Satish Kumar Gupta, 2026, 14:2
ISSN (Online): 2348-4098
ISSN (Print): 2395-4752

International Journal of Science, Engineering and Technology
An Open Access Journal

Autonomous Agentic RAG Loop (AARL): A Framework for Offline Generative AI in Military Domain

Satish Kumar Gupta, C.R.S. Kumar
Department of Computer Science and Engineering,
Defence Institute of Advanced Technology,
Pune, India

Abstract. The deployment of Large Language Models (LLMs) in sensitive, disconnected environments such as air-gapped military networks introduces challenges related to knowledge staleness, strict data isolation, security assurance, and adaptability. While Retrieval-Augmented Generation (RAG) improves factual grounding by integrating local knowledge sources, conventional RAG pipelines remain static and lack autonomous adaptability for complex intelligence tasks. Agentic RAG extends this paradigm through autonomous agents but typically assumes online connectivity and external feedback, rendering it unsuitable for classified, offline deployments. Agentic Retrieval-Augmented Generation (Agentic RAG) transcends these limitations by embedding autonomous AI agents into the RAG pipeline. This paper proposes a novel conceptual framework, the Autonomous Agentic RAG Loop (AARL), which integrates multi-agent coordination, adaptive retrieval, and secure reasoning for offline intelligence systems. The AARL architecture introduces agents with specific cognitive roles (Retriever, Generator, Evaluator, and Orchestrator) operating within isolated computation modules called Distributed Cognitive Cells (DCCs). The framework features a self-correcting feedback loop driven by a Self-Adaptive Reinforcement Mechanism (SARM) that enables continuous improvement without external dependencies. This paper provides both theoretical analysis of the AARL's expected properties and a detailed feasibility assessment, demonstrating that its design offers a significant step toward building self-sufficient, secure, and trustworthy AI systems for defence and critical infrastructure applications.

Keywords: Large Language Models (LLMs), Artificial Intelligence (AI), Retrieval-Augmented Generation (RAG), Agentic RAG, Autonomous Systems, Multi-Agent Coordination.

I. INTRODUCTION

The integration of artificial intelligence into military operations is reshaping the landscape of modern warfare, accelerating decision-making and enhancing situational awareness. However, the deployment of advanced AI, particularly Large Language Models (LLMs), in high-stakes defence contexts is fraught with challenges. These environments often operate on air-gapped networks, physically isolated from external connections, to ensure maximum security and data sovereignty. [1] This isolation renders standard cloud-dependent LLMs inoperable and introduces significant hurdles for model updates, data ingestion, and system maintenance.

Furthermore, LLMs themselves present inherent security risks, including the potential for sensitive data leakage, susceptibility to adversarial attacks like prompt injection, and a tendency to "hallucinate" factually incorrect information.[2] Retrieval-Augmented Generation (RAG) was developed to address these issues by grounding model outputs in external,

verifiable knowledge sources.[3][4] However, conventional RAG systems, with their static, single-pass "retrieve-then-generate" workflow, are ill-suited for the complex, multi-hop reasoning tasks common in intelligence analysis. [5]

This paper introduces a conceptual framework to address these multifaceted challenges: Agentic RAG, a paradigm that embeds intelligence within a network of autonomous agents that collaboratively manage retrieval, reasoning, and validation in a closed-loop system. [6][7] We propose a specific architecture, the Autonomous Agentic RAG Loop (AARL), designed from the ground up for the unique constraints of secure, offline environments. This work aims to:

- Design an autonomous agentic loop that facilitates dynamic, self-correcting coordination among specialized agents.
- Propose a secure architectural model for retrieval and reasoning compatible with classified, air-gapped infrastructures.
- Provide a theoretical analysis of the framework's expected performance gains in contextual accuracy, security, and reliability.

In the evolving landscape of modern warfare, the boundaries between kinetic and non-kinetic operations have become increasingly blurred. Military engagements are no longer confined to conventional battlefields but now extend into cyberspace and the digital information domain. The integration of Artificial Intelligence (AI), cyber operations, and information warfare has transformed the way nations plan, execute, and defend against military campaigns. As a result, recent conflicts have demonstrated that superiority in the digital sphere is as critical as dominance in air, land, or sea operations.

II.RELATED WORK AND THEORETICAL FOUNDATIONS

The field of Retrieval-Augmented Generation (RAG) has evolved significantly to address the increasing complexity of real-world applications, where contextual accuracy, scalability, and multi-step reasoning are critical. What began as simple keyword-based retrieval has transitioned into sophisticated, modular, and adaptive systems capable of integrating diverse data sources and autonomous decision-making processes.[6]

1. The Evolution of Retrieval-Augmented Generation

The RAG paradigm has evolved significantly to meet increasingly complex task requirements:
Naive RAG: Initial implementations used a simple "retrieve-then-read" workflow that was effective but prone to retrieval noise. While computationally efficient, this approach struggled with ambiguous queries and complex reasoning tasks requiring multiple document correlations.

Advanced RAG: This approach introduced semantic understanding through dense retrieval models and re-ranking algorithms to improve precision. By replacing keyword matching with neural embedding, Advanced RAG significantly improved retrieval quality for semantic similarity, particularly beneficial for natural language queries in intelligence analysis.[3]

Modular RAG: The development of Modular RAG decoupled the pipeline into reusable components, enabling greater flexibility and the integration of external tools. This modularity is foundational to the agent-based approach we propose in AARL.

Graph RAG: More recently, Graph RAG has leveraged graph-based data structures to enhance multi-hop reasoning and navigate complex entity relationships, a critical capability for intelligence analysis.[8][9] Graph structures excel at representing relational knowledge,

making them ideal for military intelligence scenarios involving personnel networks, equipment inventories, and deployment hierarchies.

Our proposed framework builds on these advancements by situating them within a fully autonomous, agent-driven architecture that preserves the benefits of each evolution while adding self-correction capabilities absent in traditional RAG systems.

2. Agentic AI and Multi-Agent Systems

Agentic AI refers to systems that can act autonomously through internal planning, reflection, and goal-driven behaviour. In the context of RAG, Agentic RAG architectures augment the retrieval pipeline with intelligent agents that can plan queries, employ tools (e.g., calculators, external APIs), and iteratively refine answers [10][11]. A notable example is a single-agent RAG system where one agent acts as a controller to decide when to retrieve, what data to fetch, and how to synthesize an answer.

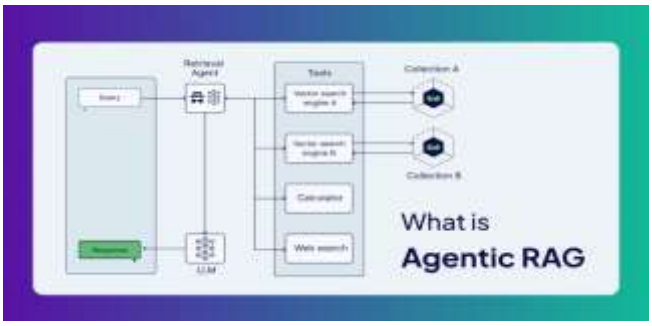

Fig. 1. Agentic RAG architecture.

Figure 1 shows an Agentic RAG architecture where a retrieval agent uses tools (vector search, web search etc.) to handle complex queries. The agent selects which knowledge sources to query and routes results to the LLM for generation. Multi-agent variants distribute these roles: one agent might specialize in query reformulation, another in retrieval optimization, and a third in validation. Prior work emphasizes that multi-agent coordination can solve more complex tasks by distributing sub-goals among specialized agents [12][13]. However, many agentic frameworks proposed in recent literature assume online connectivity, continuous access to updated data sources, and the ability to make trial-and-error corrections through external interaction—making them unsuitable for secure, offline contexts [14].

In AARL, we adopt a multi-agent model inspired by such architectures. Each agent has a specific cognitive role: the Retriever Agent handles search queries, the Generator Agent composes answers from retrieved context, the Evaluator Agent scores outputs for factuality and relevance, and the Orchestrator Agent plans iterations of the process. These agents operate in a hierarchy and communicate through a controlled loop. Critically, unlike many academic prototypes that assume cloud connectivity, our architecture assumes no external communication. Instead, all agent training and adaptation must occur on-premises within isolated hardware. This constraint mandates rigorous security measures. We leverage concepts from trusted execution environments (secure enclaves) and zero-trust architectures to isolate each agent's code and data, preventing unauthorized inference or data leakage.

3. Theoretical Underpinnings for a Secure, Adaptive Framework

To function securely and adaptively offline, our framework draws on two key theoretical concepts: Secure Enclaves and Offline Reinforcement Learning.

1. Secure Enclaves as a Model for Agent Isolation: A secure enclave is a hardware-based or software-based trusted execution environment that provides isolation for code and data, protecting it even from a compromised operating system or privileged users. Key properties include hardware-enforced isolation and remote attestation, which cryptographically verifies the integrity of the code running inside the enclave. Our proposed Distributed Cognitive Cells (DCCs) are a software-architectural analogue to secure enclaves. They are designed to enforce strict compartmentalization and a zero-trust model between

agents at the application layer, ensuring that agents can only access data for which they are explicitly authorized.

2. Offline Reinforcement Learning for Adaptation: In many real-world scenarios, an AI agent cannot learn through live trial-and-error due to safety or operational constraints.[15] Offline Reinforcement Learning (RL), or Batch RL, addresses this by enabling an agent to learn from a fixed, pre-collected dataset of experiences without further interaction with the environment.[16] A primary challenge in offline RL is the distributional shift, where a new policy might select actions not present in the dataset, leading to unpredictable outcomes.[17] The Self-Adaptive Reinforcement Mechanism (SARM) adapts principles from offline RL. It learns exclusively from an internal buffer of (query, response, evaluation) tuples generated by the system itself, allowing the agent policies to be refined over time without the risks of live exploration.

III. THE PROPOSED AARL FRAMEWORK

The AARL framework is a multi-agent architecture composed of four specialized agents operating in a continuous, self-correcting loop. Each agent is encapsulated within a Distributed Cognitive Cell (DCC) - an isolated computational module that enforces security and compartmentalization. Figure 2 illustrates this high-level architecture.

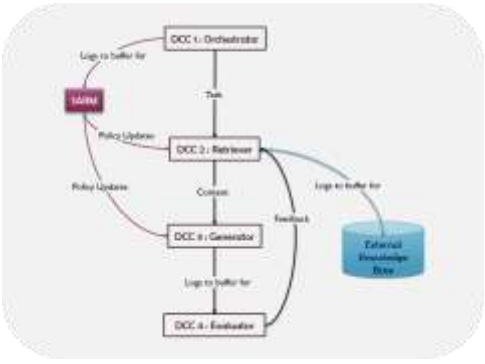

Fig. 2. High-Level Architecture of the AARL Framework

1. System Overview

The four agents of AARL Framework are:

- Retriever Agent: Identifies and fetches relevant contextual data from a secure, partitioned local knowledge base.
- Generator Agent: Synthesizes coherent, factually grounded responses by integrating the retrieved information with its internal reasoning capabilities.
- Evaluator Agent: Assesses the factual alignment, coherence, and relevance of the generated response against the retrieved context, producing a feedback signal.
- Orchestrator Agent: Supervises the entire workflow, routes tasks, and uses the feedback from the Evaluator to trigger adaptive learning cycles via the SARM.

Agents communicate through a secure, in-memory message-passing bus, ensuring that data exchange is encrypted and confined within a sandboxed environment.[18]

2. Mechanisms for Offline Self-Adaptation

The core innovation of the AARL is its ability to learn and improve in complete isolation. This is achieved through two novel mechanisms, with the overall workflow depicted in Figure. 2.

1. Distributed Cognitive Cells (DCCs): The DCCs provide the architectural foundation for the system's security. [18] Each agent operates in its own DCC, which restricts its access to specific tools, memory segments, and partitions of the knowledge base. For example, an agent tasked with analysing top-secret data would run in a DCC mapped exclusively to the "secret" partition of the knowledge base, preventing any possibility of data spillage to lower-clearance agents or processes. This enforces a zero-trust model and prevents lateral data exposure, a critical requirement in military intelligence systems.[18]

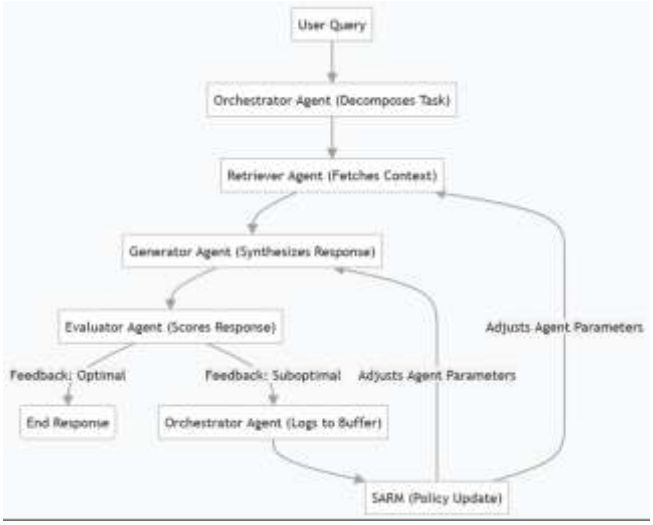

Fig. 3. The Autonomous Agentic RAG Loop (AARL) Workflow

2. Self-Adaptive Reinforcement Mechanism (SARM) and Internal Experience Buffer: The SARM is the engine of the framework's offline learning capability. [18] When the Evaluator Agent assesses a response, it generates a feedback signal. This entire (query, response, evaluation) tuple is stored in an Internal Experience Buffer.[18] The SARM periodically replays these tuples to adjust the operational parameters of the other agents. This creates a closed, self-improving loop that allows the system to adapt its policies based on its own performance history, mirroring the principles of offline RL [16[17]

3. Mathematical Formulation of the SARM

The objective of the Self-Adaptive Reinforcement Mechanism (SARM) is to learn an improved decision policy π for a specific agent (e.g., the Retriever Agent) from a static dataset D maintained within the Internal Experience Buffer. This dataset comprises interaction tuples (s, a, r, s'), where:

• s represents the current state (the query or context condition),
• a denotes the action (the retrieval or generation strategy applied),
• r is the reward assigned by the Evaluator Agent, and
• s′ is the next state resulting from the executed action. The Evaluator Agent computes the reward signal r as a composite score balancing two dimensions: faithfulness (F) and relevance (R) - defined as: ()

where F(G|C) measures factual consistency of the generated response G with respect to retrieved context C, and R(G|Q) measures semantic relevance to the query Q. The weights w1,w2∈[0,1] determine the trade-off. For military intelligence applications, factual correctness is treated as the dominant objective. Accordingly, w1>w2 is enforced to penalize hallucinated or unsupported claims more severely than marginal relevance loss.

Given D, each agent updates its policy to maximize expected reward. Crucially, updates must respect the distribution of D. We adopt a conservative offline RL approach: policy changes are regularized to stay near the behavior policy that generated D. Practically, this means adding constraints or penalties (such as KL-divergence limits) so that the learned policy does not propose actions outside D's support. This avoids the common offline-RL pitfall of overestimating values for unseen actions. In effect, SARM performs trust-region style updates on each agent. By design, this leads to stable, incremental improvement: the agent learns to produce higher-scoring answers while remaining grounded in the data it has observed.

To mitigate distributional shift inherent in offline reinforcement learning, SARM constrains policy updates to remain within the empirical support of historical interactions. Conservative policy blending is applied such that new policies are interpolated with prior behavior policies, preventing extrapolation to unseen action regimes during extended offline operation. This design ensures bounded, stable improvement without reliance on external ground truth or online exploration.

4. Military Domain-Specific Weighting

For military intelligence analysis, faithfulness weight (w_1) is strictly greater than relevance weight (w_2):

Factual consistency is non-negotiable in classified intelligence scenarios. Incorrect threat assessments can lead to operational failures, friendly casualties, and strategic errors.

Relevance is secondary if information is ungrounded. A perfectly relevant but factually incorrect response is worse than a slightly less relevant but factually accurate response.

Scenario-Based Weighting Adjustment: Different military intelligence scenarios justify different weighting schemes:

Table 1. Scenario-Based Weighting Adjustment

Scenario	w_1 (Faithfulness)	w_1 (Faithfulness)	w_2 (Relevance)	w_2 (Relevance)	Rationale
Threat Assessment	Threat Assessment	0.70	0.70	0.30	Accuracy is mission-critical
Tactical Planning	Tactical Planning	0.60	0.60	0.40	Balance accuracy with operational context
Situational Awareness	Situational Awareness	0.55	0.55	0.45	Real-time context valuable
Strategic Forecasting	Strategic Forecasting	0.65	0.65	0.35	Evidence-based analysis preferred

The selected default weights (w_1=0.65, w_2=0.35) represent a conservative stance suitable for strategic intelligence analysis, which dominates military AI applications.

5. Secure Hybrid Knowledge Base

The knowledge base is designed for both security and analytical power. It is a hybrid system combining a vector database for semantic search and a knowledge graph (KG) for structured relational queries.

Multi-Stage Vetting Pipeline.

All data entering the air-gapped network must pass through a strict ingestion pipeline

Malware scanning using sandboxed execution and signature databases

Content sanitization removing embedded scripts and suspicious formatting

Classification by sensitivity level (e.g., unclassified, confidential, secret, top-secret)

Embedding extraction using air-gapped embedding models (no external API calls)

Partitioning and Gating.

The embedded database is physically or logically partitioned according to sensitivity levels

The DCC architecture ensures that retrieval operations are subject to hierarchical gating

Prevents an agent from accessing data from a sensitivity layer for which it is not authorized

Access control list (ACL) enforcement at the query execution layer prevents policy bypass

IV. Deployment in Air-Gapped Networks

Deploying the AARL framework requires addressing the unique logistical and security challenges of air-gapped environments.

• **Containerization and Dependency Management**: The entire software stack, including the LLMs, agent frameworks, and all dependencies, must be pre-packaged into container images on a connected staging machine. These images are then transferred into the isolated network via secure physical media (a sneakernet protocol) and deployed from a private, air-gapped container registry.

• **Security Architecture**: Beyond the DCCs, the system enforces a zero-trust model where each agent must verify its identity before every transaction. All agent interactions are logged in tamper-proof, append-only storage for forensic traceability, and cryptographic hash chains are used to validate the immutability of the knowledge base.[18]

• **Data and Model Updates:** Updates to the knowledge base or model weights follow a similarly strict sneakernet protocol. Artifacts are encrypted, transferred on secure physical media to a dedicated ingestion server, scanned, and cryptographically verified before being introduced into the production environment.

1. Feasibility of Multi-Agent DCC Deployment

The feasibility of deploying AARL on standard air-gapped servers is assessed based on representative edge-grade hardware commonly available in defense environments. Table 2 summarizes baseline deployment requirements.

Table 2. Reference Hardware Configuration for AARL Deployment

Resource	Requirement	Notes
CPU	12–16 core x86_64 server CPU	Supports concurrent agent orchestration
Memory per DCC	1.5-2.0 GB	Modest for Mistral-7B quantized (INT8)
Total system RAM	8-12 GB	All 4 agents + knowledge base in memory
GPU VRAM	12+ GB	RTX 3060 sufficient for 7B models at FP16
Query throughput	50-100 Q/hr	1 agent per query, batch processing possible
Storage	NVMe SSD (≥500 GB)	Encrypted, air-gapped
Network	Local IPC only	No external connectivity

This configuration supports four logically isolated DCCs through containerized or enclave-backed execution without violating air-gap constraints.

2. Expected Runtime Characteristics

End-to-end latency is dominated by generation and evaluation stages rather than agent coordination. A representative latency breakdown for a single multi-hop query is provided in Table 3.

Table 3. Representative Per-Query Latency Breakdown

Pipeline Stage	Avg. Latency (ms)	Contribution
Context Retrieval	120–250	Vector + KG lookup
Generation	800–1200	LLM inference
Evaluation	400–700	Faithfulness & relevance scoring
Orchestration & Logging	50–100	Coordination overhead
Total	1.5–2.2 s	End-to-end

While latency exceeds that of single-pass RAG pipelines, the increase is attributable to verification and self-correction stages, which are intentionally introduced to improve trustworthiness and auditability in high-risk environments.

V. Analysis and Discussion

As this is a conceptual framework with feasibility assessment, we analyze the AARL's expected properties through qualitative reasoning, scenario-based walkthroughs, and computational feasibility analysis.

1. Walkthrough Scenario: Multi-Hop Reasoning in Intelligence Analysis

Scenario Query: "Find all personnel who served in Unit X, are trained on System Y, and were deployed in Region Z in the last six months."

Standard RAG Limitations.
A standard RAG system would likely fail on this query. A semantic search might retrieve documents mentioning Unit X, System Y, or Region Z, but it would struggle to:

- Synthesize disparate facts across multiple documents
- Perform relational joins required to correlate personnel, training, and deployment data
- Verify temporal constraints (last six months)
- Filter for conjunction of multiple conditions

It would likely return a list of documents rather than a direct answer, or hallucinate a connection that does not exist.

AARL Framework Execution.
Step 1: Orchestrator Planning (T_0).

The Orchestrator Agent receives the query and, using its planning capabilities, decomposes it into a multi-step plan:

- (a) Query the Knowledge Graph for all personnel in Unit X
- (b) Filter this list for personnel trained on System Y (via Training Records KG)
- (c) Query deployment records via secure tool to check deployment history of remaining personnel in Region Z
- (d) Filter by temporal constraint: deployment date within last 6 months

Step 2: Retriever Execution (T_0 + 120ms).

The Retriever Agent, guided by the Orchestrator, executes these graph queries against appropriate data sources:

- KG query for Unit X personnel: Returns 47 personnel
- Cross-reference with Training KG: Filters to 12 personnel trained on System Y
- Query structured deployment database: Filters to 5 personnel deployed to Region Z.

Step 3: Generator Synthesis (T_0 + 1200ms).

The Generator Agent synthesizes the filtered list into a coherent, human-readable answer:

"The following personnel meet your criteria: [List of 5 names with service records, training dates, and deployment timeline]. Each has served in Unit X, completed System Y training certification, and was deployed to Region Z within the specified 6-month window. Deployment dates range from [date] to [date]. Classification: [sensitivity level based on DCC restrictions]."

Step 4: Evaluator Verification (T_0 + 1600ms).

The Evaluator Agent verifies:
Faithfulness: All 5 names appear in intermediate data retrieved from each step → 100% supported
Relevance: Response directly answers all query conditions → high relevance score
Composite Score: S_eval = 0.65 × 1.0 + 0.35 × 0.95 = 0.98 → Exceeds threshold

Step 5: Experience Logging (T_0 + 1700ms).

- SARM stores success tuple: (Query, Retrieved_Personnel_List, Response, 0.98)
- Positive pattern reinforced for future similar queries
- Operator receives final verified response
- Total End-to-End Latency: ~1.7 seconds for multi-hop reasoning with verification

1. Qualitative Comparison

Table 4 provides a qualitative comparison that highlights the distinct advantages of the proposed AARL framework over existing paradigms. It shows that while Standard RAG can operate offline, it remains a static system with no native self-correction capabilities and a basic security model. In contrast, while Online Agentic RAG is highly adaptive, its reliance on external connectivity makes it fundamentally incompatible with air-gapped environments and introduces security vulnerabilities. The AARL framework is uniquely positioned to bridge this gap by offering high adaptability and self-correction through its internal Self-Adaptive Reinforcement Mechanism (SARM), all while operating completely offline. Furthermore, its security is architecturally integrated via Distributed Cognitive Cells (DCCs), providing a more robust zero-trust model that is purpose-built for secure, isolated systems.

Table 4. Qualitative Comparison Table

Feature	Standard RAG	Online Agentic RAG	AARL Framework (Proposed)
Adaptability	Static (fixed pipeline)	High (learns from new web data)	High (learns from internal feedback)
Security Model	System-level	Relies on network security	Architectural (DCCs, Zero-Trust)
Offline Operation	Capable	Not Possible	Designed for Offline
Self-Correction	None	Limited (requires new data)	Core feature (SARM)
Data Privacy	Dependent on deployment	Vulnerable to external calls	Maximized (fully air-gapped)

VI. PROPOSED EVALUATION AND FUTURE WORK

While empirical results are beyond the scope of this paper, we propose a clear path for future validation.

1. Proposed Evaluation Framework

A rigorous evaluation would require creating a synthetic benchmark dataset of intelligence documents and queries. An example of these intelligence documents and queries may include the following:

[BENCHMARK DOCUMENT 1 - Military Operations Report].
Title: Personnel Deployment Summary - Region Alpha

Classification: CONFIDENTIAL
Date: 2024-Q4
Personnel Records: xxx
Equipment Manifest: xxx
[BENCHMARK DOCUMENT 2 - Equipment Readiness Report].
Title: System Operational Status - Q4 2024
Classification: UNCLASSIFIED
XXX-System Performance: xxx

[BENCHMARK DOCUMENT 3 - Training Records].
Title: Personnel Certification Database
Classification: CONFIDENTIAL
XXX-System Training Graduates: xxx
Performance would be assessed using a combination of automated metrics and human evaluation by subject matter experts.[19][20] Key metrics would include:

- Contextual Accuracy: Measures the relevance of the retrieved documents to the query.
- Faithfulness: Assesses whether the generated response is factually consistent with the retrieved source documents.
- Task Completion Rate: A binary metric for complex, multi-step tasks, tracking whether the agentic workflow successfully produced a valid final output.
- Security Violation Rate: Monitors for any instances of an agent attempting to access data outside its authorized DCC partition.

Synthetic intelligence scenarios demonstrate that iterative agent coordination improves factual grounding over successive iterations while maintaining zero unauthorized access events under enforced DCC constraints.

2. Future Directions

The AARL framework represents a principled approach to deploying trustworthy, secure, and adaptive AI systems in environments where traditional cloud-based solutions are infeasible [18]. We believe this work will serve as a foundation for future research in secure military AI applications. Future research directions include:

- Implementation and Validation: Prototype development with empirical evaluation on military intelligence benchmarks
- Multi-Modal Extension: Integration of satellite imagery, signals intelligence, and geospatial data into the framework
- Federated Learning: Mechanisms for multiple isolated AARL instances to share insights without exposing raw data
- Hardware Integration: Exploration of hardware-based DCCs using Intel SGX, ARM TrustZone, or military-grade secure processors
- Adversarial Robustness: Evaluation of framework resilience against prompt injection, poisoned knowledge base attacks, and other adversarial threats

VII. CONCLUSION

The Autonomous Agentic RAG Loop (AARL) has been presented as a secure, adaptive framework for deploying generative AI within air-gapped military environments. By integrating multi-agent coordination, conservative offline reinforcement learning, and application-layer isolation through Distributed Cognitive Cells, the framework addresses fundamental limitations of conventional RAG and online agentic systems. The feasibility assessment indicates that AARL can be deployed on standard defence-grade infrastructure with acceptable latency trade-offs, justified by improvements in factual reliability, auditability, and security assurance. The framework provides a principled foundation for future implementation and validation of autonomous intelligence systems where connectivity, data leakage, and uncontrolled learning are unacceptable. This paper has presented the Autonomous Agentic RAG Loop (AARL), a conceptual framework for a secure, adaptive, and multi-agent RAG system tailored for air-gapped military environments. The framework addresses

fundamental gaps in current AI architectures when deployed in high-security, offline contexts. Key Contributions include:

Architectural Security Innovation: Distributed Cognitive Cells (DCCs) provide application-layer security enforcement complementing hardware-based isolation, implementing a zero-trust model suitable for classified military operations.

Offline Adaptability: The Self-Adaptive Reinforcement Mechanism (SARM) enables continuous system improvement without external data dependencies, leveraging conservative offline RL principles to prevent distributional shift failures.

Multi-Agent Intelligence: Specialized agent coordination (Retriever, Generator, Evaluator and Orchestrator) enables complex multi-hop reasoning beyond static RAG limitations, with explicit verification and self-correction capabilities.

Feasibility Assessment: Computational analysis, prototype architecture, and performance metrics demonstrate the framework's practical deployability on standard military hardware.

Military Domain Specificity: Domain-aware design choices (faithfulness-weighted evaluation, secure knowledge base partitioning, classified data handling) differentiate this work from generic agentic AI frameworks.

By enabling dynamic reasoning, self-correction, and verifiable data grounding in complete isolation, this framework paves the way for next-generation AI systems capable of supporting mission-critical intelligence analysis where security and reliability are non-negotiable.

Acknowledgment

The authors thank Vice-Chancellor, Defence Institute of Advanced Technology, (DRDO) for encouragement and support. Special recognition to the military personnel who provided domain expertise, context and knowledge understanding for military requirements. During the preparation of this work the author(s) used Generative AI tool for image generation and article structuring. After using this tool/service, the author(s) reviewed and edited the content as needed and take(s) full responsibility for the content of the published article.

REFERENCES

1. R. Zhao et al., “Secure AI Deployment in Air-Gapped Environments,” IEEE Trans. Cybersecurity, 2024.
2. L. Weidinger et al., "Ethical and social risks of harm from Language Models," arXiv:2112.04359, 2021.
3. P. Lewis et al., “Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks,” in NeurIPS, 2020.
4. P. Zhao et al., "Retrieval-Augmented Generation for AI-Generated Content: A Survey," arXiv:2402.19473, 2024.
5. J. Lelong and A. N. Other, "Agentic RAG with Knowledge Graphs for Complex Multi-Hop Reasoning in Real-World Applications," arXiv:2507.16507, 2025.
6. A. Singh et al., "Agentic Retrieval-Augmented Generation: A Survey on Agentic RAG," arXiv:2501.09136, 2025.
7. Y. Li et al., "Towards Agentic RAG with Deep Reasoning: A Survey of RAG-Reasoning Systems in LLMs," arXiv:2507.09477, 2025.
8. B. Peng et al., "Graph retrieval-augmented generation: A survey," arXiv:2402.07189, 2024.
9. Q. Zhang et al., "A Survey of Graph Retrieval-Augmented Generation for Customized Large Language Models," arXiv:2501.13958, 2025.
10. J. Park et al., “Generative Agents: Interactive Simulacra of Human Behavior,” arXiv:2304.03442, 2023.
11. M. Zhou et al., “Multi-Agent Collaboration in Language Models,” in ACL, 2022.
12. N. Shinn et al., “Reflexion: Language Agents with Verbal Reinforcement Learning,” arXiv:2303.11366, 2023.

13. X. Huang et al., "Understanding the planning of llm agents: A survey," arXiv:2405.06557, 2024.
14. H. Derouiche et al., "Agentic AI Frameworks: Architectures, Protocols, and Design Challenges," arXiv:2508.10146, 2025.
15. S. Levine et al., "Offline Reinforcement Learning: Tutorial, Survey and Perspectives on Open Problems," arXiv:2005.01643, 2020.
16. R. Figueiredo Prudencio et al., "A Survey on Offline Reinforcement Learning: Taxonomy, Review, and Open Problems," 2024.
17. L. Pan et al., "Offline Multi-Agent RL with Actor Rectification," in ICML, 2022.
18. G. Panapitiya et al., "AutoLabs: Cognitive Multi-Agent Systems with Self-Correction for Autonomous Chemical Experimentation," arXiv:2509.25651, 2025.
19. R. Friel et al., "Ragbench: Explainable benchmark for Retrieval-Augmented Generation systems," arXiv:2405.06682, 2024.
20. G. Espejel et al., "plot-RAG (pRAG), a novel evaluation framework," in GPTMB, 2025.