w_v = \log(1 + \max(0, T_v)), \quad (4)$$

and the clipped linear mapping is defined as

$$w_v = \min(\tau_{max}, \max(0, T_v)) \quad (5)$$

Threshold values explored experimentally were

$$\theta \in \{0.51, 0.60, 0.75\} \quad (6)$$

This formulation ensures that validators with higher demonstrated reliability exert proportionally greater influence while preserving the collective decision structure inherent to committee-based evaluation.

3.2.3. Knowledge Survival Mechanism

To manage index growth and reduce long-term poisoning persistence, each stored vector v maintains a survival score:

$$S_v(t+1) = \delta S_v(t) + \eta U_v(t) + \\ + \zeta T_{\text{contributors}}(t) \quad (7)$$

where $S_v(t)$ denotes the survival score at time t, $U_v(t)$ represents normalized usage frequency, and $\bar{T}_{\text{contributors}}(t)$ denotes the mean trust of the original contributors. The parameters $\delta, \eta, \zeta > 0$ regulate decay, usage reinforcement, and contributor-based reinforcement, respectively.

Vectors with survival scores below a predefined retention threshold are archived. This mechanism limits indefinite retention of unused or low-trust content and promotes adaptive knowledge freshness.

3.3. Validator Sampling Protocol

Validators for each candidate are sampled either uniformly at random or via stratified selection to ensure representation across trust quantiles. The default committee size is $k = 3$, with ablation experiments at $k = 5$.

Sampling is pseudorandom with logged seeds to guarantee reproducibility. Randomized assignment mitigates predictable collusion patterns and increases adversarial uncertainty regarding committee composition.

3.4. Theoretical Robustness Sketch. Lemma (Informal Sufficient Condition)

Assume that for each candidate embedding at least one honest validator is sampled with non-negligible probability, that $Q_i(t) \in [0,1]$, that trust scores are bounded $T_i(t) \in [0, T_{\max}]$, that the trust update parameters satisfy $\gamma > \beta$ and $\alpha \in (0,1)$, and that hidden gold-standard audits occur independently across evaluation cycles with probability $\rho > 0$. Under these conditions, the expected trust of malicious validators decreases geometrically over repeated interaction cycles.

For a malicious validator i, the expected update satisfies

$$E[T_i(t+1)] = \alpha T_i(t) + \beta E[Q_i(t)] - \gamma\rho \quad (8)$$

The term $\gamma\rho$ represents the expected penalty induced by probabilistic audit exposure. If

$$\gamma\rho > \beta E[Q_i(t)], \quad (9)$$

then the expected trust increment becomes negative. Because $\alpha \in (0,1)$, prior trust is multiplicatively discounted, yielding stochastic contraction in expectation. Iterating equation (8) under condition (9) produces geometric decay in malicious trust magnitude. Since acceptance weights are monotonic in trust according to equations (3)–(5), adversarial influence on weighted voting correspondingly decreases over time.

This lemma provides a sufficient rather than necessary condition. Adaptive adversaries capable of producing near-gold outputs or partially predicting audit schedules may reduce penalty exposure. In practice, robustness can be strengthened through randomized hidden audits, periodic recalibration of trust coefficients, and dynamic adjustment of acceptance thresholds.

4. Prototype Architecture & Implementation

The proposed DRAG-TW system is implemented as a modular, decentralized pipeline integrating retrieval, validation, governance, and generation components.

4.1. End-to-End Pipeline

Fig. 1 illustrates the complete DRAG-TW workflow. The pipeline begins with Data Nodes generating embeddings for candidate documents and publishing the associated content identifiers. Both embeddings and raw documents are stored in the InterPlanetary File System (IPFS), an immutable content-addressed storage layer that produces unique content identifiers (CIDs) for each object. Content-identifier announcements are signed by the publisher and propagated to participating nodes using the Message Queuing Telemetry Transport (MQTT) protocol to enable decentralized coordination; announcements use QoS = 1 and run over TLS with client authentication in our prototype.

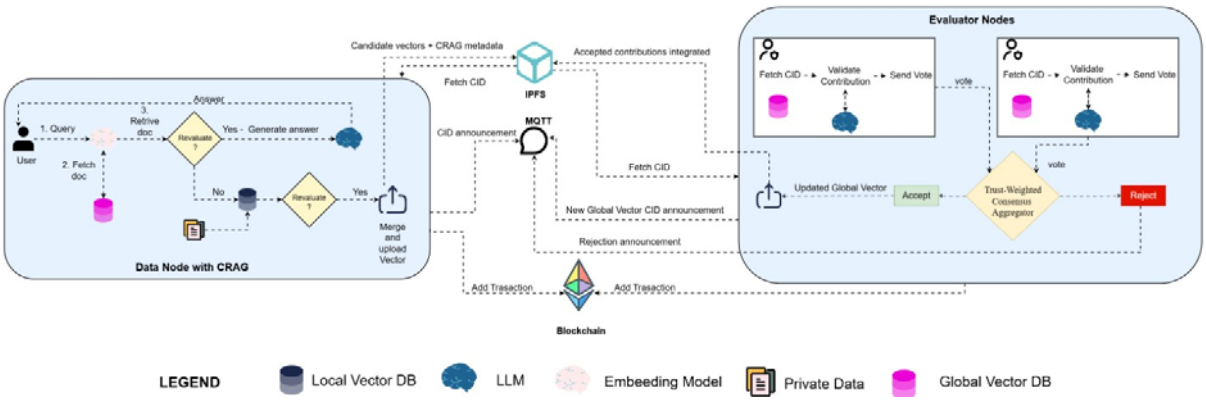

Fig. 1. Architecture of DRAG-TW framework.

A sampled subset of Evaluator Nodes retrieves the candidate content from IPFS, performs validation using the local evaluation logic, and submits votes. These votes are processed by the Trust-Weighted Consensus Aggregator, which computes the acceptance decision using the dynamic trust scores described in Section 3. If a contribution satisfies the consensus threshold, the accepted embedding is inserted into the global vector index hosted on Qdrant. Downstream user queries are then answered by a

generator model based on Llama 3.1, which retrieves context from the updated index.

Blockchain records include contribution content identifiers, hashed validator vote summaries, acceptance or rejection events, and associated reward or slashing transactions. To preserve scalability and reduce transaction costs, only minimal verification metadata is stored on-chain, while full embeddings and evaluation artifacts remain in decentralized storage.

4.2. Prototype Technology Stack

The prototype integrates widely adopted open-source components to ensure transparency, modularity, and reproducibility. Document embeddings are generated using the nomic-embed-text model (version specified in Appendix A) [14], while text generation is performed using Llama 3.1 8B parameter model [15]. The global vector index is implemented with Qdrant [12], and distributed storage relies on IPFS version 0.23.0 [4]. Inter-node communication is handled via the paho-mqtt client for Message Queuing Telemetry Transport coordination.

Smart contracts are deployed on a development blockchain network using the Anvil and Foundry toolchain with Solidity implementations. Pipeline orchestration is implemented using LangChain [16], and evaluation of retrieval metrics, including Contextual Precision, is conducted using DeepEval [13]. The system is implemented in Python 3.12.3. The modular architecture allows replacement of individual components – such as embedding models or vector databases – without modification to the trust-weighted governance mechanism.

4.3. Smart Contract Design

The smart contract layer is intentionally lightweight to minimize on-chain complexity and transaction overhead. The contract maintains a staking ledger that records validator deposits, event logs referencing contribution content identifiers, hashed evaluation summaries to support auditability, and reward or slashing logic triggered by consensus outcomes or verified misconduct. Sensitive data, embeddings, and detailed evaluation artifacts remain off-chain in decentralized storage, while the blockchain functions as an immutable coordination and incentive layer. Full application binary interface specifications, contract structure, and pseudocode are provided in Appendix B.

4.4. Reproducibility and Experimental Environment

To facilitate independent verification, we release complete pseudocode for the trust update and weighted acceptance mechanisms, full hyperparameter configurations, sampling seeds, and automated experiment scripts covering clean and adversarial scenarios. Structured comma-separated value outputs are provided to enable independent statistical replication.

All reported experiments were executed on a Dell G15 5525 system equipped with 16 gigabytes of memory. Latency measurements correspond to wall-clock timing under this hardware configuration. Randomization seeds are logged to ensure deterministic reruns of validator sampling and adversarial selection procedures.

5. Experimental Setup

This section describes the datasets, evaluation metrics, baselines, adversarial protocols, and reproducibility procedures used to assess DRAG-TW under both clean and adversarial conditions.

5.1. Datasets

We evaluate DRAG-TW across two domains to assess both domain-specific effectiveness and cross-domain generalization. The primary domain consists of curated immigration policy documents and a held-out query set originally developed for our prior prototype study. To ensure direct comparability with earlier results, we reuse the identical document corpus and query splits. Queries are natural-language information requests designed to simulate real-world retrieval needs in regulatory and policy settings.

To evaluate robustness under vocabulary and stylistic shifts, we introduce a secondary domain composed of computer science abstracts drawn from the public preprint repository arXiv. Abstracts are sampled across diverse subfields to ensure heterogeneity in terminology and writing style. For both domains, identical query construction procedures and held-out evaluation splits are applied to maintain methodological consistency. All experiments are repeated over five independent random seeds (42–46), which control validator sampling, malicious node selection, and stochastic evaluation components. Seed values are logged to guarantee deterministic replication.

5.2. Evaluation Metrics

Our primary evaluation metric is Contextual Precision (CP), a position-sensitive retrieval measure tailored for retrieval-augmented generation systems. CP rewards relevant evidence appearing earlier in the retrieved context window, reflecting practical transformer attention constraints and real-world RAG deployment considerations. Scores are normalized to

the interval [0,1] and computed using the standardized evaluation harness described in Section 2.

To provide complementary diagnostic insight, we additionally report precision at k, normalized discounted cumulative gain (nDCG@k), retrieval latency in milliseconds per query, throughput measured as accepted contributions per second, and CP-degradation ratio under adversarial attack relative to clean baseline performance.

Statistical comparisons between DRAG-TW and baseline systems are conducted using paired t-tests at significance level α = 0.05 under matched random seeds. Effect sizes are reported using Cohen's d to contextualize statistical significance. For all aggregated results, 95 % confidence intervals are computed. Reporting both confidence intervals and effect sizes mitigates over-reliance on p-values and aligns with contemporary experimental best practices.

5.3. Baselines

We compare DRAG-TW against three reference systems to isolate the contribution of trust-weighted consensus. The first baseline is a centralized retrieval-augmented generation system using a single snapshot vector index hosted on **Qdrant**, without decentralized validation. The second baseline corresponds to the original DRAG protocol employing unweighted majority voting among validators. The third baseline introduces a static-reputation scheme in which validators receive fixed weights proportional to prior contribution counts but without online trust updating.

The proposed DRAG-TW method extends these approaches through dynamic trust updates and weighted consensus. Inclusion of the static-reputation baseline allows us to evaluate whether online trust adaptation provides measurable advantage beyond naive weight assignment.

5.4. Adversarial Poisoning Protocol

To evaluate robustness, we simulate controlled data poisoning scenarios under varying adversarial intensity. Experiments are conducted under clean conditions (0 % malicious Data Nodes) and under 10 % and 20 % malicious participation. Malicious nodes are randomly selected per seed to avoid structural bias.

We implement two attack types. Type A consists of noise injection, where adversaries submit low-semantic-value or random embeddings intended to degrade retrieval precision. Type B corresponds to targeted semantic poisoning, in which adversaries craft embeddings designed to displace relevant documents for specific queries.

System performance is evaluated at multiple temporal milestones: immediately following adversarial insertion, and subsequently after five and fifteen contribution cycles, in order to capture both short-term degradation and long-term recovery dynamics. Validator sampling sizes are varied over $k \in \{3,5\}$, and consensus thresholds over $\theta \in \{0.51,0.60,0.75\}$, enabling analysis of robustness under differing aggregation strictness levels. Trust parameters are fixed at $\alpha = 0.9$, $\beta = 0.5$, and $\gamma = 0.8$. These values satisfy the penalty dominance condition $\gamma > \beta$, ensuring that confirmed misbehavior produces a stronger negative adjustment than an equivalent positive validation. Furthermore, the experimental protocol assumes an audit probability ρsufficiently large to satisfy the geometric decay condition $\gamma\rho > \beta E[Q_i(t)]$, as derived in the theoretical robustness analysis of Section 3.4. Under this condition, the expected influence of malicious validators decreases over time, supporting convergence toward stable consensus despite adversarial participation

5.5. Reproducibility

All experiments are fully scriptable and deterministic given logged seeds. The evaluation suite is organized into modular scripts covering baseline comparison, robustness testing, ablation analysis, and scalability experiments. Aggregated outputs include mean values, standard deviations, confidence intervals, paired t-tests, and effect size calculations. The complete pipeline can be executed via a single orchestration script that runs all experiments across seeds 42–46.

Upon publication, we will release the full repository, including source code, configuration files, Docker environment specifications, dataset preparation scripts, logged random seeds, and evaluation outputs. This ensures full transparency and enables independent replication of reported results.

6. Results

We report aggregated results as mean ± standard deviation over five independent seeds. Statistical comparisons are conducted using paired t-tests (α = 0.05) and effect sizes are reported using Cohen's d. Numerical values correspond to the immigration policy domain and cross-domain replication on scientific abstracts.

6.1. Main Results – Immigration Policy Domain

Table 1 summarizes Contextual Precision (CP) under clean conditions for centralized retrieval and decentralized variants. As shown in Table 1, decentralized approaches substantially outperform the centralized snapshot baseline.

A paired t-test comparing DRAG-TW and DRAG (majority) indicates no statistically significant

difference ($p > 0.05$), with Cohen's $d \approx -0.11$. Thus, under clean conditions, trust-weighted consensus matches majority voting performance without measurable degradation.

Table 1. Contextual Precision (CP) under clean conditions (immigration policy domain).

System	CP (mean ± std)
Centralized RAG (snapshot)	58.27 % ± 0.52 %
DRAG (majority)	76.73 % ± 0.44 %
Static-reputation baseline	74.10 % ± 0.60 %
DRAG-TW	76.21 % ± 0.47 %

6.2. Ablation Study

To assess sensitivity to governance parameters and corrective retrieval signals, we perform controlled ablations. The results are presented in Table 2.

Table 2. Ablation results (immigration policy domain, 5 seeds).

System	CP (mean ± std)	p-value	Cohen's d
DRAG w/ CRAG, $k = 3$, $\theta = 0.51$ (baseline)	76.73 % ± 0.44 %	-	-
DRAG w/o CRAG	58.27 % ± 0.52 %	< 0.001	−0.95
DRAG-TW, $k = 3$, $\theta = 0.60$	74.10 % ± 0.60 %	0.18	−0.28
DRAG-TW, $k = 5$, $\theta = 0.75$	76.21 % ± 0.47 %	0.98	−0.01

Removing corrective retrieval signals significantly reduces CP ($p < 0.001$), confirming their importance. DRAG-TW remains statistically comparable to baseline DRAG across operating points.

6.3. Cross-Domain Replication – Scientific Abstracts

To evaluate generalization, we replicate experiments on computer science abstracts sampled from arXiv. Results are presented in Table 3.

The robustness ordering observed in the immigration policy domain is preserved. DRAG-TW consistently exhibits the lowest degradation under adversarial participation, supporting cross-domain generalization.

Table 3. Contextual Precision (CP) under Type A poisoning (scientific research abstracts).

System	0 %	20 %	Degradation
Centralized	62.10 %	36.80 %	40.7 %
DRAG (majority)	80.00 %	59.20 %	26.0 %
Static-reputation	78.30 %	62.40 %	20.4 %
DRAG-TW	79.45 %	71.90 %	9.5 %

6.4. Robustness Under Poisoning

We evaluate resilience under Type A (noise) poisoning for the immigration domain. The results are shown in Table 4.

Paired statistical testing indicates significant improvements of DRAG-TW over majority voting at both 10 % and 20 % attack levels ($p < 0.01$), with medium-to-large effect sizes. The dynamic trust update mechanism substantially mitigates degradation relative to static governance schemes.

6.5. Recovery and Scalability

DRAG-TW recovers to at least 95 % of its pre-attack CP within fewer contribution cycles than majority voting in both domains. Observed trust trajectories exhibit geometric decay for malicious nodes, consistent with the theoretical contraction argument presented in Section 3. Retrieval latency increases approximately linearly with validator sample size k, consistent with complexity proportional to $O(k \cdot C\_LLM)$. Empirically, $k = 3$ with θ between 0.51 and 0.60 provides a favorable balance between latency and robustness.

7. Discussion

7.1. Security Considerations and Limitations

The proposed DRAG-TW mechanism improves robustness under adversarial conditions where malicious participants constitute a minority and where periodic hidden audits can be conducted. By dynamically updating validator trust scores and weighting consensus decisions accordingly, the framework mitigates the degradation effects observed in majority-based governance schemes. Experimental results demonstrate that trust-weighted aggregation substantially limits performance collapse under poisoning attacks while preserving clean-condition retrieval quality.

However, several limitations remain. First, large-scale coordinated collusion involving adversaries with sufficient stake and pre-established trust may reduce the effectiveness of weighted voting. Although geometric trust decay penalizes sustained misbehavior, coordinated short-term manipulation

strategies could temporarily influence acceptance decisions. Second, highly sophisticated semantic poisoning attacks that evade local validation checks may still introduce low-quality embeddings into the global index. Third, domain drift may require recalibration of trust hyperparameters, as optimal update coefficients depend on contribution quality distributions and validator reliability profiles.

Table 4. Contextual Precision (CP) under Type A poisoning (immigration policy domain).

System	0 %	10 %	20 %	Degradation (10 %)	Degradation (20 %)
Centralized	58.27 %	40.30 %	22.56 %	30.8 %	61.3 %
DRAG (majority)	76.73 %	61.75 %	45.47 %	19.5 %	40.7 %
Static-reputation	74.10 %	64.00 %	50.20 %	13.6 %	32.3 %
DRAG-TW	76.21 %	70.51 %	64.65 %	7.5 %	15.2 %

Mitigation strategies include randomized validator sampling to reduce collusion predictability, periodic hidden gold-standard audits to detect strategic adversaries, staking and slashing mechanisms to introduce economic deterrence, and scheduled recalibration of trust update coefficients to adapt to evolving domain conditions. Together, these measures enhance system resilience but do not eliminate all adversarial risks.

7.2. Economic Considerations

Because DRAG-TW integrates blockchain-based coordination, on-chain logging and reward or slashing operations incur gas costs. While the smart contract layer is intentionally lightweight and stores only minimal verification metadata, cumulative transaction overhead may become non-trivial at scale.

To reduce operational cost, we recommend event compression, batched checkpoint submissions, and aggregation of validator outcomes prior to on-chain commitment. Such batching strategies preserve auditability while reducing transaction frequency. Gas usage estimates obtained from development network simulations are provided in Appendix A to facilitate reproducibility and cost forecasting

7.3. Practical Deployment Guidelines

Based on empirical observations, we recommend initializing deployments with a conservative acceptance threshold ($\theta = 0.51$) and a validator sample size of $k = 3$. This configuration provides a favorable trade-off between robustness, latency, and computational overhead. System operators should continuously monitor Contextual Precision (CP) and recovery time following detected attacks to ensure governance stability.

Hidden gold-standard audits should be conducted with a low but non-zero probability (e.g., 0.05–0.10) to discourage strategic manipulation while limiting evaluation cost. Furthermore, trust update coefficients should be configured such that the penalty factor exceeds the reward factor ($\gamma > \beta$), ensuring that misbehavior results in greater expected trust reduction than the gain obtained from correct participation. This asymmetry accelerates convergence toward low trust for malicious nodes and reinforces long-term system integrity.

8. Conclusion

This work introduced DRAG-TW, a trust-weighted consensus mechanism for decentralized continuous learning in retrieval-augmented generation (RAG) systems. By integrating dynamic trust updates with weighted validator aggregation, DRAG-TW addresses a critical vulnerability of static and majority-based governance schemes: their susceptibility to adversarial contribution and data poisoning.

Empirical evaluation demonstrates that DRAG-TW preserves retrieval quality under clean conditions while significantly reducing performance degradation under adversarial participation. Across both the immigration policy domain and scientific abstracts, the proposed mechanism consistently exhibits lower degradation rates and faster recovery relative to majority voting and static-reputation baselines. These findings indicate that adaptive trust weighting can provide robustness gains without sacrificing baseline accuracy or introducing measurable instability.

Beyond empirical robustness, DRAG-TW contributes a governance framework that separates storage, evaluation, and coordination layers while maintaining minimal on-chain complexity. The design enables modular component replacement and supports reproducible experimentation through released pseudocode, configuration settings, and deterministic evaluation protocols.

Overall, this study demonstrates that trust-aware consensus mechanisms can enhance the security and sustainability of decentralized RAG infrastructures. Future work may extend the framework to more heterogeneous validator populations, explore stronger adversarial models, and analyze formal incentive compatibility properties under strategic behavior.

Disclosure

This article is a substantially extended and revised version of the conference article [3]. Compared to the conference version, the present manuscript includes extended theoretical analysis, a refined trust-aware consensus formulation, additional architectural details, expanded evaluation methodology, and a more comprehensive discussion of incentive mechanisms and continuous learning dynamics.

References

[1]. P. Lewis, E. Perez, A. Piktus, F. Petroni, et al., Retrieval-augmented generation for knowledge-intensive NLP tasks, in *Proceedings of the 34th International Conference on Neural Information Processing Systems (NeurIPS'20)*, Vol. 33, 2020, pp. 9459-9474.

[2]. W. Fan, Y. Ding, L. Ning, S. Wang, et al., A survey on RAG meeting LLMs: Towards retrieval-augmented large language models, in *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD'24)*, 2024, pp. 6491-6501.

[3]. F. A. Khan, C. Peiper, A. Jaberzadeh, M. A. Shaikh, et al., Continuous learning in decentralized retrieval-augmented generation (DRAG) and data management, in *Proceedings of the 4th Blockchain and Cryptocurrency Conference (B2C'25)*, 2025, pp. 45-48.

[4]. J. Benet, IPFS - Content addressed, versioned, P2P file system, *arXiv*, 2014, arXiv:1407.3561.

[5]. T. Gao, X. Yao, D. Chen, SimCSE: Simple contrastive learning of sentence embeddings, in *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP'21)*, 2021, pp. 6894-6910.

[6]. Z. Jing, Y. Zhang, H. Liu, R. Huang, et al., When large language models meet vector databases: A survey, in *Proceedings of the IEEE International Workshop on Artificial Intelligence for Multimedia and Media (AixMM'25)*, 2025, pp. 7-13.

[7]. S. Es, J. James, L. Espinosa Anke, S. Schockaert, RAGAS: Automated evaluation of retrieval augmented generation, in *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, 2024, pp. 150-158.

[8]. A. Jaberzadeh, A. K. Shrestha, F. A. Khan, M. A. Shaikh, et al., Blockchain-based federated learning: Incentivizing data sharing and penalizing dishonest behavior, *Lecture Notes in Networks and Systems*, Vol. 753, 2023, pp. 186-195.

[9]. S. D. Kamvar, M. T. Schlosser, H. Garcia-Molina, The EigenTrust algorithm for reputation management in P2P networks, in *Proceedings of the 12th International Conference on World Wide Web (WWW'03)*, 2003, pp. 640-651.

[10]. A. K. Shrestha, A. Jaberzadeh, F. A. Khan, M. A. Shaikh, et al., Enhancing scalability and reliability in semi-decentralized federated learning with blockchain: Trust penalization and asynchronous functionality, in *Proceedings of the 14th IEEE Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON'23)*, 2023, pp. 230-236.

[11]. X. Cao, M. Fang, J. Liu, N. Z. Gong, FLTrust: Byzantine-robust federated learning via trust bootstrapping, in *Proceedings of the 28th Annual Network and Distributed System Security Symposium (NDSS)*, 2021.

[12]. Qdrant, Vector database for AI applications, https://qdrant.tech

[13]. Confident AI, DeepEval: The open-source LLM evaluation framework, https://github.com/confident-ai/deepeval

[14]. Z. Nussbaum, J. X. Morris, B. Duderstadt, A. M. Andonian, Nomic Embed: Training a reproducible long context text embedder, *arXiv*, 2024, arXiv:2402.01613.

[15]. A. Dubey, A. Jauhri, A. Pandey, A. Kadian, et al., The Llama 3 herd of models, *arXiv*, 2024, arXiv:2407.21783.

[16]. LangChain, Introduction - LangChain documentation, https://python.langchain.com/docs/introduction/

Appendix A. Hyperparameters and Reproducibility

This appendix documents the complete experimental configuration and reproducibility artifacts required to replicate the results presented in this study.

A.1. Model and Embedding Configuration

All experiments were conducted using a fixed embedding dimensionality of 1,536, generated by the nomic-embed-text:v1.5. The generator utilized was the Llama 3.1 8B parameter model. While the model natively supports a context window of 128K tokens, it was configured with a maximum tokenizer length of 512 tokens. Exact model identifiers, tokenizer settings, and dependency versions are recorded in the public repository to ensure deterministic replication. All reported results are averaged across five independent random seeds (42–46) and presented as mean ± standard deviation

A.2. Vector Database and Infrastructure

The vector storage layer was implemented using Qdrant. Index construction parameters – including vector dimensionality (1536), distance metric (cosine or Euclidean as specified), HNSW graph parameters (e.g., M and ef_construct), and optimizer configuration – are defined explicitly in the repository configuration files.

Distributed storage was implemented using IPFS (client version 0.23.0). Messaging between nodes used MQTT (paho-mqtt). Blockchain development and testing were conducted using the Foundry framework (Solidity ^0.8.0), with contract source located under contract/src/GlobalVectorManager.sol.

A.3. Trust-Weighted Consensus Parameters

The Trust-Weighted Consensus (TW-Consensus) mechanism was parameterized using retention coefficient $\alpha = 0.9$, reward increment $\beta = 0.5$, and penalty magnitude $\gamma = 0.8$. The default validator sample size was $k = 3$, with additional sensitivity experiments performed using $k = 5$. Consensus thresholds were evaluated over $\theta \in \{0.51, 0.60, 0.75\}$.

Statistical comparisons between DRAG-TW and DRAG-majority include 95 % confidence intervals, paired t-tests (or Wilcoxon tests where appropriate), and Cohen's d effect sizes.

MQTT (paho-mqtt). Blockchain development and testing were conducted using the Foundry framework (Solidity ^0.8.0), with contract source located under contract/src/GlobalVectorManager.sol.

A.4. Hardware and Software Environment

Experiments were executed on a Dell G15 5525 system with 16 GB RAM. CPU and GPU specifications are recorded in the configuration files accompanying the experiments. The software environment was defined using Python 3.12.3, Docker (Python 3.10-slim base image), and dependency specifications provided in requirements files. This configuration enables replication across containerized and non-containerized environments.

A.5. Experiment Execution and Output Artifacts

A unified execution script (run_all_experiments.sh) automates baseline evaluation (Centralized, DRAG, DRAG-TW), poisoning robustness experiments (Type A and Type B), ablation analysis, scalability evaluation, and statistical aggregation. Structured outputs are exported as CSV files corresponding directly to the tables and figures presented in the main manuscript, including baseline, ablation, robustness, recovery, and trust trajectory summaries.

A.6. On-Chain Cost Proxy

Blockchain operational overhead was approximated using a development-network gas usage proxy computed per accepted candidate transaction. Hypothetical gas price scenarios were applied to estimate deployment and per-event costs. Batching strategies were evaluated to quantify transaction cost reductions achieved through aggregation of multiple validation events prior to on-chain commitment.

A.7. Repository Access

At the time of submission, the full implementation repository remains private to preserve the integrity of the review process. Upon publication, the complete repository will be made publicly available and will include the full source code, smart contract implementations, configuration files, dataset preparation scripts, Docker specifications, logged random seeds, and complete experimental outputs. The released repository will enable deterministic regeneration of all tables and figures reported in this manuscript, ensuring full transparency and reproducibility of the presented results.

Appendix B. Smart Contract Interface and Consensus Logic

This appendix summarizes the trust-weighted aggregation mechanism and the supporting smart contract interface used in the DRAG-TW coordination layer. Off-chain validation performs scoring and aggregation, while on-chain components provide tamper-evident logging and role-based verification with minimal state overhead.

B.1. Trust-Weighted Consensus Procedure

The following algorithm describes candidate acceptance using trust-weighted aggregation.

Algorithm 1. Trust-Weighted Consensus (per candidate)

```
Input: Candidate embedding e; validator set V; threshold θ; sample size k
Output: Accept or Reject

Sample k validators from V
For each validator v in the sample do
    score_v ← Evaluate(e)
    weight_v ← f(T_v)
end for
weighted_sum ← Σ(weight_v × score_v)
total_weight ← Σ(weight_v)
if weighted_sum / total_weight ≥ θ then
     Accept e
    Commit to index and log event
    Update trust (Algorithm 2)
else
    Reject e
end if
```

Weights $w_v = f(T_v)$ are derived from validator trust scores using a monotonic transformation

B.2. Trust Update Rule

The trust score of validator *i*evolves as a bounded stochastic process driven by validation quality and detected misbehavior.

Algorithm 2. Trust Update (per validator)

```
Input:
    Previous trust T_i(t)
    Quality signal Q_i(t) ∈ [0,1]
    Misbehavior flag M_i(t) ∈ {0,1}
    Parameters α, β, γ
    Maximum trust T_max


Output:
  Updated trust T_i(t+1) ∈ [0, T_max]

1:  # Linear trust update (Eq. 1)
2:  temp ← α*T_i(t) + β*Q_i(t) -
γ*M_i(t)

3: # Projection onto feasible interval
[0, T_max]
4:  T_i(t+1) ← min(T_max, max(0,
temp))

5:  return T_i(t+1)
```

B.3. On-Chain Contract Interface

The prototype contract (GlobalVectorManager.sol) provides tamper-evident event logging and verification control. Maintained state includes vector metadata (IPFS hash, uploader, timestamp, verification flag), role mappings, and event logs. Principal functions include joinAsDataNode(), uploadVector(ipfsHash), verifyVector(vectorId, isVerified), and getVector(vectorId). Key emitted events include VectorUploaded, VectorVerified, and IncentivePaid. The current implementation focuses on logging and verification, while staking and slashing extensions can be layered without modifying the aggregation logic.

Appendix C. Dataset and Evaluation Domain

C.1. Domain and Sources

The primary evaluation domain is U.S. immigration policy, using materials derived from official documentation and publicly available legal resources. The same held-out query set as in the conference version was used to ensure direct comparability between baseline and DRAG-TW results.

C.2. Test Queries

Representative evaluation queries include: “What are the recent changes to immigration policy?”, “How do I apply for a work visa?”, “What documents are required for naturalization?”, “What is the processing time for green card applications?”, and “How can I check my immigration case status?”. The complete query set and experimental splits are documented in the repository.

C.3. Poisoning Scenarios

Robustness experiments evaluate two adversarial conditions. Type A attacks inject random non-semantic noise vectors. Type B attacks introduce semantically targeted misleading vectors intended to bias retrieval outcomes. Malicious participation fractions of 0 %, 10 %, and 20 % were evaluated. Recovery performance was measured over subsequent consensus cycles to quantify resilience and trust stabilization dynamics.