RAG-based 5G Conformance Testing

Asif Shahriar*, K.M. Asifur Rahman*, Md. Shohrab Hossain*
*Department of CSE, Bangladesh University of Engineering and Technology, Dhaka, Bangladesh
Email: asif.asr11@gmail.com, asifurndc8030@gmail.com, mshohrabhossain@cse.buet.ac.bd

***Abstract*—Creating conformance tests for 5G network protocols is a time-consuming, detail-heavy task—one that grows more challenging as standards evolve. Manual test case generation often struggles to keep up, risking delays or oversights. To tackle this, we designed an AI-powered system that automatically generates accurate, standards-aligned test cases for the 5G Non-Access Stratum (NAS) protocol. By combining a retrieval-augmented generation (RAG) pipeline—trained on 5G NAS specifications stored in a vector database—with GPT-4's language capabilities, our approach minimizes errors and ensures test cases adhere to 3GPP requirements. The system produced 800 test cases covering critical procedures like device authentication and session management. To verify quality, we manually cross-checked a subset against the official 5G NAS documentation, confirming their technical correctness. This work demonstrates how AI can streamline telecom testing: reducing manual effort, accelerating certification, and adapting to specification updates. While challenges like ambiguous standards language remain, our framework offers a scalable, reliable foundation for automating compliance testing in complex, ever-changing domains like 5G and beyond.**

***Index Terms*—5G, vulnerability detection, GPT-4, security protocols, cellular network, AI-driven security, prompt engineering.**

I. Introduction

5G specifications are characterized by their extensive detail and intricate interdependencies, often spanning hundreds of pages and covering a multitude of protocols and procedures. This complexity poses significant challenges for both implementation and verification, as the precise operational nuances must be correctly interpreted and adhered to. Conformance testing plays a critical role in ensuring that implementations strictly follow these detailed specifications. It systematically verifies that each protocol, procedure, and functional aspect is correctly implemented, thus preventing interoperability issues and ensuring the reliability, security, and performance of the deployed 5G systems.

Generating 5G conformance test cases is inherently challenging due to the extensive complexity and sheer volume of the 5G-NAS specifications. It is nearly impossible to manually parse every detail, identify all relevant protocol conditions, and account for the intricate interdependencies and edge cases inherent in these documents. To address these challenges, several semi-automated procedures have been developed. These methods typically rely on rule-based extraction, pattern matching, and basic natural language processing techniques to identify and extract test scenarios from the specification. However, they are limited in their ability to capture the nuanced and context-dependent relationships within the text. As a result, such approaches may yield test cases that are overly generic, miss critical edge conditions, or require extensive manual adjustments to achieve comprehensive coverage.

Large Language Models (LLMs) can significantly streamline the process by automatically synthesizing and interpreting vast, complex technical documents; reducing the need for extensive manual intervention in test-case generation. Their ability to understand context and extract semantic meaning facilitates the discovery of intricate dependencies and novel test scenarios that might be overlooked by traditional rule-based methods. However, LLMs are not without their challenges. One major issue is hallucination, where the model may produce plausible-sounding but inaccurate or fabricated information, compromising the reliability of the generated test cases. Additionally, as black-box systems, LLMs lack transparency in their internal decision-making processes, making it difficult to verify the source and rationale behind specific outputs. Furthermore, the general-purpose nature of these models often results in a lack of specialized domain knowledge, which can limit their ability to fully capture the nuanced requirements and technical depth necessary for precise 5G conformance testing.

Retrieval-Augmented Generation (RAG) is an approach that integrates external information retrieval with language model generation to produce outputs grounded in verified, domain-specific data. By allowing LLMs to access specialized knowledge bases, RAG enriches responses with precise and up-to-date information and can include citations or source references alongside the generated text; this grounding minimizes the risk of fabricated content and hallucinations. Moreover, since the external data source can be easily updated, RAG ensures that the generated outputs remain current and accurate without necessitating retraining of the underlying model.

In this paper, we present an end-to-end methodology for generating 5G conformance test cases using a retrieval-augmented generation (RAG) framework. Our approach begins by converting extensive 5G-NAS specifications into structured text and indexing them in a vector store, ensuring that even large sections are segmented into manageable, context-rich chunks. By integrating explicit section text with relevant context retrieved from the vector store, we construct detailed prompts that guide an LLM to generate precise and comprehensive test cases. This workflow effectively automates a process that would be prohibitively labor-intensive if done manually, while minimizing hallucinations and ensuring up-to-date, domain-specific accuracy.

Using our novel RAG-based approach, we have generated 800 conformance test cases across key 5G NAS procedures—authentication, security mode control, registration, de-registration, and service request, highlighting critical areas of protocol compliance, security enforcement, and failure

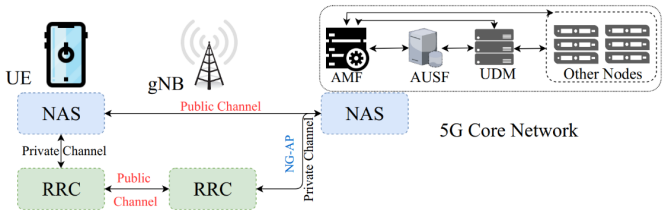

Fig. 1: Simplified 5G Architecture [1]

handling. Moreover, we needed less than 3 hours to generate 800 test-cases, which is quite efficient for these most complex 5G procedures spanning over 200 pages in total.

The major contributions of our work are as follows.

- **Novel, Fully Automated Workflow:** We introduce a novel RAG-based methodology that completely automates the generation of 5G conformance test cases, eliminating the need for any manual intervention throughout the process.
- **Efficient Test-Case Generation:** Our approach achieves remarkable efficiency by generating 800 detailed test cases in just 171 minutes, demonstrating its potential for rapid deployment and scalability.
- **Robust Handling of Cross-Section Dependencies:** The methodology effectively addresses the significant challenge of cross-section dependencies in extensive 5G-NAS specifications, ensuring that contextual coherence and interrelated protocol details are preserved during test-case generation.
- **Scalability and Generalizability:** Designed for scalability, our framework can be readily adapted to other domains with large, complex technical documents, offering a generalizable solution for automated, contextually grounded test-case derivation.

The rest of the paper is organized as follows. Section II provides background knowledge on our research. In section III we discuss existing works on detecting vulnerabilities in cellular protocols. Section IV explains our methodology for generating conformance test-cases in detail. In section V we present the findings from our experiment in detail. Section VI concludes this paper and provides direction on future work.

II. Background

In this section we discuss the basic concepts of 5G network architecture, NAS layer procedures, and prompt engineering techniques relevant to our work.

A. 5G Network Architecture

The 5G network architecture, as illustrated in Fig. 1, comprises of three main components: the User Equipment (UE), 5G Radio Access Network (5G-RAN), and the 5G Core Network (5G-CN).

UE: The UE includes devices like smartphones, tablets, and IoT devices that connect to the 5G network. Each UE contains a non-access stratum (NAS) for managing connection and mobility states with the 5G-CN.

5G-RAN: The 5G-RAN facilitates wireless communication between UEs and the core network. Its primary component is the gNodeB (gNB), which serves as the base station responsible for managing the radio interface, resources, and connectivity.

5G-CN: The 5G core network adopts a service-based architecture to allow network functions to offer services to each other. Key components include the Access and Mobility Management Function (AMF), which manages user access, mobility, and authentication; the Authentication Server Function (AUSF), responsible for handling the authentication of user devices; the Session Management Function (SMF), which manages session establishment, IP address allocation, and other service policies; and the Unified Data Management (UDM), which manages user subscription data and profiles.

The Non-Access Stratum (NAS) layer manages signaling and communication between UEs and the core network for session management, mobility management, and security. The Radio Resource Control (RRC) layer oversees radio bearer configurations, admission control, mobility measurements, and dynamic resource allocation over secure channels.

B. NAS Layer Procedures

Here we briefly discuss some NAS layer procedures relevant to our work.

***Authentication and Key Agreement (AKA)**:* AUSF generates an authentication token (AUTN). A long-term secret key (K) that is shared between the UE and the network. Along with the AUTN, a random number (RAND) and an expected response (XRES) are also generated. The network sends AUTN and RAND to the UE. The UE first validates the received AUTN and then computes a response (RES) based on the received RAND and the long-term key (K). The UE sends the RES back to the network. The network then compares this received RES with the expected response (XRES) it generated initially. If RES matches XRES, it confirms the UE's identity and the authentication is considered successful.

***Security Mode Control**:* After the UE is authenticated, the AMF initiates the Security Mode Command, specifying the algorithms for encryption (e.g., 128-NEA2) and integrity protection (e.g., 128-NIA2). The UE responds with a Security Mode Complete message if it successfully configures the security settings. This procedure ensures that all subsequent NAS messages are protected against eavesdropping and tampering.

***Registration**:* The UE initiates a connection with the 5G network by sending a registration request to the AMF, which includes its Subscription Permanent Identifier (SUPI). The AMF assigns a 5G-Globally Unique Temporary Identifier (GUTI) and updates the UE's location in the UDM for mobility management.

***Deregistration**:* Deregistration can be triggered by either the UE (e.g., when turned off) or the network. The UE sends a Deregistration Request to the AMF, which updates the network's records and releases associated resources. If the UE is inactive, the network may also initiate deregistration to free up resources and maintain efficiency.

Service Request procedure: This procedure is used by UE to establish a signaling or user-plane connection when it needs to resume communication with the network while in 5GMM-IDLE mode. It ensures that the AMF grants access to network services, allowing the UE to either establish a new session or resume an existing PDU session. It is triggered in scenarios such as uplink data transmission, paging response, or emergency service initiation. The procedure also includes handling security context validation, access barring, network congestion, and priority service requests, ensuring efficient and secure service resumption in 5G networks.

C. Retrieval Augmented Generation (RAG)

Retrieval-Augmented Generation (RAG) is an advanced natural language processing (NLP) framework that combines the capabilities of information retrieval and generative models to produce more accurate and context-aware outputs. Developed to enhance the performance of large language models (LLMs), RAG integrates an external knowledge retrieval mechanism with a generative neural network, ensuring that responses are grounded in relevant, factual data rather than relying solely on model pretraining. The RAG framework consists of two main components:

- **Retrieval –** First, the model searches a database, document collection, or knowledge base to find relevant information based on the given input. For example, in 5G conformance testing, it can pull technical details directly from specifications to ensure accuracy.
- **Augmenting –** Once the right information is retrieved, the model combines (or augments) it with the input query. This step helps the AI generate responses that are more relevant, reducing errors and keeping the output aligned with real-world facts.
- **Generation –** Finally, the AI uses a language model, such as GPT or T5, to produce a response that incorporates both the original input and the retrieved data. This ensures that the generated content is not only fluent but also grounded in reliable sources.

By using retrieval and augmentation before generating text, RAG makes AI-powered content much more precise. In fields like 5G testing, this means better test case generation that closely follows technical standards, helping ensure compliance and efficiency.

III. Related Work

A. Retrieval-Augmented Generation (RAG) and AI Verification

In the context of generative AI for VLSI design, [2] introduces a RAG framework to mitigate hallucinations in assertion writing. By leveraging Bounded Model Checking (BMC) for verifying SystemVerilog Assertions (SVAs) within the AXI4-Lite protocol, the framework ensures alignment with high-level design specifications, significantly improving accuracy and trustworthiness in automated assertion generation. Similarly, [3] enhances document context understanding through recursive embedding and clustering, reducing hallucinations in complex multi-step question-answering tasks. Their approach improves accuracy by 20% on the QuALITY benchmark when integrated with GPT-4, demonstrating the efficacy of hierarchical text summarization in addressing long-context challenges.

B. Security and Software Testing with LLMs

Security in 5G systems is addressed by [4], which presents a comprehensive testing framework for 5G Standalone User Equipment (UE). This work fills a critical gap in security methodologies by extending focus beyond older technologies like 4G/LTE, providing a foundation for securing next-generation mobile networks.

The role of Large Language Models (LLMs) in software testing is thoroughly reviewed by [5], identifying challenges such as test coverage and the test oracle problem while exploring opportunities for integrating LLMs with traditional testing methods. The work of Chen et al. [6] and Nijkamp et al. [7] highlights the potential of LLMs for code generation and bug detection, showcasing how AI-assisted tools can enhance software development processes. Our study extends this line of research by applying RAG-based techniques to test case generation, an essential aspect of software verification and validation.

Moreover, our human-in-the-loop approach differentiates our method from fully automated test case generation techniques. Iterative feedback from experienced testers refines and optimizes test cases based on domain expertise and project-specific considerations, aligning with the findings of Trivedi et al. [8] and Guo et al. [9], who emphasize the importance of human involvement in guiding language models for complex tasks.

C. AI-Driven Conformance Testing and Compliance

For relational database management systems (RDBMS), [10] introduces SEMCONT, an automatic conformance testing method that formalizes SQL semantics in Prolog. This approach uncovers bugs and inconsistencies often missed by traditional methods, demonstrating its potential to improve RDBMS reliability.

In safety-critical embedded systems, [11] leverages GPT-4 and a customized RAG framework to augment test suites, improving robustness verification. Applied to projects like PX4 Autopilot and Apollo Auto, the method enhances test coverage, defect detection, and compliance with industry standards, outperforming human-authored test suites in identifying high-severity defects.

Finally, [12] addresses the alignment of Software Requirements Specifications (SRS) with regulatory requirements using a Graph-RAG framework. This approach improves accuracy and context awareness in compliance tasks, though challenges related to complexity and scalability remain.

IV. Proposed Approach

In this section we explain the end-to-end workflow of our proposed approach for generating conformance test-cases from

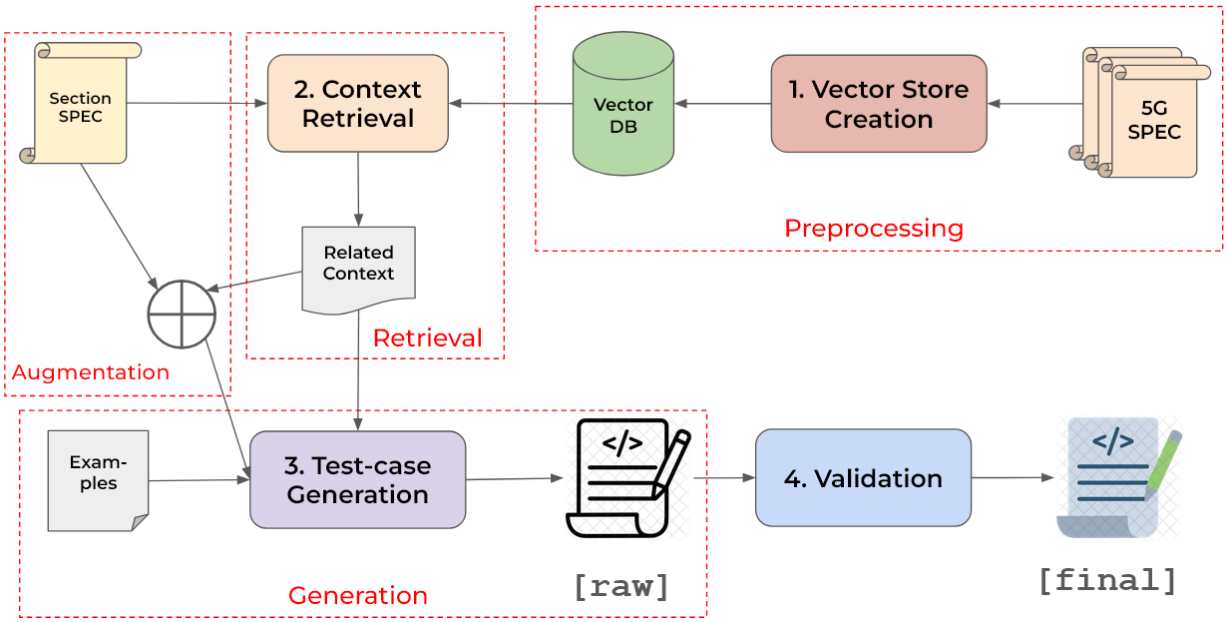

Fig. 2: Full workflow of our RAG-based approach

the 5G-NAS specification. For this work, we have focused on five elementary procedures for 5GS mobility management - authentication, security mode control, registration, deregistration, service request procedure. One significant challenge arises from the sheer size of the specification sections. Altogether, these sections span over 200 pages, while some particular sections like the registration procedure is over 100 pages in length by itself. Although it would be ideal to feed an entire section to the LLM in each run, it is impossible as the size of the section exceeds the LLM's context window. To address this, we have segmented the sections into smaller, manageable subsections. This segmentation introduces yet another challenge - the nuanced relationships and cross-references between different parts of the section could be lost, potentially leading to test cases that do not fully capture the original intent or cover all critical scenarios. To overcome this, we utilize a robust context retrieval mechanism that supplements each manageable subsection with additional, relevant context drawn from the entire specification. This ensures that even when working with segmented content, the generated test cases remain comprehensive, coherent, and reflective of the full range of dependencies outlined in the 5G-NAS specification.

In this section, we explain each component of our RAG-based workflow in detail. Fig.2 illustrates the full workflow of our approach.

A. Preprocessing

In this phase, technical specifications originally provided as PDFs are converted into a structured plain text format. First we sequentially extract text from each page of the document. Once a continuous text stream is obtained, pattern matching techniques using regular expressions are applied to identify section numbers and titles based on common formatting conventions (e.g., hierarchical numbering like "4.5.3" or headings beginning with uppercase letters). This segmentation into coherent sections is essential for the subsequent retrieval and generation stages of the RAG framework, as it ensures that each portion of the specification is clearly delineated and semantically meaningful.

B. Vector Store Creation

Technical specifications are inherently large and complex, making it inefficient to scan the entire document during test-case generation. To address this, the specification is first divided into manageable, overlapping chunks using a recursive character-based splitting strategy. This approach segments the document into pieces of approximately 1000 characters with a deliberate overlap of 200 characters. The overlap preserves local context at the boundaries of each subsection, ensuring that critical information and contextual continuity is maintained across boundaries. Each chunk is further annotated with metadata—namely, the section number and title—extracted through pattern matching tuned to the document's formatting conventions. The annotated chunks are then converted into high-dimensional vector embeddings using a pre-trained model `BAAI/llm-embedder` from HuggingFace, which captures the semantic content of the text. These embeddings are indexed with the FAISS library to create a vector store. This structure enables fast, similarity-based searches, allowing for rapid retrieval of text segments that are most relevant during the test-case generation process. Finally, the vector store is saved to persistent storage, facilitating its reuse in subsequent retrieval operations without the need to reprocess the entire specification.

C. Context Retrieval

In technical documents such as 5G specifications, individual sections often depend on information from other parts of

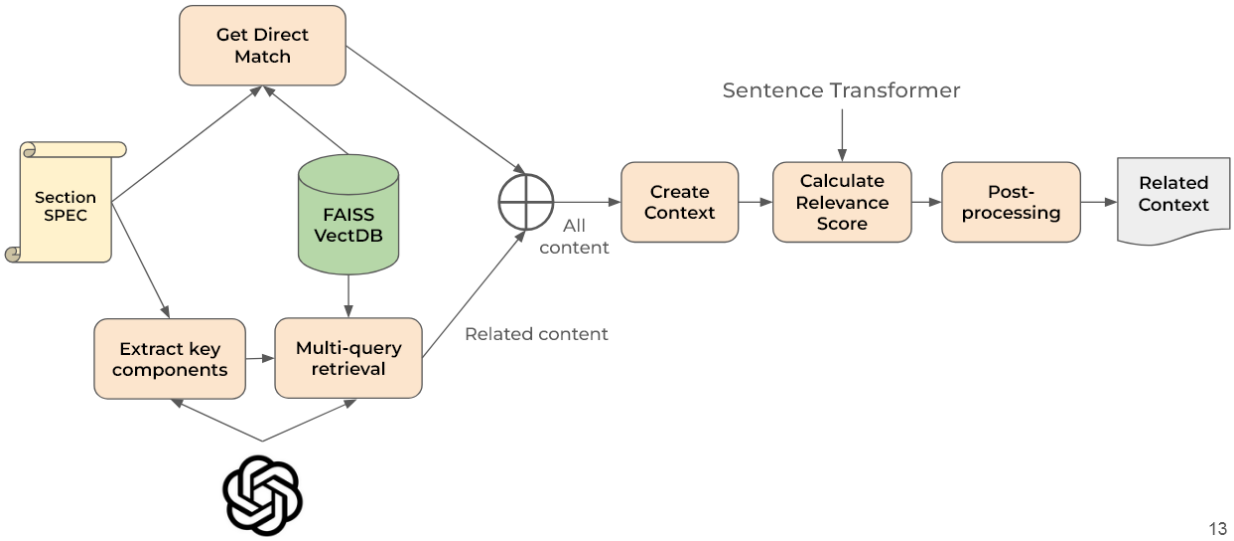

Fig. 3: Context retrieval process

the document. To capture both the direct content of a given section and its related contextual dependencies, a dedicated context retrieval process is employed. This process combines direct matching using a vector store with targeted extraction of related contexts based on key concepts, as shown in fig.3.

Direct and Related Context Extraction: For a given specification section, two types of context are retrieved: direct and related context. For direct context, the text of the target section is used to perform a similarity search within the vector store, filtering for chunks that originate from the same section. This ensures that the primary content of the section is captured accurately. Afterwards, we recognize that a section often references protocols, state transitions, messages, timing details, and other relevant information that are described in other sections. Manually retrieving these related contexts is cumbersome and defeats the purpose of creating an automated test-case generation pipeline. This is why we automate this process by using Large Language Models (LLMs). First we use ChatGPT's natural language understanding capability to extract key concepts —such as protocol elements, state transitions (e.g., Idle to Connected), message formats (e.g., Attach Request), procedures (e.g., Registration), and timer settings— from the section text. These concepts serve as queries to retrieve additional relevant contexts from the vector store. A multi-query retriever, which combines LLM capabilities with the vector store's retrieval mechanism, is used to locate and aggregate these related contexts.

Relevance Scoring and De-duplication: To quantify the importance of each retrieved context, each retrieved text chunk is assigned a relevance score. This score is computed using cosine similarity between the embedding of the section under consideration and that of the context chunk, where the embeddings are generated by the SentenceTransformer model `all-MiniLM-L6-v2`. The relevance score quantifies the semantic closeness between the context and the target section, allowing the system to rank the retrieved chunks so that those most aligned with the section are emphasized during subsequent processing.

In parallel, a de-duplication process is implemented to handle overlapping or redundant text fragments that can arise from both direct retrieval and multi-query searches. The process involves normalizing the text of each context by stripping extra whitespace and converting the text to lowercase to generate a canonical form. This normalized text is used as a key to identify duplicates. When multiple fragments with the same normalized content are found, the context with the highest relevance score is retained. This approach ensures that among similar or identical pieces of text, the one deemed most semantically relevant to the section is selected, thereby preserving the quality and informativeness of the context data for downstream test-case generation.

Context Organization: Once direct and related contexts are collected and de-duplicated, they are organized to enhance clarity and relevance for downstream test-case generation. Each context is annotated with metadata (including its originating section and title) and categorized based on content characteristics. For example, contexts containing state-related information are grouped separately from those discussing message formats or timer configurations. This categorization leverages keyword matching (e.g., detecting words such as 'state', 'message', or 'timer') and semantic overlap analysis. The organized contexts are then prioritized in a predefined order (with direct matches typically given the highest priority) and formatted coherently, including relevant details such as the originating section and computed relevance scores. This final structured presentation of the context ensures that all pertinent information — both within and outside the immediate section — is available for the subsequent generation of conformance test cases.

D. Test-case Generation

This is the core component of our approach. Our goal is not only to extract test cases directly corresponding to the content of a given specification section, but also to capture

dependencies and implicit requirements that span multiple sections. For this we use GPT-4o to generate structured test cases based on a carefully constructed prompt, and integrate several mechanisms to ensure the quality, diversity, and uniqueness of the generated test-cases.

Prompt Creation: The first important step is to design a comprehensive prompt to guide the LLM in generating test-cases. The prompt contains several components.

- **Test-case format:** The prompt includes a clearly defined test-case format that the LLM must adhere to. This format includes mandatory sections such as test purpose, initial conditions, trigger event, sequence of tests, Message details, and verification criteria. This ensures that every test case is uniformly structured and verifiable.
- **Contextual information:** The prompt integrates the actual text of the specification section along with the related context retrieved through the context retrieval procedure. This dual input helps the LLM ground its generation on both the explicit requirements of the section and the relevant dependencies or cross-references from elsewhere in the specification. This way, we enrich the model's understanding of the test scenario, which leads to more accurate and contextually relevant test-case generation. Moreover, by providing the exact section text along with related context, the LLM is anchored to verifiable, source-specific information. It minimizes the risk of hallucinations — where the model might otherwise generate test cases based on inferred or unrelated knowledge — and ensures that all generated cases are directly traceable to the documented requirements.
- **Scenario guidance:** The prompt includes additional instructions that direct the LLM to focus on edge cases, error conditions, alternative parameter combinations, varied message sequences, timing, and security aspects. This guidance encourages the generation of test cases that cover scenarios potentially overlooked in the primary specification.

Iterative Test-case Generation: Once the prompt is constructed, the LLM is invoked to generate a batch of test cases. First we configure the LLM with specific parameters such as temperature and maximum tokens. Afterwards, the LLM produces a response that contains multiple test-cases following the prescribed format and adhering to the guidance provided. The process repeats for a predefined number of iterations. Newly generated test cases that pass the uniqueness filtering are added to an evolving set of example cases. These examples then inform subsequent iterations, enabling the model to explore new angles and scenarios while avoiding redundancy. The system monitors the token length of the constructed prompt to ensure it remains within model limits, safeguarding against prompt truncation and preserving the integrity of the test-case generation instructions.

Similarity Filtering and Deduplication: Given that the iterative nature of the process might lead to overlapping or semantically similar test cases, a similarity filtering mechanism is incorporated. For each pair of test cases, embeddings are computed using the SentenceTransformer model `all-MiniLM-L6-v2`. The cosine similarity between these embeddings quantifies the semantic closeness of the test cases. Test cases that exceed a predefined similarity threshold (e.g., 0.9) relative to previously accepted cases are filtered out. This ensures that only unique and diverse test cases are retained.

V. Findings

From the five sections on elementary 5GS mobility management procedures, we have generated 800 conformance test-cases in under 171 minutes. A summary statistics is provided in Table I.

Authentication: We generated 150 conformance test-cases for the Primary Authentication and Key Agreement procedure in 5G. The test cases evaluate UE and network behavior under different authentication conditions, error scenarios, and security responses. A significant portion of the test cases examines authentication failures, including incorrect AUTN sequence numbers, invalid SUPI, untrusted server certificates, and synchronization failures. Other cases assess Security Mode Control, testing UE responses to unexpected EAP-failure messages, incorrect ngKSI values, and deprecated security algorithms. Additional test cases verify how the UE handles duplicate and unexpected authentication messages, emergency PDU session handling, and simultaneous authentication and network handover to non-3GPP access. There are also test-cases for timing-based scenarios, such as response time measurement for authentication result processing and timer T3520 expiry handling. A subset of test cases ensures that the UE properly manages retry mechanisms after failures, such as re-authentication using EAP-TLS and handling network-induced authentication failures.

Security Mode Control: Our approach has generated 51 test-cases for evaluating the Security Mode Control (SMC) procedure. These test cases assess how the User Equipment (UE) and Access and Mobility Management Function (AMF) handle various security-related events, including message timing, algorithm selection, authentication failures, and procedural collisions. The test cases cover a range of critical scenarios, including delayed responses, timer expirations, NAS security algorithm mismatches, network and UE capability inconsistencies, emergency service handling, and interactions with other NAS procedures. Key cases include verifying UE behavior when a Security Mode Command is received after the expiry of timer T3520, checking collision handling between Security Mode Control and Registration procedures, and ensuring proper security context synchronization during simultaneous 3GPP and non-3GPP access. From a statistical perspective, the majority of test cases focus on timing-related events (such as expired timers T3520 and T3560), message integrity validation, error handling in key exchange procedures, and horizontal key derivation challenges. Specific examples include scenarios where the UE must reject a Security Mode Command due to an invalid security algorithm selection, mismatched replayed security capabilities, or reception of an unexpected message during ongoing authentication retries.

Registration: For the registration procedure that spans over 100 pages of specifications, we have generated a grand total of 356 conformance test-cases. The test cases explore a wide range of registration-related challenges, including invalid GUTI usage, security context handling, conflicting NSSAI

Sl	Section	# Pages	# Test Cases	Time Required (sec)		
				Context Retrieval	Test-case Generation	Total
1	Primary authentication and key agreement procedure [5.4.1]	34	150	16.61	895.92	912.53
2	Security Mode Control Procedure [5.4.2]	8	51	7.51	384.87	392.38
3	Registration [5.5.1]	109	356	106.79	3228.84	3335.63
4	De-registration Procedure [5.5.2]	21	91	29.33	930.80	960.14
5	Service Request Procedure [5.6.1]	36	152	23.39	1568.97	1592.36
Total		**208**	**800**	**303.87**	**9984.94**	**10288.83**

TABLE I: Number of test-cases generated from each section, along with time required

requests, dual registration scenarios, and improper network feature support. Key cases verify how the UE reacts to NSSAI conflicts, unsupported slices, invalid SUCI formats, overlapping TAI lists, and incorrect security parameters. Special cases test UE behavior under simultaneous non-3GPP access, emergency service fallback, and back-off timer enforcement for rejected network slices. Statistically, a significant portion of the test cases focuses on authentication-free emergency registrations, handling of security mode control failures, and the impact of timing constraints on the registration process. Example cases include handling an initial registration with an invalid 5G-GUTI, ensuring that the AMF correctly rejects a request with unsupported NSSAI values, and verifying that the UE properly removes forbidden PLMNs from its equivalent PLMN list.

Deregistration: Our approach evaluates the 5G de-registration procedure (Section 5.5.2) by generating 91 conformance test cases to ensure compliance with the specification. These test cases cover UE-initiated de-registration, network-initiated de-registration, security-related de-registration scenarios, and edge cases like timer expirations and simultaneous registration events. The test cases assess how the UE and network handle de-registration requests triggered by events such as USIM removal, subscription revocation, network slice-specific restrictions, and service priority changes. Key cases include network-initiated de-registration requiring re-registration, handling of Timer T3521 expiry, conflicts between de-registration and initial registration, and emergency service exemptions. Additionally, tests validate how security failures, misconfigured UE settings, and excessive mobility updates lead to network-initiated de-registration. Statistically, the test cases focus on UE-initiated de-registration (41 cases) and network-initiated de-registration (50 cases), covering critical aspects such as authentication loss, NSSAI mismatches, non-allowed service areas, and back-off timer enforcement. Example cases include handling a network de-registration due to invalid SNPN subscription, verifying proper deletion of NAS security contexts upon deregistration, and ensuring that a UE correctly retries registration after multiple failures.

Service Request Procedure : Finally, for service request related procedure, we have generated 152 test-cases in total. These test cases focus on various service request scenarios, including abnormal timer expiry, emergency fallback, protocol errors, access barring, network congestion, paging restrictions, and multi-access PDU session handling. The test cases assess how the User Equipment (UE) and Access and Mobility Management Function (AMF) handle critical service request conditions. Key cases examine UE behavior during T3517 timer expiry, emergency service prioritization, AMF rejection due to protocol errors, and network overload handling. Other cases validate how the UE processes access barring, concurrent paging events, and service request reattempts due to non-responsive AMFs. The impact of forbidden PLMNs, incorrect security contexts, invalid protocol versions, and congestion control mechanisms is also tested. The test cases are distributed across general service requests, emergency scenarios, access control events, security integrity checks, and unexpected network failures. Specific examples include handling a service request in a forbidden PLMN, verifying network-initiated deregistration collisions, and assessing multi-access PDU session transitions.

VI. Conclusion & Future Work

In this work, we developed a novel approach using Retrieval-Augmented Generation (RAG) to automatically generate conformance test cases from 5G specifications. By combining intelligent retrieval with AI-driven generation, our method streamlines the traditionally manual and time-consuming process of test case creation. This not only improves efficiency but also ensures better alignment with official specifications.

We successfully generated nearly 800 test cases, showcasing the scalability of our approach. While we manually verified only a limited number due to resource constraints, the initial results were promising. However, we were unable to test these cases in a real 5G environment due to infrastructure limitations.

Despite these challenges, our work marks an important step toward AI-powered conformance testing in 5G networks. With further validation and improvements, this approach has the potential to make large-scale test generation faster, more accurate, and more accessible.

References

[1] S. R. Hussain, M. Echeverria, I. Karim, O. Chowdhury, and E. Bertino, "5GReasoner: A property-directed security and privacy analysis framework for 5G cellular network protocol," in *ACM Conference on Computer and Communications Security (CCS)*, 2019, pp. 669–684. [Online]. Available: https://doi.org/10.1145/3319535.3354263

[2] H. A. Quddus, M. S. Hossain, Z. Cevahir, A. Jesser, and M. N. Amin, "Enhanced vlsi assertion generation: Conforming to high-level specifications and reducing llm hallucinations with rag," in *DVCon Europe 2024; Design and Verification Conference and Exhibition Europe*, 2024, pp. 57–62.

[3] P. Sarthi, S. Abdullah, A. Tuli, S. Khanna, A. Goldie, and C. D. Manning, "Raptor: Recursive abstractive processing for tree-organized retrieval," *arXiv preprint arXiv:2401.18059*, 2024.

[4] E. Bitsikas, S. Khandker, A. Salous, A. Ranganathan, R. Piqueras Jover, and C. Pöpper, "Ue security reloaded: Developing a 5g standalone user-side security testing framework," in *Proceedings of the 16th ACM Conference on Security and Privacy in Wireless and Mobile Networks*, 2023, pp. 121–132.

[5] J. Wang, Y. Huang, C. Chen, Z. Liu, S. Wang, and Q. Wang, "Software testing with large language models: Survey, landscape, and vision," *IEEE Transactions on Software Engineering*, 2024.

[6] M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. D. O. Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman *et al.*, "Evaluating large language models trained on code," *arXiv preprint arXiv:2107.03374*, 2021.

[7] E. Nijkamp, B. Pang, H. Hayashi, L. Tu, H. Wang, Y. Zhou, S. Savarese, and C. Xiong, "Codegen: An open large language model for code with multi-turn program synthesis," in *The Eleventh International Conference on Learning Representations*.

[8] H. Trivedi, N. Balasubramanian, T. Khot, and A. Sabharwal, "Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions," in *The 61st Annual Meeting Of The Association For Computational Linguistics*, 2023.

[9] Z. Guo, S. Cheng, Y. Wang, P. Li, and Y. Liu, "Prompt-guided retrieval augmentation for non-knowledge-intensive tasks," in *Findings of the Association for Computational Linguistics: ACL 2023*, 2023, pp. 10 896–10 912.

[10] S. Liu, C. Tian, J. Sun, R. Wang, W. Lu, Y. Zhao, Y. Xue, J. Wang, and X. Du, "Conformance testing of relational dbms against sql specifications," *arXiv preprint arXiv:2406.09469*, 2024.

[11] A. Mackay, "Test Suite Augmentation using Language Models - Applying RAG to Improve Robustness Verification," in *ERTS2024*, ser. ERTS2024. Toulouse, France: ERTS2024, Jun. 2024. [Online]. Available: https://hal.science/hal-04615832

[12] A. Masoudifard, M. M. Sorond, M. Madadi, M. Sabokrou, and E. Habibi, "Leveraging graph-rag and prompt engineering to enhance llm-based automated requirement traceability and compliance checks," *arXiv preprint arXiv:2412.08593*, 2024.