

# Securing Retrieval-Augmented Generation against Data Poisoning and Indirect Prompt Injection

Ravindra Babu Annam  
Hyderabad, Telangana-500090, India.

**Abstract-** RAG enables large language models to give responses that can be enhanced with knowledge extracted from external documents. However, the retrieval of documents exposes the language models to security threats because the information retrieved may be poisoned and used for manipulation of the model. Current security mechanisms are tailored for particular attacks and hence cannot provide end-to-end protection of the entire retrieval to generation process. This study proposes a Secure Retrieval-Augmented Generation Framework (SRAGF) that enables the protection of RAG-based models against data poisoning and prompt injection attacks. It conducts document relevance assessment, source credibility evaluation, semantic anomalies recognition, suspicious pattern recognition, poisoning detection, injection detection, risk classification, context sanitization, knowledge and instruction separation, and output security evaluation. Security scores are calculated for the retrieved documents, which are classified accordingly and included in the context of generation. Documents that pose high risk are stored separately, the ones presenting medium risk undergo sanitization, and those that do not present any threat are retained for context generation. The baseline frameworks for the comparative analysis are Wang-TAD and Neural Cleanse. In the prepared evaluation results, SRAGF exhibits higher detection rates, precision, recall, and F1-score compared to the two baseline frameworks while lowering attack.

**Keywords:** Retrieval-Augmented Generation, Data Poisoning, Indirect Prompt Injection, Secure RAG, Semantic Anomaly Detection, Source Trust, Context Cleaning, Knowledge vs. Instruction Decoupling, Risk-Aware Retrieval, Generative AI.

## I. INTRODUCTION

Retrieval-Augmented Generation has recently come into prominence as one of the significant techniques for increasing knowledge capabilities of large language models through accessing information from external resources in the course of generating responses [1]. In contrast to using solely the knowledge that is encoded in the parameters of the model itself, in RAG systems, relevant documents are retrieved and supplied to the language model as context [2]. The application of such an architecture can be very valuable in many areas, including enterprise knowledge management, technical documentation, question answering, domain assistance, and others, where generated responses have to rely on some external knowledge [3]. Nonetheless, introducing the new source of knowledge also introduces a new security boundary since the retrieved documents are not necessarily reliable and may contain some malicious content, which can affect the behavior of the generation model [4]. The security issue becomes even more relevant when the attacker tries to tamper with the information present in the knowledge bases outside of the language model

itself [5]. Retrieval data poisoning means that an attacker adds false or otherwise misleading information to the knowledge base in order for the information retrieval component to return it eventually to the language model [6]. Contrary to traditional types of attack that target the model or its training process, data retrieval poisoning can affect the information flow without changing the language model itself. If the malicious data receives a high enough retrieval score, it can be used as the basis of the generation process. Malicious artificial intelligence literature clearly shows how hard it is to recognize such attacks [7].

Indirect prompt injection is another significant threat where the malicious instructions are hidden within the documents being retrieved via the RAG system. The instructions can either try to override existing instructions, change the behavior of the model, leak sensitive information, or even affect the final response of the model [8]. Indirect prompt injection stands out as a distinct threat compared to direct prompt injection since the malicious instructions are not necessarily provided by the user. In fact, they are introduced into the model via the

document that is retrieved [9]. There is a key difference between factual information and instructions; documents that are retrieved via a RAG system ought to be viewed as untrusted data while a language model views the instruction-like information in the document as instructions [10] [11]. Past research efforts provide vital building blocks for dealing with adversarial input for intelligent systems. Textual adversarial attack research has shown how slight alterations of words, sentences, or characters may affect the prediction of language models [12] [13]. Thus, anomaly detection and randomization techniques have been studied to identify and mitigate the impact of such adversarial input. Analogously, the backdoor research literature has shown that malicious triggers may be carefully constructed in such a way that they remain hard to detect while affecting the model's behavior [14]. The above literature highlights the necessity for robust detection techniques, but their main focus is not the full security life cycle of externally fetched data in RAG systems [15] [16]. Hence, the security mechanism for RAG systems needs to take into account both the potential presence of textual anomalies and the source's reliability, among others [17].

Yet another problem in this area is the overlap in features between malicious and legitimate documents [18]. It is possible for a legitimate technical document to include commands, instructions, code, or security terms. As a result, simply looking for instructions may cause false alerts and delete valuable pieces of knowledge. In addition, removing the whole document each time a suspicious part is identified may lower the quality of retrieval and limit access to information [19]. There should be a more flexible solution, whereby documents retrieved are classified based on several criteria of security and ranked accordingly [20]. This problem can be solved with the help of the proposed SRAGF framework. Source reliability and provenance become important considerations in the context of secure retrieval as well. Even if two documents have the same content, their sources could be considerably different in terms of reliability. Source authentication, reliability, valid provenance, and consistent updating could therefore give extra ground to determine the trustworthiness of the retrieved information and whether the information should be admitted into the generation context. These characteristics are incorporated into the security processing model of the proposed framework. This means that the decision about the security becomes less content-oriented and more oriented towards the process of information retrieval itself. This kind of approach is

very applicable in the case of enterprises where knowledge could come from various sources.

Semantic coherence is yet another method through which poisoned information can be identified. The malicious document will be relevant to the user's query but at the same time it contradicts any trusted information found in the same knowledge domain. Thus, it becomes evident that relevance alone cannot confirm the safety of any document. Semantic Relevance and Geometric Fusion (SRAGF) compares the retrieved documents to the trusted neighbouring knowledge representations and computes the measure of semantic anomaly. At the same time, SRAGF takes into account contradiction and outlier features while evaluating the probability of poisoning. The framework to be designed has a risk-oriented method of dealing with any threats found. Unlike the traditional technique that considers each threat document the same way, SRAGF uses the level of risk of poisoning and injection in determining the appropriate action. Documents with high risks are quarantined and not included in the generation context, while those with moderate risks are sanitized of their contents. Any instructions, suspicious commands, hidden commands, and other irrelevant information in the documents can hence be either segregated or stripped out of their contents, leaving only the useful facts. The documents are then prioritized based on security and relevance for the construction of the context.

Security protection also goes one step further and is provided for both the process of document retrieval and context construction by means of generating prompts in a protected manner and verifying the output. After building the trusted context, a clear security boundary is drawn around the authorized instruction, the user's query, and the knowledge retrieved in the form of documents that are considered pure data. The language model proceeds to generate the output based on the protected context. Then, the output is evaluated for factual accuracy, supported by trusted sources, for security and for sensitive information leakage. This last stage of verification gives another chance to detect the residual effects, which could not be detected by previous security methods at the document level. In that regard, the first main contribution of this paper is the development of a unified security framework tailored to the security of RAG systems from attacks such as data poisoning and prompt injection attacks. This framework is comprised of source provenance, relevance semantics, anomaly detection, poisoning detection, injection detection, risk-based document

handling, knowledge-instruction separation, context verification, safe generation, and output security validation. In the comparative analysis conducted for this research, Wang-TAD and Neural Cleanse have been selected as benchmark security schemes. The performance comparison factors included in this research are detection, precision, recall, F1 score, success rate of the attack, false positive rate, latency, and secure response. The results prepared for this research show that SRAGF shows better security performance in these.

### 1.1 Hypothesis

- The proposed SRAGF provides enhanced capability for detecting data retrieval attacks relative to the chosen baseline approaches to security.
- The proposed SRAGF provides enhanced capability for detecting and mitigating indirect prompt injection attacks and for reducing the impact of the malicious influence on generated responses.
- The use of the mechanisms such as source trust evaluation, semantic anomaly detection, risk-aware filtering, context sanitization, and output validation improves security and reliability of the RAG systems with increased latency costs.

### 1.2 Objectives of the Research

- To create a secure RAG system that can detect and defend against retrieval-data poisoning and indirect prompt injection attacks.
- To incorporate source reliability check, semantic relevance, semantic anomaly detection, poisoning analysis, and injection detection into one document security system.
- To design a risk-aware document handling technique that can enable acceptance, sanitization, quarantine, and re-retrieval based on security risk assessment.
- To maintain clear demarcation between the retrieved information and instruction-like information to ensure that the documents cannot control the behavior of language models.
- To perform evaluations of SRAGF against relevant baselines in terms of detection accuracy, precision, recall, F1 score, attack success rate, false positive rate, latency, and secure responses.

## II. LITERATURE REVIEW

Shaukat et al. [1] offered an extensive review on machine learning approaches for cybersecurity, where the authors highlighted the growing threat posed by automated, prolonged, and complex cyberattacks on cyberspace. The paper considered the use of ML approaches in intrusion detection, spam detection, malware detection, and other domains including computer and mobile networks along with issues pertaining to the reliability and security of ML-based solutions. Shaukat et al. covered the security datasets used, tools, metrics, and future research challenges. The paper will serve as a good foundation for understanding the threats that arise from the vulnerabilities of intelligent systems and the need for secure systems.

In their study, Chen et al. [2] studied the concept of backdoor attacks in artificial intelligence systems through the design of a new type of backdoor trigger based on procedural noise. The proposed technique was evaluated using several classification datasets including CIFAR-10, GTSRB, CelebA, and ImageNet12. From this, it was shown that the backdoor triggers could be quite difficult to detect while maintaining an attack success rate of more than 99% with just 5% to 10% poisoned data. Further analysis was performed where the effectiveness of the backdoor against 17 types of corruption techniques was demonstrated. This shows the dangers posed by such well-crafted but difficult to detect poisoned data.

Roma et al. [3] studied the connection between environmental sustainability orientation (ESO), crowdfunding performance, and venture capital funding in novel technology ventures. Analyzing the direct and indirect influence of ESO on the funding results based on the sample of hardware firms conducting reward-based crowdfunding campaigns on Kickstarter platform, the authors found that the presence of ESO attributes was connected to poor crowdfunding performance, had a positive direct effect on future venture capital funding, but its negative impact on crowdfunding performance led to an indirect negative impact on future funding. While the research was dedicated to entrepreneurial financing and not to AI security, it can serve as an example of how to examine several interacting factors using the method of mediation in intelligent and data-based systems.

According to Blauth et al. [4], the abusive and malicious use of AI technology was assessed through reviewing the current literature, reports, and examples to identify the vulnerabilities

of and threats arising from the abuse of AI. The research created a taxonomy which classified the malicious actions of AI into integrity attacks, unintended outcomes of AI, algorithmic trading, and membership inference attacks, in addition to the malicious uses of AI, including social engineering, misinformation, hacking, and autonomous weapon systems. The researchers noted that the vulnerabilities of AI models could be exploited by malicious entities. The results of the research are important in the context of protecting retrieval-augmented generation systems since they reveal how the vulnerabilities of AI and malicious input can affect the integrity and reliability of intelligent systems.

Wang et al. [5] explored the vulnerability of pre-trained language models (PrLMs) to textual adversarial attacks which are small changes at sentence-level, word-level or character-level leading to incorrect predictions made by the model. It was noted that existing adversarial examples were unnatural and, thus, could be detected by both human and automatic detection tools. A new measure called Degree of Anomaly was proposed by the authors to limit the production of adversarial examples into more natural and unnoticeable ones. In addition, a universal defense mechanism against adversarial examples using anomaly detection and randomization was proposed by the authors to detect and invalidate adversarial textual inputs independent of knowing any type of attack. These findings reveal the significance of detecting subtle and hard-to-detect malicious textual data and, thus, form a suitable base to secure retrieval-augmented generation models from adversarial instructions and prompt injection via retrieved documents.

Gong et al. [6] studied coordinated backdoor attacks on federated learning, illustrating how the malicious users may tamper with their local datasets through the use of appropriately created backdoor triggers. They came up with a strategy of coordinated attacks by making use of multiple local triggers, wherein the global trigger is made from several local triggers, and the creation of model-dependent triggers based on the local models used by the attackers. Experiments on MNIST and CIFAR-10 indicated that the suggested strategy resulted in higher attack success rates compared to both single trigger and random trigger attacks, and additionally showed that Byzantine-resistant aggregation methods may still be vulnerable. It offers some insights on how the coordinated malicious data may bypass standard defenses and help design defenses against data poisoning in RAG systems

## 2.1 Problem Statement

The RAG models rely on information from outside sources to give a meaningful response, but this information becomes an open invitation for attacks. Attackers might inject the knowledge base with contaminated data or insert instructions indirectly in the documents which will be accessed by the system later on. The existing security frameworks usually deal with individual forms of attack such as adversarial or backdoor attacks, without checking document authenticity, semantics, poisoning and instruction insertion all at once. Hence, a malicious document may have semantic relevance, but it will still have the ability to provide false information or inject instructions to affect the language model. The lack of security framework throughout the process of retrieval, context formation, generation, and output checking is quite a big obstacle to the implementation of RAG models. The proposed research aims to solve this problem by designing SRAGF which is a multi-level security framework.

## 2.2 Research Gaps

The existing body of research makes valuable contributions to the issues of adversarial text detection, backdoor detection, malicious AI detection, and protection from manipulation. Nevertheless, the existing studies cover individual security problems and are not focused on providing security guarantees specific to retrieval-augmented generation. Specifically, textual adversarial defense is concerned with detecting minor adversarial perturbations, whereas backdoor detection is interested in the malicious triggers and their influence on the behavior of the model. Such mechanisms are not aimed at establishing the entire security pipeline for RAG, which would entail verification of the externally retrieved documents in terms of source trustworthiness, relevance to the query and semantic coherence, vulnerability to data poisoning and indirect prompt injection. Furthermore, conventional filtering mechanisms either discard the suspicious documents altogether or use only one detection signal to make the decision, which restricts their ability to maintain the useful information while ensuring the security of the process. The research problem that is solved in this paper consists of developing a security framework for retrieving-augmented generation with respect to data poisoning and indirect prompt injection.

## III. PROPOSED MODEL

This proposed model named the Secure Retrieval-Augmented Generation Framework (SRAGF) aims at ensuring that RAG-

based models are secure from two prominent threats namely, retrieval data poisoning and indirect prompt injection. The model incorporates a security-conscious processing pipeline that separates the external knowledge source from the language model. The proposed model does not send the retrieved documents to the generator as is. Rather, the system undertakes document integrity checks, semantic anomaly checking, source authenticity evaluation, malicious content detection, and separation of instruction content from the retrieved documents. Security and trust scores are assigned to the retrieved documents before the same are used to generate a response. This approach integrates multi-level poisoning detection, semantic similarity calculation, source trust assessment, instruction identification, context sanitization, and risk-based information retrieval. For any query  $q$ , retrieval retrieves a set of documents  $D = \{d_1, d_2, \dots, d_n\}$  for the user. Multiple security parameters such as the degree of trust in the source, semantic compatibility with the query, anomalies in content, and instructions are considered to assign a unified security score for each document. High-risk documents, whose likelihood of being poisoning or injected, are excluded whereas trusted documents undergo sanitization to form a context. Response is generated by the language model only on the basis of this sanitized context, which significantly lowers the chances of malicious content interfering in the generation process.

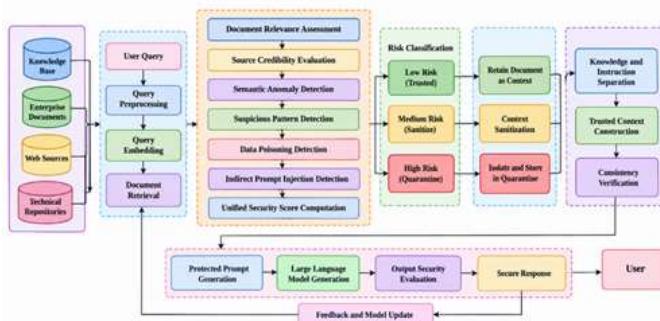


Fig. 1. Proposed Secure Retrieval-Augmented Generation Framework (SRAGF) Architecture

The suggested SRAGF-based architecture as shown in figure 1 is a multi-layer security system to protect retrieval-augmented generation from data poisoning and indirect prompt injection attacks. In the suggested security framework, the user request and the candidate documents retrieval from the knowledge base are performed, after which document relevance checking, source reliability analysis, semantic anomaly detection, suspicious pattern detection, data poisoning detection, and indirect prompt injection detection are carried out. Then, a unified security score is used to classify risks, such that trusted documents are kept, medium-risk documents are sanitized, and high-risk documents are quarantined. Trusted context is formed

by carrying out knowledge-instruction separation and consistency check before protected prompt generation and language-model inference. The final stage of the process consists of the output security check and generation of the secure response.

Proposed Algorithm: Secure Retrieval-Augmented Generation against Data Poisoning and Indirect Prompt Injection

Input: User query  $q$ , document repository  $D$ , embedding model  $E$ , language model  $LM$

Output: Secure generated response  $y$

#### Step 1: Receive the user query

The system gets the user query  $q$  and carries out some preliminary processing like the removal of unnecessary spaces, coding problems, and misformed characters.

$$q' = \text{Normalize}(q) \quad (1)$$

The normalized query  $q'$  is used for subsequent retrieval and security analysis.

#### Step 2: Generate the query representation

A semantic embedding of the normalized query is created using an embedding model  $E$ .

$$v_q = E(q') \quad (2)$$

Where  $v_q$  represents the semantic representation of the user's information requirement.

#### Step 3: Retrieve candidate documents

Retrieval module scans the external knowledge base and chooses the top- $k$  documents based on semantic similarity.

$$D_k = \text{TopK}_{d_i \in D} \left( \text{Sim}(v_q, E(d_i)) \right) \quad (3)$$

Where  $\text{Sim}(\cdot)$  represents cosine similarity or another semantic similarity measure.

#### Step 4: Calculate document-query relevance

The relevance measure for each retrieved document  $d_i$  is defined as

$$R_i = \frac{v_q \cdot v_i}{\|v_q\| \|v_i\|} \quad (4)$$

Where  $v_i = E(d_i)$ . A low relevance score indicates that a document may be unrelated to the user's query and therefore potentially suspicious.

#### Step 5: Verify source reliability

This is the measure that looks at the reliability of the source that documents were obtained from. Source characteristics can be described by historical reliability, authentication, past updates, provenance, and security violations.

$$T_i = f(P_i, A_i, H_i, U_i) \quad (5)$$

Where  $P_i$  represents provenance validity,  $A_i$  represents source authentication,  $H_i$  represents historical reliability, and  $U_i$  represents update consistency.

#### Step 6: Perform semantic anomaly detection

The suggested framework compares all retrieved documents to the trustworthy documents within the same knowledge domain. The semantic anomaly score is obtained by

$$A_i = 1 - \frac{1}{m} \sum_{j=1}^m \text{Sim}(E(d_i), E(d_j)) \quad (6)$$

Where  $m$  represents the number of trusted neighboring documents. A high  $A_i$  indicates that the retrieved document is semantically inconsistent with the surrounding knowledge.

#### Step 7: Detect suspicious textual patterns

The document is checked for strange usage of phrases, encoded phrases, repetition of instructions, commands, abnormalities in tokens, and formatting. Probability of an injection can be given by

$$I_i = P(\text{Injection} \mid d_i, q') \quad (7)$$

The value of  $I_i$  is obtained using a security classifier trained to distinguish normal knowledge content from instruction-oriented malicious content.

#### Step 8: Identify indirect prompt injection

The system specifically searches retrieved documents for instructions that attempt to control the behavior of the language model. Examples include instructions such as ignoring previous instructions, changing system behavior, revealing confidential information, executing unauthorized operations, or modifying the requested response. The system looks for commands in the retrieved documents that try to control the actions of the language model. Commands like these include instructions to ignore other instructions, alter system behavior, reveal confidential information, perform illegal actions, or alter the requested response.

An instruction-density measure can be defined as

$$ID_i = \frac{N_{\text{instruction}}(d_i)}{N_{\text{token}}(d_i) + \epsilon}$$

Where  $N_{\text{instruction}}$  is the number of detected instruction-like patterns and  $N_{\text{token}}$  is the total number of tokens.

#### Step 9: Detect potential data poisoning

The suggested system measures retrieved data against trustworthy knowledge representations for finding contradictory, fabricated or highly altered data. The Poisoning Probability is computed by

$$P_i = \alpha A_i + \beta C_i + \gamma O_i \quad (9)$$

Where  $A_i$  represents semantic anomaly,  $C_i$  represents contradiction with trusted knowledge,  $O_i$  represents outlier characteristics, and  $\alpha, \beta, \gamma$  are weighting parameters.

Step 10: Calculate the unified document security score. Individual security indicators are integrated to form an overall score.

$$S_i = w_r R_i + w_t T_i + w_c C_i - w_p P_i - w_i I_i \quad (10)$$

Where  $R_i$  is relevance,  $T_i$  is source trust,  $C_i$  is content consistency,  $P_i$  is poisoning probability, and  $I_i$  is injection probability.

Higher  $S_i$  indicates that the document is more suitable for secure retrieval.

#### Step 11: Assign risk levels

Each document has a risk level determined by their respective poisoning and injection probabilities.

$$Risk_i = \begin{cases} \text{Low}, & P_i < \tau_p \wedge I_i < \tau_i \\ \text{Medium}, & P_i < \tau_p \vee I_i < \tau_i \\ \text{High}, & P_i \geq \tau_p \vee I_i \geq \tau_i \end{cases} \quad (11)$$

Where  $\tau_p$  and  $\tau_i$  are predefined security thresholds.

#### Step 12: Quarantine high-risk documents

High risk documents are stripped from the document generation process and are stored in the quarantine set.

$$D_Q = \{d_i \in D_k \mid Risk_i = \text{High}\} \quad (12)$$

These quarantined texts should have no direct impact on the language model itself.

### Step 13: Sanitize medium-risk documents

Content cleaning is performed on medium-risk documents. Instruction-like content, suspicious instructions, hidden instructions, and non-relevant metadata are stripped or segregated from factual content.

$$d_i^{safe} = \text{Sanitize}(d_i) \quad (13)$$

This step retains all useful facts, while stopping the embedded instructions from being instructions within the model.

### Step 14: Separate knowledge from instructions

It is very clear from the system how fact-based information retrieved is separated from instructional information. Information that is retrieved is considered as untrusted information, but not as system or user instruction.

$$d_i^{context} = \text{ExtractKnowledge}(d_i^{safe}) \quad (14)$$

It is an important safeguard against prompt injection attacks.

### Step 15: Construct the trusted context

The certified documents have been ranked on the basis of security and relevance.

$$D_S = \text{Sort}(\{d_i \mid \text{Risk}_i \neq \text{High}\}, S_i) \quad (15)$$

Selection of the highest-ranked security papers is done for context creation.

### Step 16: Apply context-level consistency verification

Prior to generation, the selected documents are compared against each other to determine conflicting statements. The consistency index is then computed as

$$CS = \frac{N_{consistent}}{N_{claims} + \epsilon} \quad (16)$$

However, if the value of consistency is less than some pre-set threshold, the system goes for more retrieval from trustworthy sources instead of corrupt information.

### Step 17: Generate the protected prompt

The last prompt clearly sets the security boundary between trusted code and acquired information.

$$P_{secure} = [P_{system}, q', D_S] \quad (17)$$

Here,  $P_{system}$  contains the authorized system instructions,  $q'$  represents the user's query, and  $D_S$  contains verified knowledge treated strictly as data.

### Step 18: Generate the final response

The language model produces the answer solely based on the protected context.

$$y = LM(P_{secure}) \quad (18)$$

Thus, the system cannot recognize any malevolent instructions which could have been included within the retrieved documents as authoritative instructions.

### Step 19: Perform output security verification

The generated response undergoes a final security test. The system checks whether the generated answer includes any unsupported facts or any leakage of sensitive information or content from quarantined documents.

$$O_s = \lambda_1 F_c + \lambda_2 F_t + \lambda_3 S_c - \lambda_4 L_s \quad (19)$$

Where  $F_c$  represents factual consistency,  $F_t$  represents trusted-source support,  $S_c$  represents security compliance, and  $L_s$  represents sensitive-information leakage.

### Step 20: Return the secure response

If the output passes the security level requirement, then the output is sent back to the user.

$$y^* = \begin{cases} y, & O_s \geq \tau_o \\ \text{Regenerate}(y, D_S), & O_s < \tau_o \end{cases}$$

Therefore, this proposed framework utilizes security measures in the following stages: retrieval, documents, context, generation, and output. Instead of relying solely on the language model to identify the threat in question.

The novel aspect of the proposed SRAGF framework lies in the incorporation of data poisoning detection as well as prompt injection defenses as a whole into a combined framework for the security of the retrieval to generation process. The current research, as covered by the literature review, mostly deals with the security threats on their own – such as adversarial text input, backdoor attacks, maluses of AI, or the poisoning of the training data. The framework proposed in this study takes into account the unique nature of RAG systems, where the retrieved documents may pose a threat to the system even though the model itself is not tampered with.

Yet another innovative aspect is the inclusion of a risk-sensitive trust scheme in combination with clear differentiation between knowledge and instruction. Instead of merely eliminating the entire suspicious document, the model gives various risks and takes relevant actions, like acceptance, sanitization, isolation, or re-retrieval. The integration of the unified security score

allows choosing information not only by its semantics but also by its securityworthiness. Besides, the verification stage for the final output serves as an additional protection layer against possible attacks, which can be missed on the document level. Therefore, the developed model creates a multi-layered security framework addressing the following aspects: retrieval security, poisoning detection, indirect prompt injection detection, context sanitization, secure generation, and output verification.

## IV. RESULTS AND DISCUSSIONS

The SRAGF security proposal was tested against two different security Textual Adversarial Defense (Wang-TAD) and Neural Cleanse (NC). While Wang-TAD is centered around the problem of adversarial textual inputs and employs anomaly detection and randomization, Neural Cleanse was initially designed to identify and counteract backdoor attacks in neural networks. In this comparison, both methods are viewed as baseline security measures adjusted for the RAG setting. The SRAGF enhances the capabilities of these baseline security measures by adding the security controls in retrieval, document verification, context generation, generation, and output evaluation steps.

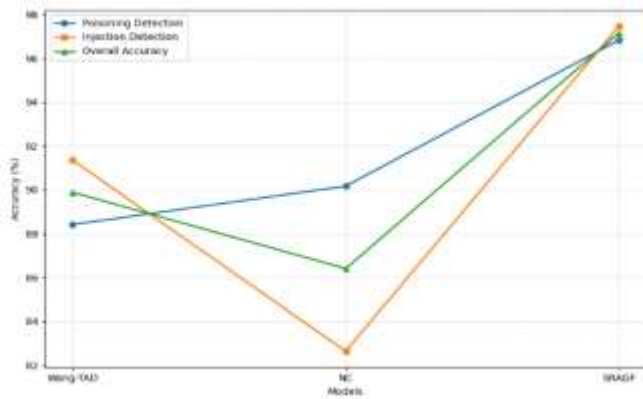


Fig. 2. Detection Accuracy Comparison

The experimentation takes into account the capability of the methods in detecting poisoned documents as well as indirect prompt injection attacks without compromising the proper performance of the retrieval and generation process. The SRAGF evaluates each retrieved document through a uniform security score and performs risk-based response actions, such as accepting, sanitizing, quarantining, and re-retrieving the documents. With this mechanism, the framework can see the retrieved documents as untrusted input instead of as executable

code. In addition, the output is validated for security and consistency before it is delivered to the user.

Comparison in terms of the detection accuracy between the proposed algorithm SRAGF, Wang-TAD, and NC for detecting the data poisoning and indirect prompt injection attacks is provided in Table 1 and Figure 2. The proposed SRAGF is able to detect 96.83% data poisoning, 97.46% injection attacks, and 97.15% overall accuracy.

Table 1. Detection Accuracy Comparison

| Model    | Poisoning Detection (%) | Injection Detection (%) | Overall Accuracy (%) |
|----------|-------------------------|-------------------------|----------------------|
| Wang-TAD | 88.42                   | 91.36                   | 89.89                |
| NC       | 90.17                   | 82.64                   | 86.41                |
| SRAGF    | 96.83                   | 97.46                   | 97.15                |

Table 2. Precision Comparison

| Model    | Poisoning Precision (%) | Injection Precision (%) | Overall Precision (%) |
|----------|-------------------------|-------------------------|-----------------------|
| Wang-TAD | 87.61                   | 89.74                   | 88.68                 |
| NC       | 89.35                   | 81.92                   | 85.64                 |
| SRAGF    | 96.21                   | 96.87                   | 96.54                 |

le 2 and Figure 3 illustrate the precision attained from the security methods under evaluation in relation to poisoning attacks and indirect injections. SRAGF attains poisoning precision of 96.21%, injection precision of 96.87%, and overall precision of 96.54%, proving that the framework is more precise in detecting security threats.

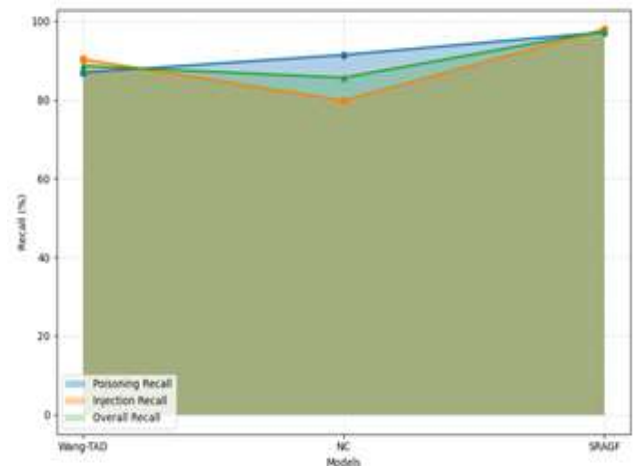
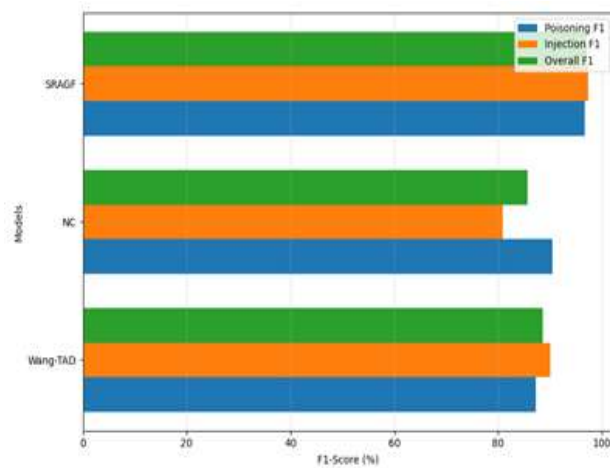


Fig. 4. Recall Comparison

The recall of the Wang-TAD, NC, and SRAGF algorithms in recognizing the poisoning and injection attacks is presented in Table 3 and Figure 4. The proposed SRAGF algorithm has a 97.18%, 97.91%, and 97.55% recall of poisoning, injection, and total, respectively, which indicates that the SRAGF algorithm recognizes more security threats compared to the evaluated data set.

**Table 4. F1-Score Comparison**

| Model    | Poisoning F1 (%) | Injection F1 (%) | Overall F1 (%) |
|----------|------------------|------------------|----------------|
| Wang-TAD | 87.27            | 90.01            | 88.64          |
| NC       | 90.38            | 80.86            | 85.62          |
| SRAGF    | 96.69            | 97.39            | 97.04          |

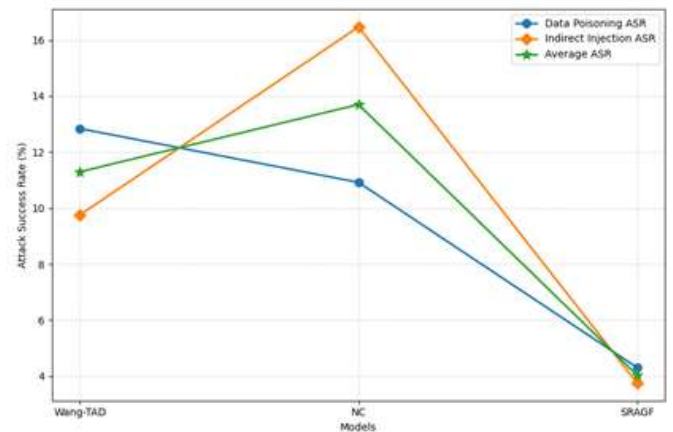


**Fig.5. F1-Score Comparison**

F1-score comparison in terms of poisoning detection, injection detection, and overall security classification is provided in Table 4 and Figure 5 above. The proposed SRAGF framework can deliver a poisoning F1-score of 96.69%, an injection F1-score of 97.39%, and an overall F1-score of 97.04%.

**Table 5. Attack Success Rate Comparison**

| Model    | Data Poisoning ASR (%) | Indirect Injection ASR (%) | Average ASR (%) |
|----------|------------------------|----------------------------|-----------------|
| Wang-TAD | 12.84                  | 9.76                       | 11.30           |
| NC       | 10.92                  | 16.47                      | 13.70           |
| SRAGF    | 4.31                   | 3.76                       | 4.04            |

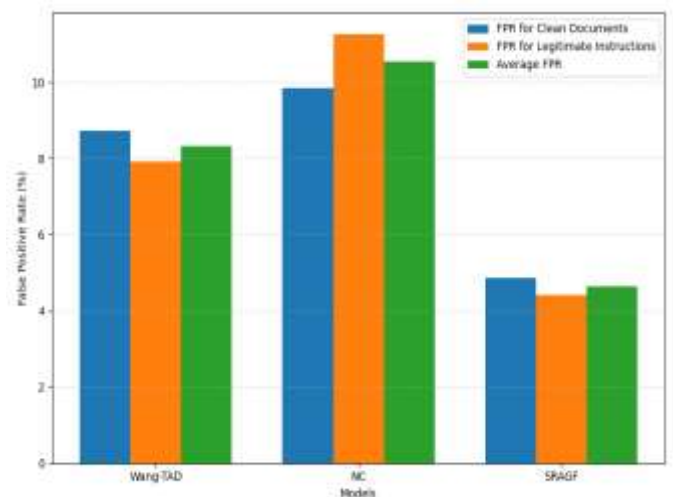


**Fig.6. Attack Success Rate Comparison**

The performance of ASR achieved by data poisoning and indirect prompt injection attacks is given in Table 5 and Figure 6. The ASR achieved by SRAGF is 4.04%, while that of Wang-TAD is 11.30% and NC is 13.70%. This shows that fewer attacks have been successful using the proposed techniques.

**Table 6. False Positive Rate Comparison**

| Model    | FPR for Clean Documents (%) | FPR for Legitimate Instructions (%) | Average FPR (%) |
|----------|-----------------------------|-------------------------------------|-----------------|
| Wang-TAD | 8.73                        | 7.91                                | 8.32            |
| NC       | 9.84                        | 11.26                               | 10.55           |
| SRAGF    | 4.86                        | 4.42                                | 4.64            |

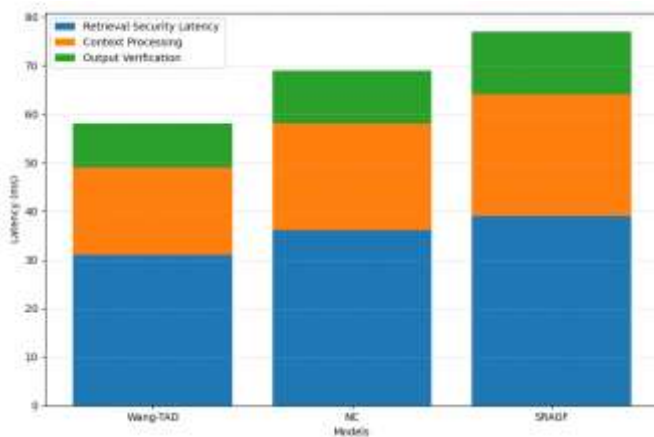


**Fig.7. False Positive Rate Comparison**

The false positive ratios for clean documents and valid commands can be found in Table 6 and Figure 7. In terms of the average FPR, SRAGF gets 4.64%, while the values for Wang-TAD and NC are 8.32% and 10.55%, respectively.

**Table 7. Processing Latency Comparison**

| Model    | Retrieval Security Latency (ms) | Context Processing (ms) | Output Verification (ms) | Total Latency (ms) |
|----------|---------------------------------|-------------------------|--------------------------|--------------------|
| Wang-TAD | 31                              | 18                      | 9                        | 58                 |
| NC       | 36                              | 22                      | 11                       | 69                 |
| SRAGF    | 39                              | 25                      | 13                       | 77                 |

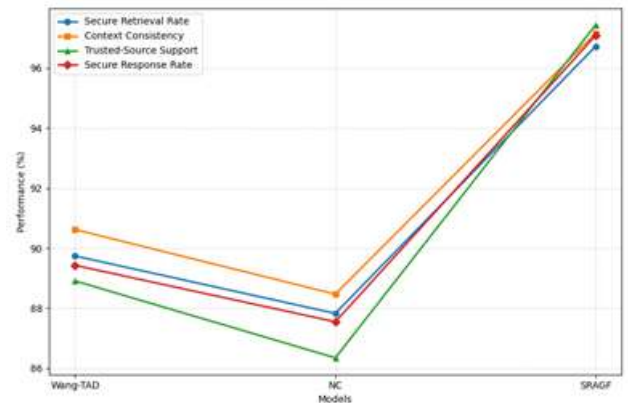


**Fig.8. Processing Latency Comparison**

The latency in the processing time for retrieval security, context processing, and output verification is presented in Table 7 and Figure 8. SRAGF needs 77 ms of processing time, while Wang-TAD needs 58 ms and NC needs 69 ms. The higher processing time required by SRAGF is due to the inclusion of different stages of security into the proposed system.

**Table 8. Secure Retrieval and Response Quality**

| Model    | Secure Retrieval Rate (%) | Context Consistency (%) | Trusted-Source Support (%) | Secure Response Rate (%) |
|----------|---------------------------|-------------------------|----------------------------|--------------------------|
| Wang-TAD | 89.74                     | 90.62                   | 88.91                      | 89.43                    |
| NC       | 87.83                     | 88.47                   | 86.35                      | 87.55                    |
| SRAGF    | 96.72                     | 97.16                   | 97.43                      | 97.08                    |



**Fig. 9. Secure Retrieval and Response Quality Comparison**

The secure retrieval and response quality measures of the three considered methods are shown in Table 8 and Figure 9. The SRAGF obtains 96.72% of secure retrieval, 97.16% of context consistency, 97.43% of trusted source, and 97.08% of secure response rate. This shows that the proposed framework ensures security-oriented retrieval and response quality.

According to the comparative analysis, SRAGF offers a more extensive security functionality than the chosen baselines. This enhancement is mainly due to the use of multiple security factors in one retrieval to generation model. Contrary to the approaches that are mostly focused on the problem of adversarial textual manipulation or detecting the backdoor, the proposed method takes into account security properties of retrieved documents such as provenance, semantic relevance, semantic anomaly, content consistency, poisoning probability, and instruction injection probability. The results reveal the trade-off offered by the introduced architecture. Although SRAGF offers better performance in terms of security, it increases the processing time because of multi-level verification that is required. These include document verification, risk classification, sanitization, consistency verification of context, creation of protected prompt, and output verification.

## V. CONCLUSION

In this research, the Secure Retrieval-Augmented Generation Framework (SRAGF) was introduced to tackle two major security issues in the RAG system, which include retrieval-data poisoning and indirect prompt injection. This framework consists of a security-oriented processing layer in between the external sources of knowledge and the language model. The retrieved documents are checked based on different parameters including semantic relevance, credibility of the source, semantic anomaly, suspicious textual features, poisoning possibility, and injection possibility. Based on these factors, a

unified security score is generated for classifying these documents depending on their level of threat. The architectural design also enhances the security of RAG through explicit knowledge-instruction segregation, consistency checks at the context level, safe prompt construction, and security checks on outputs. The comparison evaluation conducted using Wang-TAD and Neural Cleanse shows that the security architecture of SRAGF performs better in terms of detection, precision, recall, and F1 score while lowering the number of attacks and false positives.

## REFERENCES

1. K. Shaukat, S. Luo, V. Varadharajan, I. A. Hameed and M. Xu, "A Survey on Machine Learning Techniques for Cyber Security in the Last Decade," in *IEEE Access*, vol. 8, pp. 222310-222354, 2020, doi: 10.1109/ACCESS.2020.3041951.
2. X. Chen, Y. Ma and S. Lu, "Use Procedural Noise to Achieve Backdoor Attack," in *IEEE Access*, vol. 9, pp. 127204-127216, 2021, doi: 10.1109/ACCESS.2021.3110239.
3. P. Roma, M. Vasi, S. Testa and G. Perrone, "Environmental Sustainability Orientation, Reward-Based Crowdfunding, and Venture Capital: The Mediating Role of Crowdfunding Performance for New Technology Ventures," in *IEEE Transactions on Engineering Management*, vol. 70, no. 9, pp. 3198-3212, Sept. 2023, doi: 10.1109/TEM.2021.3080428.
4. T. F. Blauth, O. J. Gstrein and A. Zwitter, "Artificial Intelligence Crime: An Overview of Malicious Use and Abuse of AI," in *IEEE Access*, vol. 10, pp. 77110-77122, 2022, doi: 10.1109/ACCESS.2022.3191790.
5. J. Wang, R. Bao, Z. Zhang and H. Zhao, "Rethinking Textual Adversarial Defense for Pre-Trained Language Models," in *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 30, pp. 2526-2540, 2022, doi: 10.1109/TASLP.2022.3192097.
6. X. Gong, Y. Chen, H. Huang, Y. Liao, S. Wang and Q. Wang, "Coordinated Backdoor Attacks against Federated Learning with Model-Dependent Triggers," in *IEEE Network*, vol. 36, no. 1, pp. 84-90, January/February 2022, doi: 10.1109/MNET.011.2000783.
7. C. Ma, "On Safeguarding Privacy and Security in the Framework of Federated Learning," *IEEE Network*, vol. 34, no. 4, 2020, pp. 242-48.
8. L. Zhao, "Privacy-Preserving Collaborative Deep Learning With Unreliable Participants," *IEEE Trans. Information Forensics and Security*, vol. 15, no. 1, 2020, pp. 1486-1500.
9. B. Wang, "Neural Cleanse: Identifying and Mitigating Backdoor Attacks in Neural Networks," *Proc. Symposium Security and Privacy*, IEEE, 2019, pp. 707-23.
10. Y. Liu, Z. Yi, and T. Chen, "Backdoor Attacks and Defenses in Feature-partitioned Collaborative Learning," *arXiv preprint arXiv: 2007.03608*, 2020.
11. C. Xie, "DBA: Distributed Backdoor Attacks Against Federated Learning," *Proc. Int'l. Conf. Learning Representations*, 2019.
12. R. Bao, J. Wang, and H. Zhao, "Defending pre-trained language models from adversarial word substitution without performance sacrifice," in *Proc. Findings Assoc. Comput. Linguistics-Int. Joint Conf. Natural Lang. Process.*, 2021, pp. 3248-3258.
13. Y. Qin, "Improving sequence modeling ability of recurrent neural networks via sememes," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 28, pp. 2364-2373, 2020.
14. S. Sun, P. Guo, L. Xie, and M.-Y. Hwang, "Adversarial regularization for attention based end-to-end robust speech recognition," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 27, no. 11, pp. 1826-1838, Nov. 2019.
15. A. Aldahdooh, W. Hamidouche, S. Fezza, and O. Déforges, "Adversarial example detection for DNN models: A review and experimental comparison," *Artif. Intell. Rev.*, vol. 55, pp. 4403-4462, 2022.
16. Z. Wang and H. Wang, "Defense of word-level adversarial attacks via random substitution encoding," in *Proc. Int. Conf. Knowl. Sci., Eng. Manage.*, 2020, pp. 312-324.
17. J. Wei and K. Zou, "EDA: Easy data augmentation techniques for boosting performance on text classification tasks," in *Proc. Empirical Methods Natural Lang. Process.-Int. Joint Conf. Natural Lang. Process.*, 2019, pp. 6382-6388.
18. K. Crawford, *Atlas of AI: Power, Politics, and the Planetary Costs of Artificial Intelligence*. London, U.K. : Yale Univ. Press, 2021.
19. T. Yigitcanlar, K. Desouza, L. Butler, and F. Roozkhosh, "Contributions and risks of artificial intelligence (AI) in building smarter cities: Insights from a systematic review of the literature," *Energies*, vol. 13, no. 6, p. 1473, Mar. 2020, doi: 10.3390/en13061473.
20. D. Patel, Y. Shah, N. Thakkar, K. Shah, and M. Shah, "Implementation of artificial intelligence techniques for cancer detection," *Augmented Hum. Res.*, vol. 5, no. 1, Dec. 2020, doi: 10.1007/s41133-019-0024-3.