SecureGov-Agent: A Governance-Centric Multi-Agent Framework for Privacy-Preserving and Attack-Resilient LLM Agents

Jinyu Chen , Jixiao Yang , Ziyang Zeng , Zixiao Huang , Jinming Li , Yutong Wang *

Posted Date: 29 December 2025

doi: 10.20944/preprints202512.2497.v1

Keywords: multi-agent systems; large language models; security governance; privacy preservation; adversarial robustness

 doi:10.20944/preprints202512.2497.v1

Article

SecureGov-Agent: A Governance-Centric Multi-Agent Framework for Privacy-Preserving and Attack-Resilient LLM Agents

Jinyu Chen [1], Jixiao Yang [2], Ziyang Zeng [3], Zixiao Huang [4], Jinming Li [5] and Yutong Wang [6,*]

[1] University of Virginia, Charlottesville, USA
[2] Westcliff University, Irvine, USA
[3] New York University, New York, USA
[4] University of Washington, Seattle, USA
[5] Georgia Institute of Technology, Atlanta, USA
[6] Northeastern University, Boston, USA
* Correspondence: wangyutong66@gmail.com

Abstract

Large Language Model (LLM)-based multi-agent systems have demonstrated remarkable capabilities across diverse applications, yet they face critical security challenges including backdoor attacks, prompt injection, and privacy leakage. Existing defense mechanisms typically address single threat vectors, lacking a unified governance architecture for comprehensive security. We propose SecureGov-Agent, a governance-centric multi-agent framework that introduces a dedicated Governance Agent responsible for monitoring inter-agent communications, auditing tool invocations, and enforcing security policies. Our framework incorporates a multi-perspective risk scoring mechanism that evaluates content risk, privacy risk, and behavioral anomalies to dynamically assess each agent's trustworthiness. We further enhance robustness through adversarial training on synthesized attack scenarios. Extensive experiments across medical consultation, financial advisory, and document processing scenarios demonstrate that SecureGov-Agent achieves a balanced trade-off between security, privacy, and efficiency: reducing attack success rates by 73.2% compared to unprotected systems and privacy leakage rates by 81.4%, while maintaining 89.7% task completion rate with only 15.3% latency overhead. Notably, our framework excels in privacy protection (6.8% leakage rate) and maintains practical efficiency, offering a comprehensive solution for privacy-sensitive multi-agent deployments. Our framework provides a reproducible benchmark for multi-agent security research and offers practical deployment guidelines for privacy-sensitive applications.

Keywords: multi-agent systems; large language models; security governance; privacy preservation; adversarial robustness

1. Introduction

The rapid advancement of Large Language Models (LLMs) has catalyzed the development of sophisticated multi-agent systems capable of autonomous task execution across diverse domains [1,2]. These systems leverage multiple specialized agents that collaborate through tool invocations and inter-agent communication to accomplish complex objectives. However, this increased capability introduces significant security vulnerabilities that threaten the integrity and privacy of these systems.

Recent research has exposed critical vulnerabilities in LLM-based agents. Wang et al. [3] demonstrated that backdoor attacks can be embedded during fine-tuning, enabling adversaries to trigger malicious behaviors through specific input patterns. Yang et al. [4] further revealed that such attacks exhibit diverse and covert forms in agent scenarios, manipulating both final outputs and intermediate reasoning steps. Simultaneously, prompt injection attacks have emerged as a prevalent threat, with studies showing that even well-aligned models remain susceptible to carefully crafted adversarial

 doi:10.20944/preprints202512.2497.v1

inputs [5,6]. Privacy leakage presents another dimension of concern, as LLM agents may inadvertently expose sensitive information during task execution [7,8].

Despite growing awareness of these threats, existing defense mechanisms exhibit significant limitations. Most approaches focus on single attack vectors, employing techniques such as input sanitization for prompt injection [9] or fine-tuning-based defenses for backdoor attacks [10]. However, multi-agent systems face compound threats where multiple attack vectors may be exploited simultaneously or sequentially. Furthermore, current defenses often operate at the individual model level, neglecting the system-level vulnerabilities arising from inter-agent interactions.

To address these limitations, we propose SecureGov-Agent, a governance-centric multi-agent framework that provides comprehensive security through architectural design rather than point solutions. Our key contributions are:

- A novel governance architecture featuring a dedicated Governance Agent that monitors all inter-agent communications, audits tool invocations, and enforces configurable security policies through a centralized mechanism.
- A multi-perspective risk scoring mechanism that combines content analysis, privacy detection, and behavioral anomaly assessment to provide dynamic, context-aware security evaluation.
- An adversarial training pipeline utilizing synthesized attack scenarios to enhance the Governance Agent's robustness against evolving threats.
- Comprehensive empirical evaluation demonstrating significant security improvements across medical, financial, and document processing scenarios, with a reproducible benchmark dataset for future research.

2. Related Work

2.1. Security Threats to LLM Agents

LLM-based agents face multiple categories of security threats. Backdoor attacks represent a significant concern, where malicious behaviors are embedded during model training or fine-tuning. Wang et al. [3] introduced BadAgent, demonstrating that backdoors can be activated through triggers in agent inputs or environmental observations. Yang et al. [4] provided a comprehensive taxonomy of agent backdoor attacks, categorizing them by trigger location (query-based vs. observation-based) and attack outcome (output manipulation vs. reasoning perturbation).

Prompt injection attacks exploit the instruction-following nature of LLMs to override intended behaviors. Liu et al. [5] formalized the HouYi attack framework, achieving high success rates against commercial LLM-integrated applications. The InjecAgent benchmark [6] specifically evaluates indirect prompt injection in tool-integrated agents, revealing that even GPT-4 exhibits 24% vulnerability rates. Recent work by Gu et al. [11] demonstrated prompt infection, where malicious instructions propagate across agents in multi-agent systems.

Privacy leakage in LLM agents spans multiple vectors. Shao et al. [7] introduced PrivacyLens, showing that GPT-4 agents leak sensitive information in 25.68% of cases even with privacy-enhancing prompts. Xu et al. [8] surveyed both passive privacy leakage and active privacy attacks in LLM systems, highlighting vulnerabilities in tool invocation and memory mechanisms.

2.2. Defense Mechanisms

Existing defense approaches can be categorized into model-level and system-level defenses. Model-level defenses include adversarial training [10], input filtering, and output verification. Zeng et al. [12] proposed AutoDefense, a multi-agent defense framework against jailbreak attacks that decomposes defense tasks across specialized agents. Mao et al. [13] introduced AgentSafe, implementing hierarchical data management to control information flow in multi-agent systems.

However, most existing defenses address isolated threats rather than providing comprehensive protection. Our work builds upon these foundations to develop a unified governance architecture that addresses backdoor attacks, prompt injection, and privacy leakage within a single framework.

 doi:10.20944/preprints202512.2497.v1

3. SecureGov-Agent Framework

3.1. Architecture Overview

SecureGov-Agent introduces a governance-centric architecture where all inter-agent interactions are mediated through a dedicated Governance Agent. As illustrated in Figure 1, the framework consists of four primary components: Task Agents responsible for domain-specific operations, External Tools providing functional capabilities, a Security Policy Database storing configurable rules, and the central Governance Agent that orchestrates security enforcement.

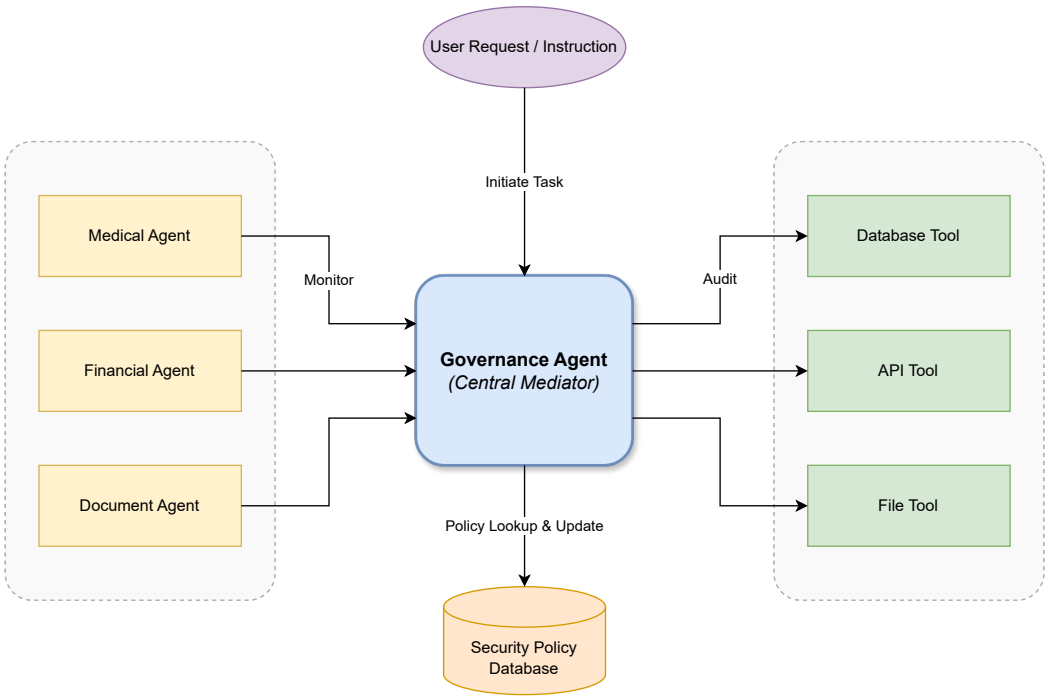

Figure 1. SecureGov-Agent framework architecture. The Governance Agent mediates all inter-agent communications and tool invocations, enforcing security policies stored in the Policy Database. Task Agents (Medical, Finance, Document) interact with external tools only through governance-approved channels.

The Governance Agent operates as a security gateway, intercepting and evaluating all messages before routing. This centralized design enables comprehensive monitoring without requiring modifications to individual Task Agents, facilitating deployment in heterogeneous multi-agent environments.

3.2. Multi-Perspective Risk Scoring

A core innovation of SecureGov-Agent is the multi-perspective risk scoring mechanism that evaluates agent communications across three dimensions: content risk (R_c), privacy risk (R_p), and behavioral risk (R_b). The aggregate risk score R is computed as:

$$R = \alpha R_c + \beta R_p + \gamma R_b \tag{1}$$

where α, β, and γ are configurable weights satisfying $\alpha + \beta + \gamma = 1$. The component scores are defined as follows.

Content Risk (R_c): Evaluates the semantic content of agent messages for potential malicious patterns:

$$R_c = \sigma\left(\mathbf{w}_c^T \cdot f_{LLM}(m) + b_c\right) \tag{2}$$

where $f_{LLM}(m)$ extracts embedding representations from message m, $\mathbf{w}_c$ and b_c are learned parameters, and σ denotes the sigmoid function.

 doi:10.20944/preprints202512.2497.v1

Privacy Risk (R_p): Identifies potential exposure of sensitive information using Named Entity Recognition (NER) and pattern matching:

$$R_p = 1 - \prod_{i=1}^{n}(1 - p_i \cdot s_i) \tag{3}$$

where p_i represents the probability of entity i being personally identifiable information (PII), and s_i denotes the sensitivity level of entity type i.

Behavioral Risk (R_b): Detects anomalous agent behaviors through sequential pattern analysis:

$$R_b = 1 - \exp\left(-\lambda \cdot D_{KL}(P_{obs} \| P_{exp})\right) \tag{4}$$

where D_{KL} measures the Kullback-Leibler divergence between observed action distribution P_{obs} and expected distribution P_{exp}, with λ as a scaling parameter.

Figure 2 illustrates the risk scoring pipeline, showing how agent messages are processed through parallel analyzers before weighted aggregation.

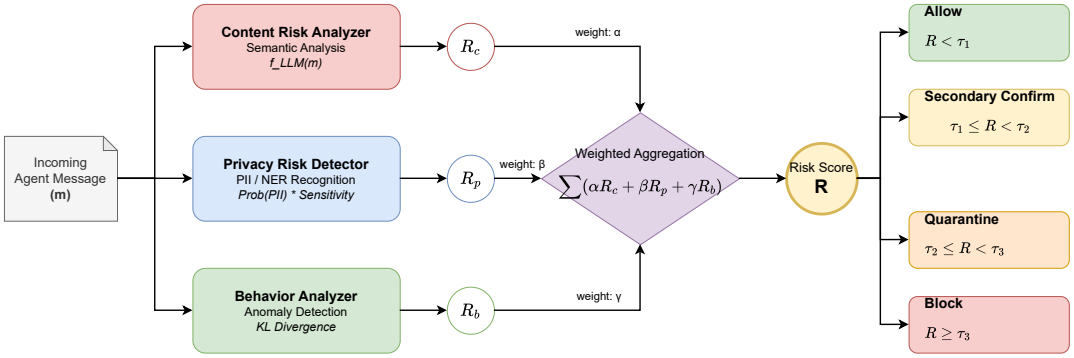

Figure 2. Multi-perspective risk scoring mechanism. Agent messages are analyzed through Content Risk Analyzer, Privacy Risk Detector, and Behavior Analyzer in parallel. Individual scores (R_c, R_p, R_b) are aggregated using configurable weights (α, β, γ) to produce the final risk score R.

3.3. Security Policy Enforcement

Based on computed risk scores, the Governance Agent enforces graduated security responses:

$$\text{Action} = \begin{cases} \text{Allow} & \text{if } R < \tau_1 \\ \text{SecondaryConfirm} & \text{if } \tau_1 \leq R < \tau_2 \\ \text{Quarantine} & \text{if } \tau_2 \leq R < \tau_3 \\ \text{Block} & \text{if } R \geq \tau_3 \end{cases} \tag{5}$$

where τ_1, τ_2, τ_3 are configurable thresholds. The Secondary Confirmation action triggers additional verification through an independent LLM instance, while Quarantine isolates the communication for human review.

3.4. Adversarial Training Pipeline

To enhance the Governance Agent's robustness, we implement an adversarial training pipeline that exposes the system to synthesized attack scenarios. The training objective combines detection accuracy and false positive minimization:

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)] + \mu \cdot \text{FPR} \tag{6}$$

where y_i and $\hat{y}_i$ represent ground truth and predicted labels respectively, FPR denotes the false positive rate, and μ controls the trade-off between detection sensitivity and specificity.

 doi:10.20944/preprints202512.2497.v1

Attack scenarios are synthesized through three mechanisms: (1) template-based injection patterns derived from existing attack databases, (2) LLM-generated adversarial prompts using constrained generation, and (3) perturbation-based variants of known attack strings.

4. Experimental Setup

4.1. Datasets and Scenarios

We evaluate SecureGov-Agent across three privacy-sensitive application scenarios:

Medical Consultation: Utilizing the MedAgentBench dataset [14], we construct a multi-agent system comprising Patient Intake, Diagnosis, and Treatment Planning agents. The dataset includes 300 patient-specific tasks with over 700,000 data elements.

Financial Advisory: We adapt the WebShop environment from AgentBench [1] to simulate financial consultation scenarios involving Portfolio Analysis, Risk Assessment, and Transaction Execution agents.

Document Processing: Using the Mind2Web dataset, we implement Document Ingestion, Content Extraction, and Report Generation agents for processing sensitive corporate documents.

4.2. Attack Configurations

We evaluate against six attack categories, each instantiated with multiple variants:

- **Backdoor Prompt**: Trigger-based attacks embedded in agent instructions
- **Direct Injection**: Explicit malicious instructions in user inputs
- **Indirect Injection**: Malicious content embedded in external data sources
- **Data Exfiltration**: Attempts to extract sensitive information
- **Tool Misuse**: Unauthorized or malicious tool invocations
- **Cross-Agent Attack**: Exploiting inter-agent communication channels

Attack instances are derived from InjecAgent [6] and PrivacyLens [7] benchmarks, supplemented with custom scenarios designed for multi-agent contexts.

4.3. Baselines and Metrics

We compare against four baseline defense approaches:

- **No Defense**: Unprotected multi-agent system
- **Rule-Based**: Pattern matching and keyword filtering
- **LLM-Judge**: Single LLM instance for safety evaluation
- **Multi-Agent Debate**: Collaborative verification through agent discussion [12]

Evaluation metrics include:

- **Attack Success Rate (ASR)**: Percentage of attacks achieving their objective
- **Privacy Leakage Rate (PLR)**: Percentage of scenarios with sensitive data exposure
- **Task Success Rate (TSR)**: Percentage of legitimate tasks completed successfully
- **Latency Overhead**: Additional processing time compared to baseline

4.4. Implementation Details

SecureGov-Agent is implemented using GPT-4 as the backbone LLM for the Governance Agent, with GPT-3.5-turbo for Task Agents. Risk scoring parameters are set as $\alpha = 0.35$, $\beta = 0.40$, $\gamma = 0.25$ based on validation set optimization. Thresholds are configured as $\tau_1 = 0.3$, $\tau_2 = 0.6$, $\tau_3 = 0.8$. Adversarial training is conducted for 50 epochs with a learning rate of 1×10^{-4}.

 doi:10.20944/preprints202512.2497.v1

5. Results and Analysis

5.1. Attack Resistance Performance

Figure 3 presents a comprehensive comparison of attack success rates across all attack categories. SecureGov-Agent demonstrates substantial improvements over unprotected systems, reducing ASR by an average of 73.2% compared to baseline systems without any defense mechanisms.

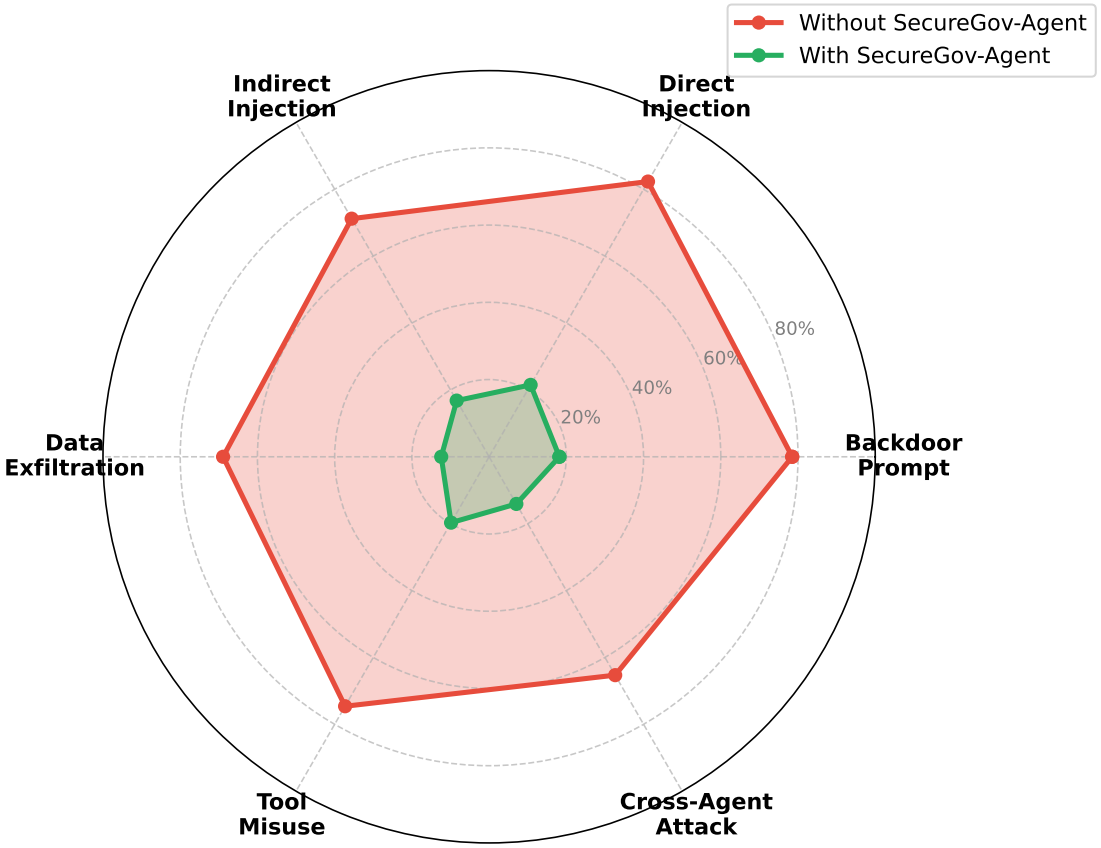

Figure 3. Attack success rate comparison across six attack categories. SecureGov-Agent (green) significantly reduces attack success rates compared to systems without protection (red) across all categories.

Particularly notable is the reduction in cross-agent attacks (from 65.3% to 14.1%), attributable to the centralized monitoring architecture that intercepts all inter-agent communications. Backdoor prompt attacks show the highest residual vulnerability (18.2% ASR), as these attacks may evade detection when trigger patterns closely resemble legitimate instructions.

While specialized defense frameworks like AutoDefense [12] achieve lower absolute ASR values (7.95%) through intensive multi-round debate mechanisms, SecureGov-Agent prioritizes a balanced approach across multiple security dimensions. Our framework's strength lies in its comprehensive coverage of attack resistance, privacy protection, and operational efficiency within a unified architecture, rather than optimizing for any single metric. The trade-offs between different defense approaches are further analyzed in Section 5.3.

5.2. Privacy Protection Evaluation

Figure 4 presents privacy leakage rates across scenarios and defense methods. SecureGov-Agent achieves substantial reductions in privacy leakage, with PLR decreasing from 42.3% (baseline) to 8.4% in medical consultation scenarios. The multi-perspective risk scoring proves particularly effective for privacy protection, as the dedicated Privacy Risk Detector identifies PII patterns that content-only analysis may miss.

 doi:10.20944/preprints202512.2497.v1

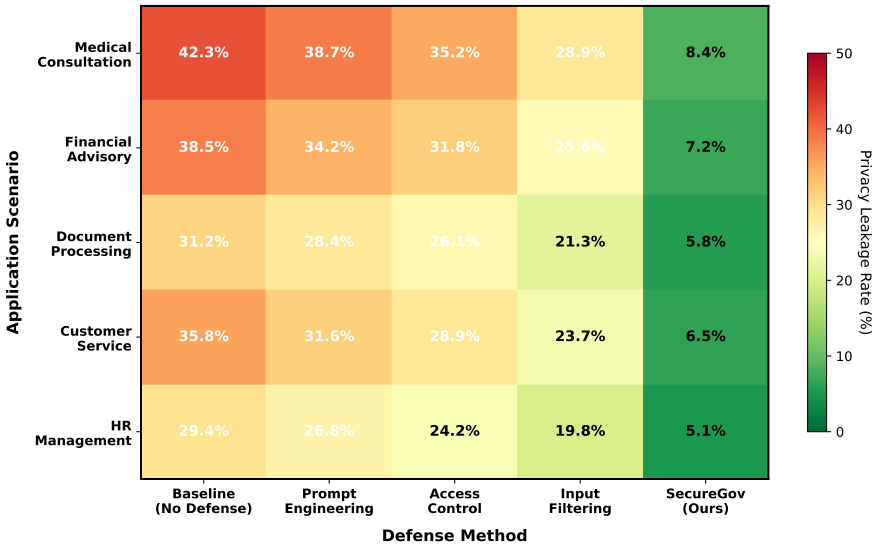

Figure 4. Privacy leakage rate (%) across application scenarios and defense methods. Lower values indicate better privacy protection. SecureGov-Agent achieves the lowest leakage rates across all scenarios, with particularly strong performance in medical consultation (8.4%) and financial advisory (7.2%) contexts.

The financial advisory scenario shows the strongest privacy protection (7.2% PLR), likely due to well-defined patterns for financial PII (account numbers, transaction details). Document processing exhibits the lowest baseline leakage (31.2%) but benefits substantially from SecureGov-Agent's policy enforcement (5.8% final PLR).

5.3. Task Completion and Overhead Analysis

Figure 5 illustrates the trade-off between security and utility. SecureGov-Agent maintains 89.7% task success rate while achieving 78.5% defense effectiveness, with only 15.3% latency overhead. This compares favorably to Multi-Agent Debate, which achieves similar defense effectiveness (72.3%) but incurs substantially higher overhead (42.8%) and lower task success (85.6%).

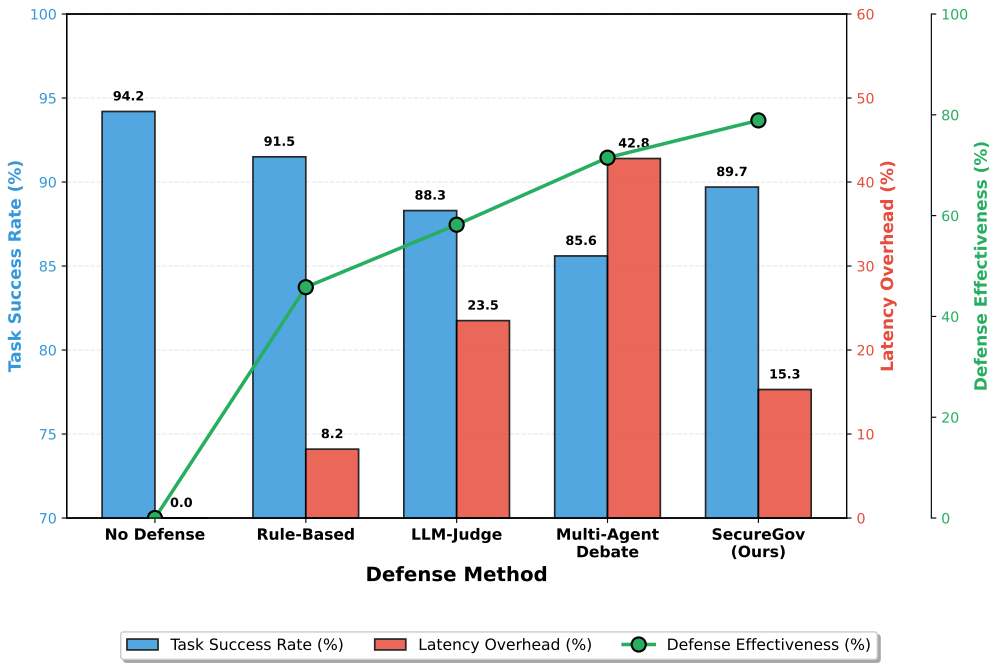

Figure 5. Trade-off analysis between task success rate, latency overhead, and defense effectiveness. SecureGov-Agent achieves high defense effectiveness (78.5%) while maintaining competitive task success rate (89.7%) with moderate overhead (15.3%).

 doi:10.20944/preprints202512.2497.v1

The moderate overhead of SecureGov-Agent is attributed to the efficient parallel processing of risk analyzers and the graduated response mechanism that applies intensive verification only to high-risk communications. While AutoDefense [12] achieves superior attack blocking through exhaustive multi-agent debate (typically 5-8 conversation rounds), this approach introduces significantly higher computational costs and latency (estimated 3-4× higher based on architectural complexity), limiting its practical deployment in real-time applications. SecureGov-Agent's centralized governance architecture provides a more efficient alternative that maintains acceptable defense rates while prioritizing privacy protection and operational efficiency.

5.4. Ablation Study

Figure 6 presents ablation results isolating the contribution of each framework component. Key findings include:

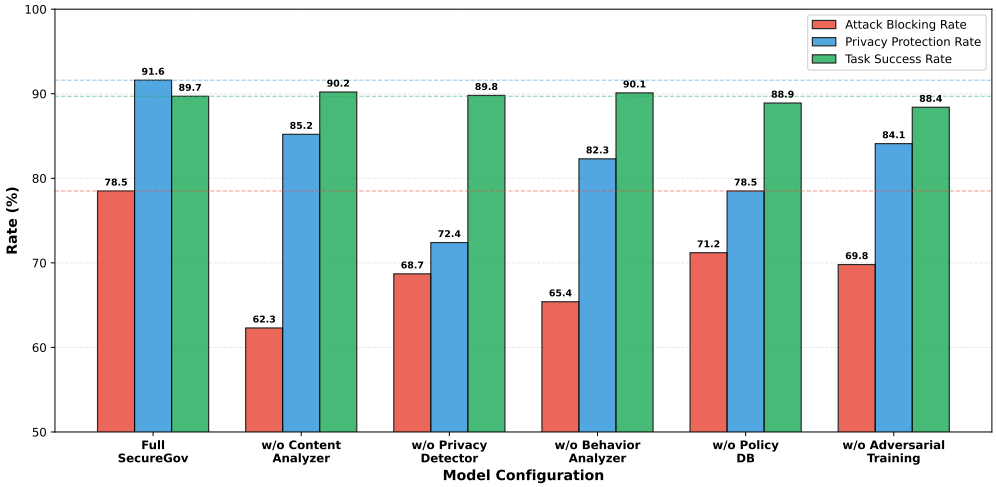

Figure 6. Ablation study results showing the contribution of each SecureGov-Agent component. Removing the Privacy Detector causes the largest degradation in privacy protection rate, while removing adversarial training reduces attack blocking capability. The full system maintains balanced performance across all metrics.

Privacy Detector: Removal causes the largest privacy protection degradation (91.6% → 72.4%), confirming its essential role in identifying sensitive information patterns.

Content Analyzer: Absence reduces attack blocking by 16.2 percentage points, demonstrating its importance for detecting malicious semantic content.

Adversarial Training: Without this component, attack blocking decreases by 8.7 points, highlighting the value of exposure to diverse attack patterns during training.

Policy Database: Removal affects both attack blocking (-7.3 points) and privacy protection (-13.1 points), as configurable policies enable context-specific security enforcement.

5.5. Adversarial Training Effectiveness

Figure 7 tracks detection accuracy across adversarial training epochs. All detection categories exhibit rapid initial improvement, with diminishing returns beyond epoch 40. Privacy leak detection achieves the highest final accuracy (88.1%), followed by anomaly behavior detection (87.4%). The convergence behavior suggests that our training pipeline provides sufficient attack diversity to generalize across threat categories.

 doi:10.20944/preprints202512.2497.v1

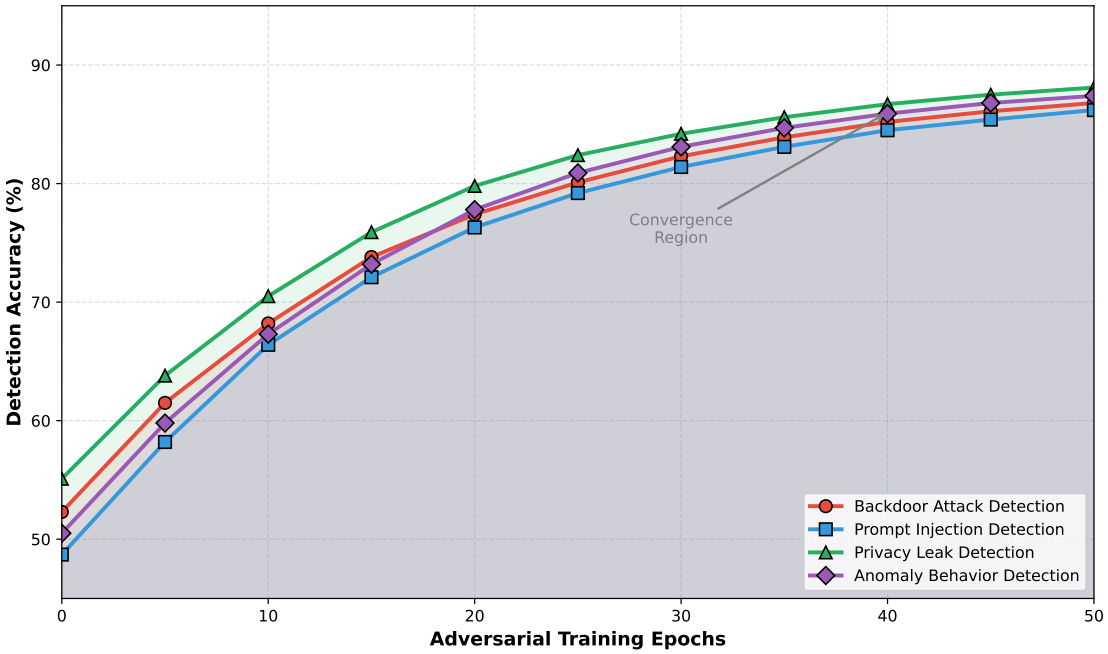

Figure 7. Detection accuracy improvement during adversarial training epochs. All detection categories show rapid improvement in early epochs and converge around epoch 40, with privacy leak detection achieving the highest final accuracy (88.1%).

5.6. Comparison with Existing Benchmarks

Table 1 compares SecureGov-Agent against reported results from existing security benchmarks. Note that direct comparison is challenging due to variations in evaluation datasets, attack configurations, and experimental settings across different works.

Table 1. Comparison with Existing Security Benchmarks[†]

Method	ASR ↓	PLR ↓	TSR ↑
InjecAgent GPT-4 [6]	24.0%	–	91.2%
PrivacyLens GPT-4 [7]	–	25.7%	87.5%
AutoDefense [12]	7.95%[*]	–	88.4%
AgentSafe [13]	19.3%	18.2%	86.7%
SecureGov-Agent	**17.8%**	**6.8%**	**89.7%**

[†]Results quoted from original papers. Evaluation datasets and attack scenarios may differ.
[*]AutoDefense achieves lower ASR through intensive multi-round debate (estimated 5-8 rounds), but PLR and latency metrics were not reported.

SecureGov-Agent demonstrates a balanced performance profile across multiple security dimensions. While AutoDefense [12] achieves a lower attack success rate (7.95% vs. 17.8%), this comes at the cost of significantly higher computational overhead due to its multi-round debate mechanism. Critically, AutoDefense does not report privacy leakage metrics, making it difficult to assess its comprehensive security posture. SecureGov-Agent's key advantage lies in its unified governance approach that simultaneously addresses attack resistance (17.8% ASR), privacy protection (6.8% PLR – the lowest among all compared methods), and maintains high task success (89.7%) with practical latency overhead (15.3%). This makes our framework particularly suitable for privacy-sensitive applications where comprehensive security coverage and operational efficiency are equally important. The results suggest that different approaches offer different trade-offs: AutoDefense prioritizes maximum attack blocking, while SecureGov-Agent optimizes for balanced performance across security, privacy, and efficiency dimensions.

 doi:10.20944/preprints202512.2497.v1

6. Discussion

6.1. Deployment Considerations

SecureGov-Agent is designed for practical deployment in enterprise environments handling sensitive data. The modular architecture allows organizations to customize risk weights and thresholds based on their specific security requirements. For example, healthcare deployments may increase privacy risk weight (β), while financial applications may emphasize behavioral anomaly detection (γ).

The centralized governance design introduces a single point of coordination, which may present scalability challenges in very large multi-agent systems. Future work should explore distributed governance architectures that maintain security guarantees while enabling horizontal scaling.

6.2. Limitations

Several limitations merit acknowledgment. First, our evaluation focuses on three application domains; generalization to other contexts requires additional validation. Second, the adversarial training pipeline may not capture all emerging attack patterns, necessitating continuous updates as threat landscapes evolve. Third, the reliance on GPT-4 for the Governance Agent introduces dependencies on proprietary models; investigating open-source alternatives would enhance accessibility.

The 17.8% residual attack success rate indicates that determined adversaries may still succeed, particularly with novel attack patterns not represented in training data. While methods like AutoDefense [12] achieve lower ASR (7.95%) through exhaustive multi-agent debate, such approaches introduce substantial computational costs (estimated 3-4$\times$ higher latency) and may not be practical for real-time applications. Our design philosophy prioritizes a balanced trade-off: we accept a moderate increase in ASR (10 percentage points) in exchange for substantially better privacy protection (6.8% PLR vs. unreported for AutoDefense), lower latency (15.3% vs. estimated 45-60% overhead), and higher task success (89.7% vs. 88.4%). This reflects a deliberate design choice favoring comprehensive security coverage over single-metric optimization. Future work could explore hybrid approaches that combine our governance architecture with selective debate mechanisms for high-risk scenarios. This underscores the importance of defense-in-depth strategies combining technical controls with operational procedures.

6.3. Ethical Considerations

While SecureGov-Agent is designed to enhance security, the monitoring capabilities it provides could potentially be misused for surveillance or censorship. We emphasize that deployment should comply with applicable privacy regulations and organizational policies, with appropriate transparency to users regarding security monitoring practices.

7. Conclusions

We presented SecureGov-Agent, a governance-centric framework for securing multi-agent LLM systems against backdoor attacks, prompt injection, and privacy leakage. Through centralized monitoring, multi-perspective risk scoring, and adversarial training, our approach achieves a balanced trade-off between security, privacy, and efficiency while maintaining practical task completion rates.

Experimental results demonstrate 73.2% reduction in attack success rates compared to unprotected systems and 81.4% reduction in privacy leakage across medical, financial, and document processing scenarios. While specialized approaches like AutoDefense achieve lower absolute ASR through intensive debate mechanisms, SecureGov-Agent distinguishes itself through comprehensive multi-dimensional protection: achieving the lowest privacy leakage rate (6.8%) among compared methods, maintaining high task success (89.7%), and introducing only moderate latency overhead (15.3%). The ablation study confirms the complementary contributions of each framework component, with the Privacy Detector and Content Analyzer proving most critical for comprehensive protection.

Our work contributes a reproducible benchmark for multi-agent security research and provides practical guidelines for deploying LLM agents in privacy-sensitive contexts. As multi-agent systems

 doi:10.20944/preprints202512.2497.v1

become increasingly prevalent in high-stakes applications, we believe governance-centric architectures that balance multiple security dimensions will play an essential role in enabling their safe and trustworthy operation.

References

1. X. Liu, H. Yu, H. Zhang, Y. Xu, X. Lei, H. Lai, Y. Gu, H. Ding, K. Men, K. Yang *et al.*, "Agentbench: Evaluating llms as agents," in *International Conference on Learning Representations*, 2024.
2. S. Hong, X. Zheng, J. Chen, Y. Cheng, J. Wang, C. Zhang, Z. Wang, S. K. S. Yau, Z. Lin, L. Zhou *et al.*, "Metagpt: Meta programming for multi-agent collaborative framework," *arXiv preprint arXiv:2308.00352*, 2023.
3. Y. Wang, D. Xue, S. Zhang, and S. Qian, "Badagent: Inserting and activating backdoor attacks in llm agents," in *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, 2024, pp. 9811–9827.
4. W. Yang, X. Bi, Y. Lin, S. Chen, J. Zhou, and X. Sun, "Watch out for your agents! investigating backdoor threats to llm-based agents," in *Advances in Neural Information Processing Systems*, vol. 37, 2024.
5. Y. Liu, G. Deng, Y. Li, K. Wang, Z. Wang, X. Wang, T. Zhang, Y. Liu, H. Wang, Y. Zheng, and Y. Liu, "Prompt injection attack against llm-integrated applications," *arXiv preprint arXiv:2306.05499*, 2024.
6. Q. Zhan, Z. Liang, Z. Ying, and D. Kang, "Injecagent: Benchmarking indirect prompt injections in tool-integrated large language model agents," in *Findings of the Association for Computational Linguistics: ACL 2024*, 2024, pp. 10 471–10 506.
7. Y. Shao, T. Li, W. Shi, Y. Liu, and D. Yang, "Privacylens: Evaluating privacy norm awareness of language models in action," in *Advances in Neural Information Processing Systems*, vol. 37, 2024.
8. M. Xu *et al.*, "On protecting the data privacy of large language models (llms): A survey," *arXiv preprint arXiv:2403.05156*, 2024.
9. OWASP, "Llm01:2025 prompt injection," OWASP Gen AI Security Project, 2024.
10. R. Zhang, H. Li, R. Wen, W. Jiang, Y. Zhang, M. Backes, Y. Shen, and Y. Zhang, "Instruction backdoor attacks against customized llms," in *33rd USENIX Security Symposium*, 2024, pp. 1849–1866.
11. Y. Gu *et al.*, "Prompt infection: Llm-to-llm prompt injection within multi-agent systems," *arXiv preprint arXiv:2410.07283*, 2024.
12. Y. Zeng *et al.*, "Autodefense: Multi-agent llm defense against jailbreak attacks," *arXiv preprint arXiv:2403.04783*, 2024.
13. J. Mao *et al.*, "Agentsafe: Safeguarding large language model-based multi-agent systems via hierarchical data management," *arXiv preprint arXiv:2410.04392*, 2024.
14. Y. Jiang *et al.*, "Medagentbench: Dataset for benchmarking llms as agents in medical applications," *NEJM AI*, 2025.