CellularSpecSec-Bench: A Staged Benchmark for Evidence-Grounded Interpretation and Security Reasoning over 3GPP Specifications

Anonymous ACL submission

Abstract

Cellular networks are critical infrastructure supporting billions of worldwide users and safety- and mission-critical services. Vulnerabilities in cellular networks can therefore cause service disruption, privacy breaches, and broad societal harm, motivating growing efforts to analyze 3GPP specifications that define required device and operator behavior. While large language models (LLMs) have demonstrated the capability for reading technical documents, cellular specifications impose unique challenges: faithful interpretation of normative language, reasoning across cross-referenced clauses, and verifiable conclusions grounded in multimodal evidence such as tables and figures. To address these challenges, we propose *CellSpecSec-ARI*, a unified Adapt–Retrieve–Integrate framework for systematic understanding and standard-driven security analysis of 3GPP specifications; *CellularSpecSec-Bench*, a staged benchmark, containing newly constructed high-quality datasets with expert-verified and corrected subsets from prior open-source resources. Together, they establish an accessible and reproducible foundation for quantifying progress in specification understanding and security reasoning in the cellular network security domain.

1 Introduction

Cellular networks have become the critical infrastructure of modern society. They serve not only billions of users worldwide, but also safety- and mission-critical services such as financial transactions, public safety systems, and medical communications. Therefore, security vulnerabilities in cellular networks can have severe consequences: once exploited, they may lead to service disruptions, privacy breaches, and large-scale societal harm. Ensuring the security of mobile networks has thus long been a fundamental and enduring challenge. Beyond code review (Gomes et al., 2025; Karim et al., 2021) and fuzz testing of protocol implementations and network functions (Godefroid, 2020), increasingly efforts are now spent on analyzing 3GPP specifications, which define the required behavior of devices and operators worldwide.

However, 3GPP specification analysis is challenging. These documents are written in highly domain-specific language and interleave dense normative statements with tables and figures. Crucial details are often scattered across cross-referenced clauses and even across multiple specifications, requiring integration of evidence across documents to reconstruct end-to-end behavior.

Existing efforts on cellular-standard security analysis largely rely on conventional NLP techniques, such as sentence-pair semantic relation identification (e.g., entailment, conflict) (Chen et al., 2023; Rahman et al., 2024). While effective for their targeted tasks, such methods do not directly capture the procedural semantics and dependency structure that often determine security outcomes in standards. Recent LLM-based frameworks (Maatouk et al., 2025; Nikbakht et al., 2024; Xie et al., 2025) have begun to explore pilot architectures for interpreting 3GPP specifications, primarily focusing on whether models can answer relatively direct questions about the standards. Even with retrieval-augmented generation (RAG) (Nikbakht et al., 2024), models can still misinterpret specification-specific terminology, mishandle tables and figures, or miss cross-clause dependencies that are essential for correct security conclusions. For example, given the question "During Emergency Registration, at which step is NAS integrity established?", a retriever may surface TS 24.501 (3GP, 2022a), Clause 5.5.1 (Registration procedure) and Clause 5.4.2 (Security Mode Control procedure), but a model may fail to connect these relevant dependencies under particular preconditions. Correctly answering such questions requires multi-source, cross-clause integration; precise interpretation of normative statements; and evidence-grounded reasoning over procedures and security properties. More importantly, while prior work has identified and studied many cellular vulnerabilities, there remains a lack of a unified methodology and corresponding benchmark for evaluation. Without a comprehensive and reproducible benchmark for 3GPP specification understanding and security reasoning, it is difficult to quantify the progress and compare systems on the safety-critical reasoning demanded by both the research community and 3GPP.

To address the gap, we present *CellSpecSec-ARI* (Cellular Specification Security Analysis based on Adapt-Retrieve–Integrate) and *CellularSpecSec-Bench* in this work. *CellSpecSec-ARI* is a unified framework to solve standardized tasks in (1) interpreting complex 3GPP specifications written in domain-specific language with multimodal context, and (2) performing the challenging security analysis over cellular network standards. *CellularSpecSec-Bench* covers the core specifications in Release-17 (3GP, 2022a), (3GP, 2022b), (3GP, 2020), processing over 1,760 paragraphs, 13,800 sentences, 424 figures, and 340 tables, capturing both textual and multimodal specification semantics. The security reasoning dataset incorporates 43 real-world vulnerabilities reported in prior works (Shaik et al., 2015; Bassil et al., 2013; Kambourakis et al., 2011; Lee et al., 2009; Leong et al., 2014; Kim et al., 2019; Van Den Broek et al., 2015; Park et al., 2016; Chlosta et al., 2019; Yu and Chen, 2019; Michau and Devine, 2016; Hussain et al., 2019b; Borgaonkar et al., 2018; Cao et al., 2020; Chlosta et al., 2021; Hussain et al., 2019a; Al Ishtiaq et al., 2024; Hussain et al., 2018; Xie et al., 2025).

In summary, we make three key contributions: (1) We present the first unified framework, *CellSpecSec-ARI*, to enable systematic analysis for 3GPP specifications; (2) we establish *CellularSpecSec-Bench*, a staged benchmark to evaluate frameworks' capabilities of 3GPP specifications interpretation and security reasoning. It not only includes expert-constructed security reasoning tasks grounded in real-world vulnerabilities, but also selectively integrates compatible tasks from prior datasets after careful correctness validation; (3) we apply *CellSpecSec-ARI* to *CellularSpecSec-Bench* to provide baseline results and discuss the strengths and remaining challenges of LLM-based standards analysis.

2 Background - 3GPP Specification

The 3rd Generation Partnership Project (3GPP), an association of seven Organizational Partners, including ATIS in the U.S., ETSI in Europe, and CCSA in China, standardized the cellular network architectures and services via technical specifications (i.e., 3GPP specifications) for the operational 4G/5G networks to ongoing 6G network. Thousands of specifications are published by 3GPP to ensure global interoperability across mobile networks and devices. Mobile equipment vendors and operators are required to ensure compliance with 3GPP specifications. However, this standardization introduces systemic risks: design flaws in 3GPP specifications can propagate into mobile networks on a global scale. Moreover, the increasing complexity, frequent revisions, and evolving new generations make 3GPP specifications one of the most comprehensive and technically intricate corpora in the cellular networking domain.

3GPP specifications are published as Technical Specifications (TS) and Technical Reports (TR), and are organized into numeric series that roughly reflect functional scope. For example, TS 24 series defines core network signaling protocols for 4G/5G, and TS 38 series focuses on 5G New Radio (NR) and radio protocols.

3 Related Work

Cellular Network QA and Retrieval Benchmarks. Recent benchmarks evaluate LLM-based QA (question answering) and retrieval over telecom and 3GPP resources. TSpec-LLM (Nikbakht et al., 2024) releases a large-scale corpus of 3GPP specifications (Release 8–19). Telco-DPR (Saraiva et al., 2024) targets RAG retrieval through synthetic QA pairs. TeleQnA (Maatouk et al., 2023) provides 10K multiple-choice questions from specifications and broader sources, including research literature, and a telecom lexicon. Tele-LLMs (Maatouk et al., 2025) builds Tele-Data from specifications and an open-ended QA set for domain knowledge adaptation evaluation. Despite broad coverage, they primarily evaluate the understanding of a single document rather than task-driven reasoning, such as identifying security vulnerabilities in specifications. Moreover, these datasets rely on LLM-generated entries without expert verification, which limits their reliability for safety-critical evaluation.

Cellular Network Specific Task Datasets. Several datasets support telecom-oriented NLP tasks over specifications. SPEC5G (Karim et al., 2023) builds a 5G-focused corpus from specifications, augmented with content scraped from blogs and technical forums, and defines two tasks: security-related text classification and specification summarization. CellularLint (Rahman et al., 2024) and ConTester (Chen et al., 2023) derive sentence pairs from 3GPP specifications and annotate fine-grained semantic relations to support Natural Language Inference (NLI)-style reasoning and inconsistency detection. While valuable, these datasets have notable limitations: (1) they primarily measure sentence-/paragraph-level understanding (classification, summarization, or pairwise relations) rather than standards-driven procedural reasoning that requires integrating evidence across multiple clauses, tables, and figures; (2) they do not explicitly evaluate evidence-grounded performance, which is critical for complex reasoning; (3) their security focus is indirect. Classifying security-relevant text or detecting sentence-level inconsistencies does not require reasoning about how end-to-end protocol behavior in 3GPP specifications and adversarial exploitation conditions.

4 *CellSpecSec-ARI*

In this section, we present a unified framework, *CellSpecSec-ARI* (Cellular Specification Security Analysis based on Adapt-Retrieve–Integrate), to enable the reproducible and systematic security analysis for 3GPP specifications. This three-stage framework is inspired by the Dreyfus model of skill acquisition (Dreyfus et al., 1986): the *Adapt* stage corresponds to novice-level rule-based comprehension, *Retrieve* aligns with competent, evidence-guided decision making, and *Integrate* reflects expert-level synthesis across multiple sources.

Flex Framework. Importantly, *CellSpecSec-ARI* specifies what capabilities each layer should provide, rather than enforcing a particular algorithmic design. Thus, each layer is implementation-agnostic and can be instantiated with different parsers, retrievers, or knowledge representations for advanced performance. *CellSpecSec-ARI* follows this three-layer architecture, composed by three modules as shown in Figure 1: ***SpecFusion*** (Specification Fusion), ***SpecRAG*** (Specification Retrieval-augmented Generation), and ***SpecReasoning*** (3GPP Specifications Knowledge Integration Reasoning), which progressively function as the abilities to *adapt* 3GPP specifications, *retrieve* knowledge from different standards, and *integrate* these knowledge into conducting security analysis on 3GPP standards.

4.1 Adaption - *SpecFusion*

The first module, *SpecFusion*, is to enable LLMs, which are the core of *CellSpecSec-ARI*, to *adapt* to the domain knowledge in 3GPP specifications. LLMs trained on open-domain corpora often struggle to interpret the stylistic and structural domain-specific conventions of technical standards. In particular, they tend to (1) misinterpret domain-specific text for hallucinations and (2) mishandle multimodal contents, including tables and figures for depicting concepts that are hard to describe in natural language (e.g., transitions between multiple steps).

Formally, each 3GPP specification can be decomposed into 3 components $\mathcal{C} = \mathcal{C}_{\text{text}} \cup \mathcal{C}_{\text{table}} \cup \mathcal{C}_{\text{figure}}$. Textual components ($\mathcal{C}_{\text{text}}$) correspond to normative paragraphs written in natural language that describe procedures, operational conditions, definitions, or constraints. In 3GPP specifications, these paragraphs define how entities behave in different conditions and settings. Such text chunks provide *CellSpecSec-ARI* with fine-grained exposure to the wording and normative style of standardization text. Table components ($\mathcal{C}_{\text{table}}$) represent normative tables that encode structured relationships among parameters, states, timers, configuration options, and actions. Including tabular chunks in the adaptation process allows *CellSpecSec-ARI* to learn how rules and mappings are expressed in the tables of 3GPP specifications. Figure components ($\mathcal{C}_{\text{figure}}$) correspond to diagrams such as signaling flows, state machines, and message or information-element formats. Including figure components in the adaptation process provides *CellSpecSec-ARI* with the ability to interpret protocol information that is expressed graphically in specifications.

Multimodal adaptation approach. Instead of applying a specific encoder for each modality, *SpecFusion* transforms non-textual components into a unified textual representation. Specifically, tables and figures in 3GPP specifications are extracted and converted into structured JSON descriptions as shown in Appendix B. Together, these component types form the unified component space $\mathcal{C}$ used by *SpecFusion* for multi-ability adaptation. Rather than removing multimodal components from 3GPP specifications in many prior works (Karim et al., 2023; Rahman et al., 2024), *SpecFusion* enables

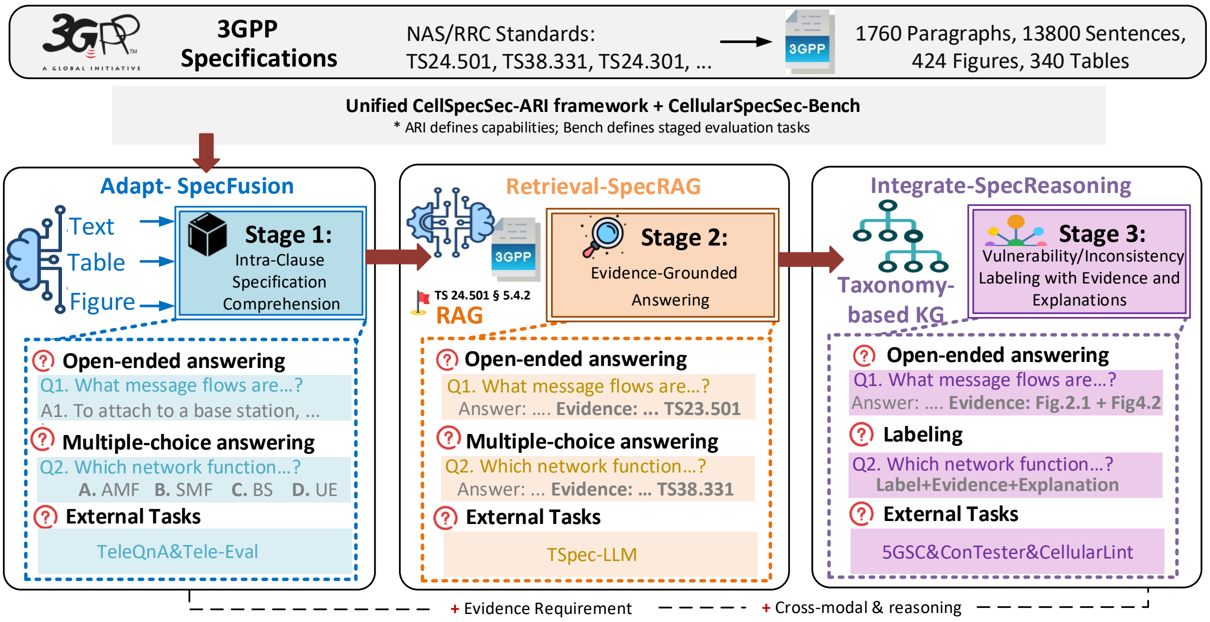

Figure 1: Overview of *CellSpecSec-ARI* and *CellularSpecSec-Bench*

CellSpecSec-ARI to adapt to multimodal knowledge in standards.

4.2 Retrieval - *SpecRAG*

Since each specification spans hundreds to thousands of pages and there are thousands of specifications for operational cellular networks, it is difficult to simply rely on the models' memory to generate trustworthy outputs for further decision-making. Thus, *SpecRAG* performs retrieval-augmented reasoning that endorses generated outputs via the direct evidence from 3GPP specifications.

Formally, for each query q, the *SpecRAG* researches and returns a set of relevant evidence chunks, from its RAG database, developed using the multimodal components from *SpecFusion*. This set is denoted as $E(q) = \{(c_j, \ell_j)\}_{j=1}^{m}$, where each chunk c_j corresponds to a subclause, table, or figure, and ℓ_j is its citation label (e.g., clause number). This set is further provided to the framework to ensure each output can be explicitly grounded in 3GPP specifications.

The prototype *SpecRAG* uses a hybrid retrieval strategy. The sparse component $s_{\text{sp}}(q, c)$ is computed using a BM25 ranker that captures exact keyword matches between the query q and candidate chunk c. The dense component $s_{\text{de}}(q, c)$ is obtained by encoding the query and each chunk into TF–IDF vectors, projecting them into a lower-dimensional latent space via truncated SVD, and computing cosine similarity between the resulting vectors. These dense embeddings approximate semantic similarity using statistical co-occurrence patterns and are efficiently indexed for similarity search. The final relevance score is a weighted combination:

$$s(q, c) = \alpha\, s_{\text{sp}}(q, c) + (1-\alpha)\, s_{\text{de}}(q, c), \alpha \in [0, 1],$$

where α is tuned on a small validation set to balance lexical precision and semantic recall. The top-k highest-scoring chunks form the Retrieved Evidence context supplied to the model. See Appendix E for the selection analysis of k.

Sample query and output. When queried with "During Emergency Registration, when is NAS integrity established?", *SpecRAG* retrieves multimodal evidence including TS 24.501, Clause 5.5.1 Registration procedure and Clause 5.4.2 Security Mode Control procedure. The former describes how the UE initiates registration, while the latter specifies when NAS integrity protection is activated. Concatenating these clauses yields the evidence-grounded context, from which *CellSpecSec-ARI* generates the trustworthy answer: "NAS integrity is established by the Security Mode Control procedure initiated by the AMF during Registration." accompanied by explicit citations: TS 24.501, Clauses 5.5.1 and 5.4.2.

4.3 Integration - *SpecReasoning*

While prior two modules expose 3GPP specifications to *CellSpecSec-ARI* and enable it to retrieve multimodal evidence to generate evidence-grounded trustworthy outputs, the pieces of knowledge from the clause, table, and figure have not been integrated together for performing complex reasoning. Likewise, cross-modal relationships among textual, tabular, and visual evidence are often obscured, since tabular semantics depend on multi-row conditions and figures encode procedural or causal flows that cannot be captured through text retrieval alone. To overcome these limitations and enable relational reasoning, we transform the multimodal outputs of *SpecRAG* into structured, interlinked representations. Specifically, we convert them into a taxonomy-based knowledge graph (KG) that explicitly models the structural and procedural dependencies among 3GPP specification entities.

Entity types for taxonomy-based KG. We first construct an entity taxonomy that reflects the hierarchical organization of 3GPP specifications. By systematically analyzing each clause, including all multimodal components, we identify the recurring entities from the majority of clauses and summarize a set of entity types as shown in Table 2: signaling procedures, messages, information elements, identifiers, timers, states, conditions, and system-level properties. They have consistent meanings across 3GPP specifications and form the basic entity vocabulary used in the taxonomy-based KGs.

Relation types for taxonomy-based KG. We next derive the relation schema that shows how these entities interact in 3GPP specifications. Two complementary families of relations are defined. **Core relations** $\mathcal{R}_{\text{core}}$ represent the structural dependencies that govern protocol execution logic. These relations describe, for example, how a procedure is decomposed into signaling steps, how a message contains a set of information elements, how the execution of a step requires the device to be in a specific protocol state, and how timers are started, stopped, or reset during the procedure. They capture the functional organization of the specification independently of whether the behavior has security relevance. **Security relations** $\mathcal{R}_{\text{sec}}$ capture dependencies that explicitly relate to protection requirements and security outcomes. These relations encode whether a message or information element must be integrity protected or ciphered, whether an exception exists in which an otherwise protected object may appear in unprotected form, whether the processing of a message is permitted only after an integrity check is validated, and whether the reception of a message triggers a network action or induces a transition in the protocol state machine. Table 3 summarizes the full inventory of relation types used in *SpecReasoning*.

Building taxonomy-based KG. Given the entity and relation types, *SpecReasoning* instantiates the KGs with the retrieved evidence. Formally, a *SpecGraph* instance is a labeled multi-relational graph $G = (V, E, \tau, \rho, \lambda)$,, where V is the set of nodes and $E \subseteq V \times V$ is the set of directed edges. Each node in V is assigned an entity type $\tau(v)$ chosen from the entity table Procedure, Message, . . . , Property. Each edge in E is assigned a relation type $\rho(e)$ drawn from $\mathcal{R}$core$\cup\mathcal{R}$sec. λ stores the provenance metadata for each node and edge, including its defining clause, table, or figure identifier (e.g., "TS 24.501, Clause X.Y.Z"). See Appendix D for the example of a taxonomy-based KG.

Taxonomy-based KG enhanced reasoning. We use the taxonomy-based KGs as the additional retrievable evidence source. Each clause-level KG is converted into a text block, which includes its clause identifier, the extracted entities, relations, and the original text in specifications. The same hybrid retrieval strategy used in *SpecRAG* is applied over the KG-derived text blocks to retrieve top-k relevant KG blocks to help with the reasoning correctness.

5 *CellularSpecSec-Bench*

We develop two dataset groups:

Original datasets. These datasets are developed and constructed from core control-plane 3GPP specifications that govern signaling procedures, state transitions, and protection mechanisms in 4G and 5G networks. They are also where many real-world security issues originate, including downgrade (Shaik et al., 2015; Bassil et al., 2013; Kambourakis et al., 2011; Lee et al., 2009; Leong et al., 2014; Hussain et al., 2019b, 2018; Kim et al., 2019; Cao et al., 2020), replay (Hussain et al., 2019b; Al Ishtiaq et al., 2024), and denial-of-service (Bassil et al., 2013; Kambourakis et al., 2011; Lee et al., 2009; Leong et al., 2014) attacks, often due to insecure transitions, missing integrity protection, or ambiguous normative requirements. Appendix A lists details of source specifications.

Verified and corrected datasets. To broaden coverage and support additional task formats, *CellularSpecSec-Bench* selectively integrates state-of-the-art benchmarks from the telecom and cellular NLP literature. It includes TeleQnA (Maatouk et al., 2023), Telco-DPR (Saraiva et al., 2024), TSpec-LLM (Nikbakht et al., 2024), and Tele-LLMs (Maatouk et al., 2025) for assessing how models understand 3GPP specifications; it incorporates SPEC5G (Karim et al., 2023) for 5G security classification; ConTester (Chen et al., 2023) for semantic-equivalence sentence pairs for 4G; CellularLint (Rahman et al., 2024) for NLI-style sentence pairs for 4G/5G. We exclude datasets that are not centered on specification interpretation and standards-driven reasoning (e.g., TeleMath (Colle et al., 2025), which primarily targets mathematical problem solving).

Notably, external datasets are not always expert-validated. For example, TeleQnA, TSpec-LLM, and Tele-LLMs are mainly constructed by LLM-generated entries. We observe that a significant number of entries are inaccurate from an expert's perspective. Consequently, *CellularSpecSec-Bench* does not ingest these datasets verbatim. Instead, we curate, verify, and correct the integrated subsets, ensuring that each retained instance is consistent with specification evidence and intended task definition. Through this selective integration and expert verification, *CellularSpecSec-Bench* combines standards-grounded control-plane reasoning with broader telecom knowledge, resulting in a benchmark that is more comprehensive and reliable than prior datasets in this domain.

5.1 Task Types

We formalize the tasks involved in comprehending 3GPP specifications and identifying security design vulnerabilities into three types. Examples for each task type are provided in Appendix G.

Open-ended answering. It is formalized as

$$f_{\text{open}} : (Q) \to \begin{cases} A, & \text{answer only,} \\ (A, E), & \text{answer with evidence,} \end{cases}$$

where Q is a query relevant to the 3GPP specification corpus, A is a textual answer, and E is an optional evidence set consisting of clause numbers, table identifiers, or figure labels.

Multiple-choice answering. It is defined as

$$f_{\text{mc}} : (Q, \mathcal{O}) \to (A, E),$$

where $\mathcal{O}$ is a fixed set of candidate options, and exactly one option is compliant with the 3GPP specification. Evaluation is performed by comparing both the model-selected answer and its supporting evidence to the gold annotations, yielding answer accuracy and evidence accuracy.

Vulnerability/inconsistency Labeling. Formally, it is defined as

$$f_{\text{vul}} : S \to (\mathbf{y}, E), \qquad \mathbf{y} = (y_{\text{label}}, \mathbf{y}_{\text{multi-label}}),$$

where S is a normative sentence and y is the label assigned for S. There are two main types of labels: y_{label} indicates whether the sentence contains a vulnerability or contradicts other parts of specifications, reflecting a semantic or procedural inconsistency; $\mathbf{y}_{\text{multi}}$ is a multi-label (e.g., denial-of-service, replay, downgrade, privacy/tracking, spoofing, authentication bypass, etc.).

5.2 Task Construction

We follow the Adapt–Retrieve–Integrate progression to develop a three-stage benchmark, where each stage introduces additional requirements. Stage 1 targets clause-local understanding. Each question is answerable from a single clause and does not require integrating information across clauses, tables, or figures. Stage 2 adds an explicit evidence grounding requirement to enforce verifiability, requiring models to cite the supporting clause/table/figure identifiers. Stage 3 consolidates the remaining challenges by requiring cross-clause integration and security-aware interpretation. Table 5 in Appendix F summarizes the coverage of the external benchmarks incorporated in each stage, including their task formats and the corresponding capabilities they assess. *CellularSpecSec-Bench*, including the verified and corrected tasks from other datasets, is publicly available on the website (Anonymous, 2026).

5.3 Task Scopes

CellularSpecSec-Bench is designed to evaluate not only whether models can read 3GPP specifications, but also whether they can perform standards-driven security analysis. Accordingly, questions span two scopes. **Specification understanding** tasks target definitions, message formats, parameter constraints, and normative requirements, and test whether a model can correctly interpret standard descriptions. **Security analysis** tasks go further by requiring models to reason about how specification-defined procedures and protections can fail under adversarial settings, such as when a message lacks integrity protection, a state transition can be triggered without authentication, or a precondition is underspecified. These tasks evaluate vulnerability-oriented reasoning and require evidence-grounded conclusions linked to the relevant multimodal context.

5.4 *CellularSpecSec-Bench* Statistics

In Stage 1, there are 4,500 questions, including open-ended answering and multiple-choice questions. Two verified and corrected tasks from Tele-QnA and Tele-Eval contribute a total of 1,000 questions. Stage 2 tasks require returning the evidence from specifications. There are a total of 4,500 questions. Stage 2 also incorporates the verified and corrected external dataset from TSpec-LLM, contributing 500 questions. Stage 3 focuses on cross-clause integration and security-aware interpretation. There are 222 Cross-Clause Question Answering (CCQA) and 747 Table-and-Figure Question Answering (TFQA). For security analysis, Stage 3 includes 300 vulnerability/inconsistency labeling instances with complete evidence and explanations. External verified and corrected datasets from 5GSC, ConTester, and CellularLint contribute a total of 1,454 sentence-level questions, 320 sentence-pair questions, and 55 questions. Details of tasks are elaborated in Appendix H.

6 Evaluation

6.1 Baseline

We deploy *CellSpecSec-ARI* using a remote server on Jetstream2 (jet, 2025) with a H100 and access the DeepSeek-V3.2-Exp (DeepSeek API Docs, 2025) API for all experiments. To ensure deterministic and reproducible results, we set the model temperature to 0. We evaluate two configurations: (1) Base, which uses the raw DeepSeek-V3.2-Exp model without any *CellSpecSec-ARI* modules, and (2) *CellSpecSec-ARI*, which augments the same underlying model with the pipeline enabled module by module. Additional implementation details are in Appendix C.

6.2 Metrics

Open-ended answering. We evaluate answer correctness using an LLM-as-judge protocol with a three-level rubric: 2 (fully correct), 1 (partially correct), and 0 (incorrect). The judge is provided with (i) the question, (ii) the model's answer, and (iii) the gold reference answer; for tasks that require evidence, it is provided with (iv) the gold evidence, which is clause numbers, table/figure identifiers, and their extracted text. The judge model is vanilla DeepSeek-V3.2-Exp, which is instructed to base its decision only on the provided specification evidence and to ignore any external knowledge. We run the judge with temperature 0 and a single decoding to ensure deterministic scoring. To reduce bias, we hide system identities and randomize the presentation order of candidate answers. The full prompt template and rubric are included in Appendix N. The method to evaluate evidence correctness is described in Appendix M.

Multiple-choice answering. For this task, we report accuracy, since each question has exactly one specification-compliant option. When evidence is required, we also report evidence correctness.

Labeling tasks. For vulnerability/inconsistency labeling, we evaluate outputs hierarchically. For the binary label (vulnerable/inconsistent vs. non-vulnerable), we report positive-class F1. For the multi-label vulnerability categories, we evaluate only on instances with gold positives and report Micro-F1 and Macro-F1. For the evidence and explanations, we treat evidence correctness as the primary metric, since the task requires retrieving the complete set of clauses/tables/figures that collectively justify why a vulnerability arises. We score the explanation using the aforementioned LLM-as-judge approach.

External Tasks. For verified and corrected external tasks, we apply the same task-specific metrics after curation: accuracy for multiple-choice, the LLM-as-judge rubric for open-ended answering, and the above classification metrics for labeling tasks.

6.3 Overall Performance

Table 1 reports the performance of *CellSpecSec-ARI* on *CellularSpecSec-Bench*. We highlight four key observations. **(1) Performance gains from *CellSpecSec-ARI* modules.** The base model performs poorly on most tasks, indicating that the base model struggles with the normative style, precision, and logic in 3GPP specifications. Incorporating *CellSpecSec-ARI* modules substantially improves performance, demonstrating that the domain adaptation and structured reasoning improve standards comprehension and reasoning capability. With *CellSpecSec-ARI* enabled, we observe better performance on all tasks. We report module-level gains in Appendix O. **(2) Effective evidence grounding via retrieval augmentation.** The base model has poor performance when tasks require evidence. Enabling *SpecRAG* improves both answer quality

Task	Metric	BASE	*CellSpecSec-ARI*
Stage 1: Intra-Clause Specification Comprehension			
Extractive QA (EQA)	Score=2	36%	97.75%
Abstractive QA (AQA)	Score=2	34%	97%
MCQA	Accuracy	76%	100%
TeleQnA (external)	Accuracy	90%	96%
Tele-Eval (external)	Score=2	38%	97.4%
Stage 2: Evidence-Grounded Answering			
EQA-E	Score=2 / Evidence Correct	39.2% / 0%	96.8% / 96.4%
AQA-E	Score=2 / Evidence Correct	30% / 0%	96.8% / 96%
MCQA-E	Accuracy / Evidence Correct	79.4% / 0%	100% / 100%
TSpec-LLM (external)	Accuracy / Evidence Correct	76% / 0%	94% / 92%
Stage 3: Vulnerability/Inconsistency Labeling with Evidence and Explanations			
CCQA	Score=2 / Evidence Correct	20% / 0%	95.06% / 91.36%
TFQA	Score=2 / Evidence Correct	17.28% / 0%	98% / 95.33%
Vulnerability / Inconsistency Labeling	Binary (F1) / Multi (Micro-F1 / Macro-F1)	80% / 39.02% / 38.81%	93.05% / 70.59% / 89.33%
Evidence and Explanations	Evidence & Explanations Correctness	10%	88%
5GSC (external)	Accuracy	46%	100%
ConTester (external)	Accuracy	96%	100%
CellularLint (external)	Accuracy	36%	100%

Table 1: Overall Task Performance Comparison of Base LLM and *CellSpecSec-ARI*

and evidence correctness by retrieving and anchoring responses to relevant clauses, tables, and figures, leading to large gains on evidence-grounded tasks. **(3) Strong integration reasoning for security analysis.** Stage 3 represents the most challenging setting, requiring cross-clause integration, vulnerability/inconsistency labeling with complete evidence and explanations. In this difficult setting, strong performance is observed when *CellSpecSec-ARI* is applied. **(4) Robust improvements on verified and corrected external tasks.** On all verified and corrected external tasks, *CellSpecSec-ARI* consistently outperforms the base model, showing that its benefits generalize beyond our newly constructed datasets.

6.4 Error Analysis

We analyze the incorrect answers and discuss the potential improvements.

Answer–evidence mismatch. *CellSpecSec-ARI* sometimes generated a correct answer but cited an incorrect clause/table/figure as the evidence. This typically occurs because 3GPP specifications contain multiple semantically related components that share near-identical terminology (e.g., repeated definitions across procedures), making it easy for retrieval a plausible but incorrect reference. This mismatch can be mitigated by strengthening evidence selection and verification, such as using clause-aware reranking that jointly optimizes answer relevance and citation, or adding a citation consistency check that validates whether the cited unit explicitly contains the key conditions/terms used in the answer.

Evidence incompleteness in security reasoning and labeling. For security-focused tasks, vulnerability/inconsistency labeling, the dominant failures stem from incomplete evidence, rather than purely incorrect security reasoning. Explanations are directionally correct but incomplete, and the cited evidence set is missing one or more critical dependencies. These patterns emphasize that security reasoning in standards is compositional. To mitigate this, we can add a completeness verifier that rejects conclusions or causal explanations unless the assembled evidence jointly supports the claim.

7 Conclusion

This work addresses a key gap in cellular security research: the lack of a unified methodology and reproducible benchmark for evaluating 3GPP specification understanding and standards-driven security reasoning. We introduce *CellSpecSec-ARI*, a unified framework that supports domain adaptation to 3GPP specifications, retrieval of multimodal evidence with verifiable citations, and integration reasoning for security analysis. We establish *CellularSpecSec-Bench* a staged benchmark spanning clause-local comprehension, evidence-grounded answering, and cross-clause, security-relevant reasoning, built from core specifications and complemented with carefully verified and corrected subsets from prior datasets.

Limitation

We acknowledge several limitations in this study and aim to address them in future work. First, *CellularSpecSec-Bench* focuses on a limited set of 3GPP specifications in Release 17. While this design scope enables controlled and reproducible evaluation, it does not cover the full breadth of the 3GPP corpus or reflect the ongoing evolution of specifications across releases. Second, although *CellularSpecSec-Bench* incorporates real-world vulnerabilities reported in prior work to construct security-focused tasks, its coverage is inherently limited to publicly documented and curated cases. Third, for tasks that require explicit evidence sets, *CellularSpecSec-Bench* prioritizes verifiability by enforcing strict matching against clause, table, and figure identifiers, and, in advanced settings, by requiring complete evidence that jointly supports a conclusion. While this strict evaluation aligns with the goal of trustworthy specification reasoning, it may under-credit partially correct evidence or alternative but valid supporting sets. Last, while this work focuses on establishing the first unified framework *CellSpecSec-ARI* and developing the reproducible *CellularSpecSec-Bench*, our experiments apply DeepSeek-V3.2-Exp as the base model. Evaluating a broader range of base models and system variants is an important direction for future work and may reveal additional insights.

Ethical Statement

Our work studies automatic security reasoning over 3GPP specifications and introduces a benchmark related to procedures and security issues. Our framework and benchmark aim to improve defensive monitoring of standards and implementations. Our datasets are derived from 3GPP technical specifications and from publicly reported vulnerabilities in academic and industry works. The dataset does not involve human subjects or personal data. We will release it under the MIT license, requiring users to conform its terms. We encourage the use of our benchmark for research purposes, and the authors disclaim liability of any misuse.

References

2020. *Universal Mobile Telecommunications System (UMTS); LTE; 5G; Non-Access-Stratum (NAS) protocol for Evolved Packet System (EPS); Stage 3 (3GPP TS 24.301 version 16.8.0 Release 16)*. 3rd Generation Partnership Project (3GPP). [Online]. Available: http://www.3gpp.org/dynareport/24301.htm.

2022a. *5G; Non-Access-Stratum (NAS) protocol for 5G System (5GS); Stage 3 (3GPP TS 24.501 version 17.8.0 Release 17)*. 3rd Generation Partnership Project (3GPP). [Online]. Available: http://www.3gpp.org/dynareport/24501.htm.

2022b. TS 38.331: Nr; radio resource control (RRC); protocol specification. Technical Report TS 38.331 V17.0.0, 3rd Generation Partnership Project (3GPP). Release 17.

2025. Jetstream2 launches large language model (llm) inference service. Online; Indiana University Jetstream2 News & Events. Accessed on 28 August 2025.

Abdullah Al Ishtiaq, Sarkar Snigdha Sarathi Das, Syed Md Mukit Rashid, Ali Ranjbar, Kai Tu, Tianwei Wu, Zhezheng Song, Weixuan Wang, Mujtahid Akon, Rui Zhang, and 1 others. 2024. Hermes: Unlocking security analysis of cellular network protocols by synthesizing finite state machines from natural language specifications. In *33rd USENIX Security Symposium (USENIX Security 24)*, pages 4445–4462.

Anonymous. 2026. CellularSpecSec-Bench: A three-stage benchmark for cellular specification understanding and security reasoning. Accessed: 2026-01-01.

Ramzi Bassil, Imad H. Elhajj, Ali Chehab, and Ayman Kayssi. 2013. Effects of signaling attacks on lte networks. In *2013 27th International Conference on Advanced Information Networking and Applications Workshops*, pages 499–504.

Ravishankar Borgaonkar, Lucca Hirschi, Shinjo Park, and Altaf Shaik. 2018. New privacy threat on 3g, 4g, and upcoming 5g aka protocols. *Cryptology ePrint Archive*.

Jin Cao, Maode Ma, Hui Li, Ruhui Ma, Yunqing Sun, Pu Yu, and Lihui Xiong. 2020. A survey on security aspects for 3gpp 5g networks. *IEEE Communications Surveys & Tutorials*, 22(1):170–195.

Yi Chen, Di Tang, Yepeng Yao, Mingming Zha, XiaoFeng Wang, Xiaozhong Liu, Haixu Tang, and Baoxu Liu. 2023. Sherlock on specs: Building LTE conformance tests through automated reasoning. In *32nd USENIX Security Symposium (USENIX Security 23)*, pages 3529–3545, Anaheim, CA. USENIX Association.

Merlin Chlosta, David Rupprecht, Thorsten Holz, and Christina Pöpper. 2019. Lte security disabled: misconfiguration in commercial networks. In *Proceedings of the 12th conference on security and privacy in wireless and mobile networks*, pages 261–266.

Merlin Chlosta, David Rupprecht, Christina Pöpper, and Thorsten Holz. 2021. 5g suci-catchers: Still catching them all? In *Proceedings of the 14th ACM*

Conference on Security and Privacy in Wireless and Mobile Networks, pages 359–364.

Vincenzo Colle, Mohamed Sana, Nicola Piovesan, Antonio De Domenico, Fadhel Ayed, and Merouane Debbah. 2025. Telemath: A benchmark for large language models in telecom mathematical problem solving. *Preprint*, arXiv:2506.10674.

DeepSeek API Docs. 2025. Introducing deepseek-v3.2-exp. `https://api-docs.deepseek.com/news/news250929`. Accessed: 2025-09-29.

Hubert L. Dreyfus, Stuart E. Dreyfus, and Tom Athanasiou. 1986. *Mind over machine: the power of human intuition and expertise in the era of the computer*. The Free Press, USA.

Patrice Godefroid. 2020. Fuzzing: Hack, art, and science. *Communications of the ACM*, 63(2):70–76.

Diego Gomes, Eduardo Felix, Fernando Aires, and Marco Vieira. 2025. Static code analysis for iot security: A systematic literature review. *ACM Comput. Surv.*, 58(3).

Syed Hussain, Omar Chowdhury, Shagufta Mehnaz, and Elisa Bertino. 2018. Lteinspector: A systematic approach for adversarial testing of 4g lte. In *Network and Distributed Systems Security (NDSS) Symposium 2018*.

Syed Rafiul Hussain, Mitziu Echeverria, Omar Chowdhury, Ninghui Li, and Elisa Bertino. 2019a. Privacy attacks to the 4g and 5g cellular paging protocols using side channel information. *Network and distributed systems security (NDSS) symposium2019*.

Syed Rafiul Hussain, Mitziu Echeverria, Imtiaz Karim, Omar Chowdhury, and Elisa Bertino. 2019b. 5greasoner: A property-directed security and privacy analysis framework for 5g cellular network protocol. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, pages 669–684.

Georgios Kambourakis, Constantinos Kolias, Stefanos Gritzalis, and Jong Hyuk Park. 2011. Dos attacks exploiting signaling in umts and ims. *Computer Communications*, 34(3):226–235.

Imtiaz Karim, Syed Rafiul Hussain, and Elisa Bertino. 2021. Prochecker: An automated security and privacy analysis framework for 4g lte protocol implementations. In *2021 IEEE 41st International Conference on Distributed Computing Systems (ICDCS)*, pages 773–785. IEEE.

Imtiaz Karim, Kazi Samin Mubasshir, Mirza Masfiqur Rahman, and Elisa Bertino. 2023. Spec5g: A dataset for 5g cellular network protocol analysis. *Preprint*, arXiv:2301.09201.

Hongil Kim, Jiho Lee, Eunkyu Lee, and Yongdae Kim. 2019. Touching the untouchables: Dynamic security analysis of the lte control plane. In *2019 IEEE Symposium on Security and Privacy (SP)*, pages 1153–1168. IEEE.

Patrick PC Lee, Tian Bu, and Thomas Woo. 2009. On the detection of signaling dos attacks on 3g/wimax wireless networks. *Computer Networks*, 53(15):2601–2616.

Wai Kay Leong, Aditya Kulkarni, Yin Xu, and Ben Leong. 2014. Unveiling the hidden dangers of public ip addresses in 4g/lte cellular data networks. In *Proceedings of the 15th Workshop on Mobile Computing Systems and Applications*, pages 1–6.

Ali Maatouk, Kenny Chirino Ampudia, Rex Ying, and Leandros Tassiulas. 2025. Tele-llms: A series of specialized large language models for telecommunications. *Preprint*, arXiv:2409.05314.

Ali Maatouk, Fadhel Ayed, Nicola Piovesan, Antonio De Domenico, Merouane Debbah, and Zhi-Quan Luo. 2023. Teleqna: A benchmark dataset to assess large language models telecommunications knowledge. *Preprint*, arXiv:2310.15051.

Benoit Michau and Christophe Devine. 2016. How to not break lte crypto. In *ANSSI Symposium sur la sécurité des technologies de l'information et des communications (SSTIC)*.

Rasoul Nikbakht, Mohamed Benzaghta, and Giovanni Geraci. 2024. Tspec-llm: An open-source dataset for llm understanding of 3gpp specifications. *Preprint*, arXiv:2406.01768.

Shinjo Park, Altaf Shaik, Ravishankar Borgaonkar, and Jean-Pierre Seifert. 2016. White rabbit in mobile: Effect of unsecured clock source in smartphones. In *Proceedings of the 6th Workshop on Security and Privacy in Smartphones and Mobile Devices*, pages 13–21.

Mirza Masfiqur Rahman, Imtiaz Karim, and Elisa Bertino. 2024. CellularLint: A systematic approach to identify inconsistent behavior in cellular network specifications. In *33rd USENIX Security Symposium (USENIX Security 24)*, pages 5215–5232, Philadelphia, PA. USENIX Association.

Thaina Saraiva, Marco Sousa, Pedro Vieira, and António Rodrigues. 2024. Telco-dpr: A hybrid dataset for evaluating retrieval models of 3gpp technical specifications. *Preprint*, arXiv:2410.19790.

Altaf Shaik, Ravishankar Borgaonkar, N Asokan, Valtteri Niemi, and Jean-Pierre Seifert. 2015. Practical attacks against privacy and availability in 4g/lte mobile communication systems. *arXiv preprint arXiv:1510.07563*.

Fabian Van Den Broek, Roel Verdult, and Joeri De Ruiter. 2015. Defeating imsi catchers. In *Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security*, pages 340–351.

Ke Xie, Xingyi Zhao, Yiwen Hu, Munshi Saifuzzaman, Wen Li, Shuhan Yuan, Tian Xie, and Guan-Hua Tu. 2025. Cellsecinspector: Safeguarding cellular networks via automated security analysis on specifications. *arXiv preprint arXiv:2512.24682*.

Chuan Yu and Shuhui Chen. 2019. On effects of mobility management signalling based dos attacks against lte terminals. In *2019 IEEE 38th International Performance Computing and Communications Conference (IPCCC)*, pages 1–8.

A Details of Specifications

◇ **TS 24.501 version 17.7.1 (Release 17) (3GP, 2022a).** The 5G NAS (Non-Access Stratum) protocol specification, which governs registration, authentication, security mode control, mobility management, and emergency procedures. It serves as the primary locus of end-to-end signaling protection between the UE and the core network.
◇ **3GPP TS 38.331 version 17.0.0 (Release 17) (3GP, 2022b).** The 5G RRC (Radio Resource Control) protocol, defining radio connection establishment, reconfiguration, and security activation within the access stratum, crucial for understanding where and how radio-layer security is triggered and maintained.
◇ **TS 24.301 version 17.6.0 (Release 17) (3GP, 2020).** The 4G NAS specification, providing continuity with legacy EPS procedures and exposing downgrade and compatibility behaviors that remain relevant in dual-connectivity and inter-RAT (Radio Access Technology) scenarios.

B Extracting multimodal content from 3GPP specifications

Figure 2, 3, 4, 5 present the key prompts used to extract texts, tables, and figures from 3GPP specifications into the structured JSON descriptions.

Clause-level text extraction

You are an expert in reading 3GPP specification documents.

You will be given raw text extracted from a 3GPP specification PDF.

Your task is to segment the document into individual clauses.

For each clause:
- Identify the clause number (e.g., "5.6.1.4", "4.17", "8.3.6.12").
- Identify the clause title.
- Extract the full clause text belonging to that clause, including:
  - Normative statements
  - Sub-paragraphs
  - Lists and conditions
- Do NOT merge text across different clauses.
- Do NOT summarize or paraphrase.
- Preserve the original wording exactly.

Return the result as a JSON array in the following format:

```
[
 {
  "clause_id": "",
  "clause_title": "",
  "text": ""
 }
]
```

Return ONLY valid JSON.

Figure 2: Clause-level text extraction

Figure and table identification extraction

You are given text lines extracted from a 3GPP specification page.

Your task is to identify figure and table definitions.

For each detected figure or table:
- Extract the figure/table identifier (e.g., "Figure 5.3.7.1", "Table 10.2.2").
- Extract the full title following the identifier.
- Record the page number.

Rules:
- Do NOT paraphrase titles.
- Remove only trivial noise such as "(continued)".
- Keep identifiers exactly as in the document.

Return the result as a JSON list:

```
[
 {
  "id": "",
  "title": "",
  "page": ""
 }
]
```

Return ONLY valid JSON.

Figure 3: Figure and table identification extraction

Figure content extraction

You will receive an image of a figure from 3GPP TS 24.301, TS 24.501, or TS 38.331.

IMPORTANT:
- The correct figure_id MUST EXACTLY match the number in the file name.
- Always use the number from the file name as figure_id.
- Do NOT return multiple figure IDs.

1) First classify the figure into exactly one of:
- "STATE_TRANSITION"
- "PROCEDURE_FLOW"
- "BIT_LAYOUT"
- "OTHER"

2) For figures of type "STATE_TRANSITION" or "PROCEDURE_FLOW":
- Treat each arrow as one transition.
- Arrow tail = "from" state
- Arrow head = "to" state
- Text on or near the arrow = transition condition or message name
- Never reverse arrow directions.

3) Extract content into the following JSON schema:

```
{
 "figure_id": "",
 "title": "",
 "type": "",
 "content": {
  "states": [],
  "transitions": [
   {
    "from": "",
    "to": "",
    "conditions": []
   }
  ],
  "entities": [],
  "messages": [
   {
    "from": "",
    "to": "",
    "name": "",
    "notes": ""
   }
  ],
  "fields": [
   {
    "name": "",
    "bit_range": "",
    "octet_range": "",
    "description": ""
   }
  ],
  "notes": []
 }
}
```

Return ONLY valid JSON.

Figure 4: Figure content extraction

Table Content Extraction

You are an expert in 3GPP specification table extraction.

Your tasks:
1. Extract the table TITLE, including table number and name.
2. Extract the FULL table content:
   - Column headers
   - All data rows
   - All NOTE rows appearing below or inside the table

Rules for NOTE handling:
- Every NOTE must be preserved.
- Represent NOTE rows as:
  ["NOTE X", "<full note text>"]
- Even if NOTE does not match column count, keep it as two columns.
- Maintain the original order of rows and notes.

Output strictly in the following JSON format:

```
{
 "table_title": "",
 "columns": [],
 "rows": [
  [],
  ...
 ]
}
```

Do NOT hallucinate.
If text is unreadable, return "".

Return ONLY valid JSON.

Figure 5: Table Content Extraction

C *CellSpecSec-ARI* Implementation Details

Figure 6 illustrates the *SpecFusion* preprocessing pipeline, which normalizes raw 3GPP specification text and deterministically segments it into sentence-level and paragraph-level chunks. Figure 7 presents the overall *SpecFusion* framework. Figure 8 shows the construction of clause-aware RAG chunks enriched with explicit clause identifiers and titles, enabling precise evidence attribution. Figure 9 depicts the *SpecRAG* module, which performs hybrid retrieval over these chunks to support evidence-grounded question answering. Figure 10 illustrates the process of constructing a taxonomy-based knowledge graph. Finally, Figure 11 summarizes the complete *SpecReasoning* workflow.

SpecFusion Preprocessing

TASK:
Convert a preprocessed 3GPP specification text (applicable to 4G NAS, 5G NAS, and 5G RRC) into paragraph-level and sentence-level JSONL chunks for deterministic downstream use.

INPUT:
A UTF-8 encoded plain-text file containing processed 3GPP specification content. The text may include clause headers, bullet lists, and multi-line paragraphs.

PROCESSING RULES:
1. Text normalization
- Remove all carriage return characters (\r)
- Collapse consecutive spaces or tabs into a single space
- Replace three or more consecutive newlines with exactly two newlines

2. Clause title and separator removal
Remove any line that matches either of the following:
- Clause or subclause titles beginning with numeric identifiers such as: X, X.Y, X.Y.Z, or X.Y.ZA followed by text (e.g., "4.4 Security handling", "5.2.1A NAS procedures")
- Lines composed solely of repeated separator characters (e.g., "-", "=", "_")

3. Paragraph segmentation
- Split the remaining text into paragraphs using blank lines
- Trim whitespace from each paragraph
- Discard empty paragraphs

4. Sentence segmentation (structure-aware)
For each paragraph:
- Process text line by line
- If a line starts with "- ", treat the remainder as a standalone sentence
- Otherwise:
  - Normalize excessive spaces
  - Split sentences only at ".", "!", or "?" when followed by an uppercase letter or an opening parenthesis "("
  - Preserve sentence-ending punctuation
- Remove leading and trailing hyphens and extra spaces
- Remove immediately repeated sentences within the same paragraph

5. Global sentence deduplication
- Deduplicate sentences across the entire document using case-insensitive comparison
- Preserve original sentence text for output

OUTPUT:
Produce two JSON Lines (.jsonl) files:
1. Paragraph chunks:
   { "text": "<paragraph text>" }
2. Sentence chunks:
   { "text": "<unique sentence text>" }

CONSTRAINTS:
- Do not rewrite, paraphrase, or interpret specification text
- Preserve original wording exactly
- Do not add metadata beyond the "text" field
- Output must be deterministic and reproducible

REPORTING:
- Output the total number of paragraph chunks
- Output the total number of unique sentence chunks

Figure 6: Specification Text Normalization and Chunking Pipeline (*SpecFusion* Preprocessing)

SpecFusion

SpecFusion
Applicable to: 4G NAS, 5G NAS, 5G RRC
Tasks: AQA / EQA / MCQA

GOAL:
- Always enable base reasoning
- Use specification text only as optional knowledge augmentation
- Ensure retrieved text never degrades base performance

STEP 1. Core Design Principle
- The model acts as a domain expert for the target 3GPP specification
Internal knowledge is the primary reasoning source
- Retrieved text is treated as raw, potentially noisy excerpts

STEP 2. Task-Specific Prompting
- EQA (Extractive QA):
  - Quote exact statements from context if available
  - Otherwise answer concisely using base knowledge
- AQA (Analytical QA):
  - Reason about conditions, state transitions, and expected behavior
  - Use text only to anchor constraints or exceptions
- MCQA:
  - Do not provide any text context
  - Require the model to answer using base reasoning only
  - Output only the selected option letter

STEP 3. Answer Generation Constraints
- The model may choose to ignore retrieved text
- The model must not invent rules from vague or descriptive text
- Procedural claims must be supported by either base knowledge or concrete text

STEP 4. LLM-as-Judge Evaluation
- Each answer is evaluated using an independent grading prompt
- Scores:
  - 2 = correct
  - 1 = partially correct
  - 0 = incorrect or contradicts the specification
- Grading is repeated multiple times and decided by majority vote

STEP 5. Output and Metrics
- Save model answers, text usage flags, scores, and justifications
- Report average scores for AQA / EQA and accuracy for MCQA

Figure 7: *SpecFusion*

SpecRAG Preprocessing

TASK:
Convert a processed 3GPP specification text into clause-aware paragraph-level and sentence-level chunks for retrieval-augmented generation (RAG).

INPUT:
A UTF-8 plain-text file of a processed 3GPP specification.
The text contains numbered clauses, clause titles, separators, bullet items, and multi-line procedural descriptions.

PROCESSING RULES:
1. Text Normalization
- Remove carriage return characters (\r)
- Collapse multiple spaces or tabs into a single space
- Replace three or more consecutive newlines with exactly two newlines
- Trim leading and trailing whitespace
2. Clause Extraction
- Identify clause headers using the following pattern:
  - Clause number: X, X.Y, X.Y.Z, or X.Y.ZA
  - Followed by a clause title on the same line
  - Followed by a separator line consisting of repeated '-' or '_'
- For each detected clause:
  - Extract:
    - clause_id (the numeric identifier)
    - clause_title
    - clause_body (all text until the next clause header)
3. Line Filtering within Clauses
For each clause body:
- Split into non-empty lines
- Remove lines that start with numeric clause identifiers
- If a line starts with "- ", remove the bullet marker
- Preserve the original wording of the remaining text
4. Sentence-level Chunk Generation
- Treat each cleaned line within a clause as an individual sentence chunk
- For each sentence chunk, output a JSON object with:
  - clause_id
  - clause_title
  - text (the sentence content)
5. Paragraph-level Chunk Generation
- Merge consecutive lines within a clause body into paragraphs:
  - If the previous line does NOT end with punctuation (., ;, :), concatenate the next line with a space
  - Otherwise, start a new paragraph
- Exclude clause title lines and bullet markers during merging
- For each merged paragraph, output a JSON object with:
  - clause_id
  - clause_title
  - text (the paragraph content)

OUTPUT:
Produce two JSON Lines (.jsonl) files:
1. Sentence-level RAG chunks:
  { "clause_id": "...", "clause_title": "...", "text": "<sentence>" }
2. Paragraph-level RAG chunks:
  { "clause_id": "...", "clause_title": "...", "text": "<paragraph>" }

CONSTRAINTS:
- Do not rewrite, paraphrase, or interpret the specification text
- Preserve original wording exactly
- Do not add metadata beyond clause_id, clause_title, and text
- Chunking must be deterministic and reproducible

REPORTING:
- Output the total number of paragraph chunks
- Output the total number of unique sentence chunks

Figure 8: *SpecRAG* RAG Chunk Construction from 3GPP Specifications

SpecRAG

SpecRAG
Applicable to: 4G NAS, 5G NAS, 5G RRC
Tasks: AQA / EQA / MCQA

Goal:
Evidence-grounded retrieval-augmented reasoning over large-scale 3GPP specifications,
where each answer is explicitly endorsed by verifiable specification evidence.

STEP 1. Hybrid Retrieval:
For each query q and candidate chunk c, compute
s(q,c) = α · s_sp(q,c) + (1−α) · s_de(q,c),
where s_sp is a BM25-based sparse score capturing exact lexical matches, and s_de is a
dense score computed via TF–IDF embeddings with truncated SVD and cosine similarity.
The top-k highest-scoring chunks form the retrieved evidence set E(q).

STEP 2. Context Construction:
Retrieved chunks are formatted as explicit evidence with clause identifiers and titles, preserving original specification wording without paraphrasing.

STEP 3. Task-Specific Prompting:
- AQA: Reason about procedures and conditions using retrieved evidence without contradiction.
- EQA: Prefer verbatim extraction from evidence whenever possible.
- MCQA: Select the correct option with evidence-grounded justification.

STEP 4. Evidence Attribution and Verification:
Answers are attributed to the highest-ranked retrieved clause by default and evaluated
using an independent LLM-as-judge with a three-level scoring rubric.

Metrics:
Report MCQA accuracy, average AQA/EQA scores, and evidence accuracy.

Figure 9: *SpecRAG*

Taxonomy-based Knowledge Graph Construction

TASK:
Given a single clause JSON extracted from a 3GPP specification (4G NAS, 5G NAS, or 5G RRC),
construct a compact clause-level SpecReasoning knowledge graph summarizing key entities
and their relations.

INPUT:
One clause JSON object with fields such as:
- spec, clause_id, title
- text (list of lines)
- figures, tables (optional)

OUTPUT:
Return exactly ONE valid JSON object (no extra text) with:
- spec
- clause_id
- title
- entities: list of entity objects
- relations: list of relation objects

ENTITIES:
Extract only important entities explicitly supported by the clause text, figures, or tables.
Each entity MUST include:
- id (short, stable)
- name
- entity_type ∈ {Procedure, Message, Information Element, Identifier,
  Timer, State, Condition, Property}
- description (1–3 sentences)
- provenance (clause / figure / table references)

RELATIONS:
Connect entities using typed relations. Each relation MUST include:
- id
- source (entity id)
- target (entity id)
- relation_type ∈
  Core: {has_step, contains_IE, requires_state, starts, stops,
  resets, guards_by, establishes, defined_in}
  Security: {requires_protection, integrity_required,
  may_be_unprotected, action_triggered_by, state_transition}
- family ∈ {core, sec}
- description
- provenance

GUIDELINES:
- Do NOT invent entities, states, timers, or messages.
- Use contains_IE for Message–IE structure.
- Use guards_by for conditional presence or behavior.
- Use requires_state for protocol state preconditions.
- Use starts/stops/resets for timers.
- Use security relations ONLY when explicitly stated.
- Keep the graph small but meaningful (≈3–15 entities, 3–20 relations).

Figure 10: *SpecReasoning* Taxonomy-based Knowledge Graph Construction

SpecReasoning

SpecReasoning
Applicable to:
4G NAS (TS 24.301), 5G NAS (TS 24.501), 5G RRC (TS 38.331)

Tasks:
CCQA · TFQA · Vulnerability/inconsistency Labeling · Causal security reasoning with complete evidence

Goal:
Perform evidence-grounded reasoning over clause-level 3GPP specifications, where every output is explicitly supported by retrieved clause evidence.

Input:
A query q (question, vulnerability description, or classification input).

Step 1. Hybrid Retrieval
For each query q and clause c:
- Compute sparse score s_sp(q,c) via BM25 (lexical match).
- Compute dense score s_de(q,c) via TF–IDF + truncated SVD + cosine similarity.
- Combine scores: s(q,c) = α·s_sp(q,c) + (1−α)·s_de(q,c).
- Select top-k clauses as evidence set E(q).

Step 2. Context Construction
Format evidence as:
[i] Clause <clause_id> — <clause_title>
<original clause text or KG-derived description>
(Identifiers and wording are preserved.)

Step 3. Task-Specific Reasoning
- CCQA / TFQA: Answer concisely using evidence.
- Vulnerability/inconsistency Labeling: Output JSON with binary label/inconsistency label and security labels.
- Causal security reasoning with complete evidence: Explain vulnerability and cite supporting clauses.

Step 4. Evidence Attribution
Cite evidence as:
“TS <spec>, Clause <clause_id>, <clause_title>”.

Step 5. Verification
Grade outputs using an LLM-as-judge (2 = correct, 1 = partial, 0 = incorrect).

Outputs & Metrics:
Store answers, retrieved evidence, scores, and evidence correctness.
Report average score, classification accuracy, and evidence accuracy.

Figure 11: *SpecReasoning*

D The example of taxonomy-based KG

The taxonomy-based KG defines a finite set of entity and relation types that capture the recurring structural, procedural, and security-relevant concepts in 3GPP specifications. Tables 2 and 3 summarize the entity types and relation types used in *SpecReasoning*, respectively.

We illustrate how this taxonomy is instantiated in practice through a concrete example drawn from TS 24.501. Specifically, we show how *SpecReasoning* transforms a UE-initiated de-registration procedure into a set of typed entities and relations, grounding each extracted fact in its normative source. For example, from TS 24.501, Clause 5.5.2.2 (UE-initiated De-registration procedure), *SpecGraph* analyzes the procedural description and the referenced message definitions. The specification states that the procedure is initiated by the UE sending a DEREGISTRATION REQUEST message, that the De-registration Type information element indicates whether the de-registration is due to a switch-off, and that the Access Type specifies whether it applies to 3GPP or non-3GPP access. Based on this passage, *SpecGraph* extracts a procedure node (UE-initiated De-registration), a message node (DEREGISTRATION REQUEST), information-element nodes (De-registration Type and Access Type), the timer node (T3521), and the state node (5GMM-DEREGISTERED). It then instantiates both structural and security relations: the procedure includes a signaling step involving the transmission of the DEREGISTRATION REQUEST message (has_step), the message contains the De-registration Type IE (contains_IE), execution of the procedure starts timer T3521 (starts) and requires the UE to be in the 5GMM-DEREGISTERED state (requires_state), and the DEREGISTRATION REQUEST message may be sent without integrity protection (may_be_unprotected). Furthermore, *SpecGraph* records that the AMF's action of sending a DEREGISTRATION ACCEPT message is triggered by the reception of the unprotected request (action_triggered_by), causing a transition of the UE's mobility management state from REGISTERED to DEREGISTERED-INITIATED (state_transition). Each of these relations is anchored to its normative source through the defined_in relation and the provenance function λ referencing TS 24.501, Clause 5.5.2.2.

Entity Type	Description
Procedure	Representing complete signaling workflows (e.g., Registration, Security Mode Control).
Message	Denoting individual NAS or RRC signaling messages (e.g., REGISTRATION REQUEST, DEREGISTRATION ACCEPT).
Information Element (IE)	Representing message-level fields.
Identifier	Covering identifiers such as SUCI or GUTI.
Timer	Corresponding to protocol timers (e.g., T3510, T3521).
State	Denoting UE or network states (e.g., 5GMM-DEREGISTERED).
Condition	Expressing logical or optional constraints (e.g., Presence: O).
Property	Capturing higher-level system or security attributes (e.g., NAS Integrity Protection).

Table 2: Entity types defined in *SpecReasoning*

Relation Type	Description
Core relations $\mathcal{R}_{core}$	
has_step	Connects a procedure to its constituent signaling steps.
contains_IE	Associates a message with its contained information elements.
requires_state	Execution preconditions (e.g., UE must be in state s).
starts / stops / resets	Timer operations induced by a step/message.
guards_by	Conditional constraints (e.g., Presence/If-Then).
establishes	Creation/activation of contexts (e.g., NAS security context).
defined_in	Links each fact to its source clause/table/figure reference.
Security relations $\mathcal{R}_{sec}$	
requires_protection	An object (message/IE/step) shall be integrity/cipher protected.
integrity_required	Integrity protection is mandated for subsequent handling.
may_be_unprotected	Object may appear before security is established (exceptions).
action_triggered_by	A network/UE action is triggered upon receiving a (possibly unauthenticated) message.
state_transition	A message/event causes a state transition (e.g., REGISTERED $\rightarrow$ DEREGISTERED).

Table 3: Relation types defined in *SpecReasoning*

E Impact of retrieval context size k

To examine the sensitivity of *CellSpecSec-ARI* to retrieval context size. Table 4 summarizes the performance of Stage 2 and Stage 3 tasks under two settings, $k = 8$ and $k = 6$. All results are reported on the first 50 instances of each task, using the same task definitions and evaluation metrics as in the main experiments.

Stage	**Task**	$k = 8$	$k = 6$
2	AQA-E	S2: 96.8%; E: 96.4%	S2: 88%; E: 72%
2	EQA-E	S2: 96%; E: 98%	S2: 86%; E: 86%
2	MCQA-E	Acc: 100%; E: 100%	Acc: 88%; E: 86%
3	CCQA	S2: 95.06%; E: 91%	S2: 74%; E: 88%
3	TFQA	S2: 92%; E: 95%	S2: 88%; E: 92%
3	Vulnerability / Inconsistency Labeling	Bin/Multi: 96% / 90%	Bin/Multi: 88% / 86%
3	Evidence and Explanations	E: 84%	E: 80%

Table 4: Impact of retrieval context size (k) on stage 2 and stage 3 tasks

F External datasets

Table 5 provides a consolidated summary of the external benchmarks incorporated at each stage, together with their task formats and the corresponding capabilities they assess.

Stage	**Benchmark(s)**	**Task**	**Capability Evaluated**
1	TeleQnA; Tele-LLMs (Tele-Eval)	Multiple-choice reasoning, Open-ended answering	Specifications comprehension; terminology and normative language understanding
2	TSpec-LLM	Multiple-choice reasoning	Specifications reasoning
3	5GSC; ConTester; CellularLint	Sentence-level classification, Sentence-pair semantic/NLI	Fine-grained semantics (security vs. non-security); semantic relations

Table 5: Overview of external datasets included in *CellularSpecSec-Bench*

G Examples for each task type

Here we provide examples for each task type included in *CellularSpecSec-Bench*, open-ended answering (Figure 12), multiple-choice answering (Figure 13), and vulnerability/inconsistency labeling with evidence and explanations (Figures 14, 15).

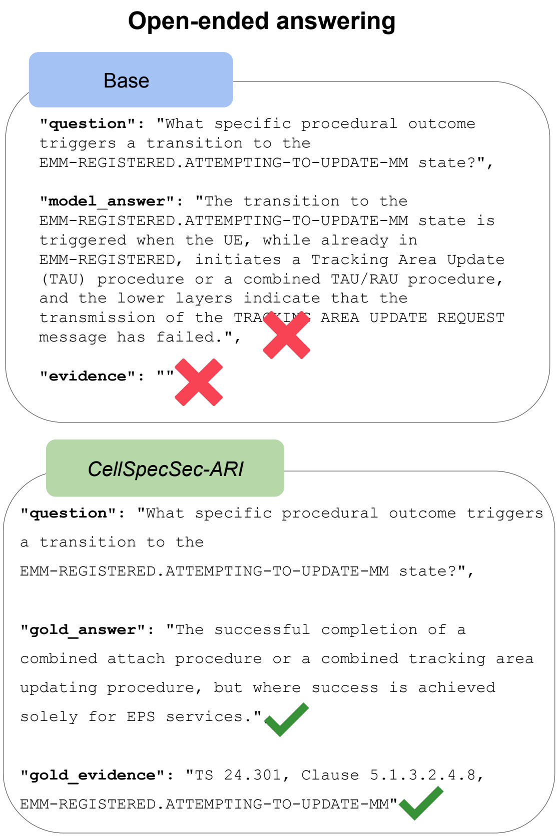

Figure 12: Open-ended answering

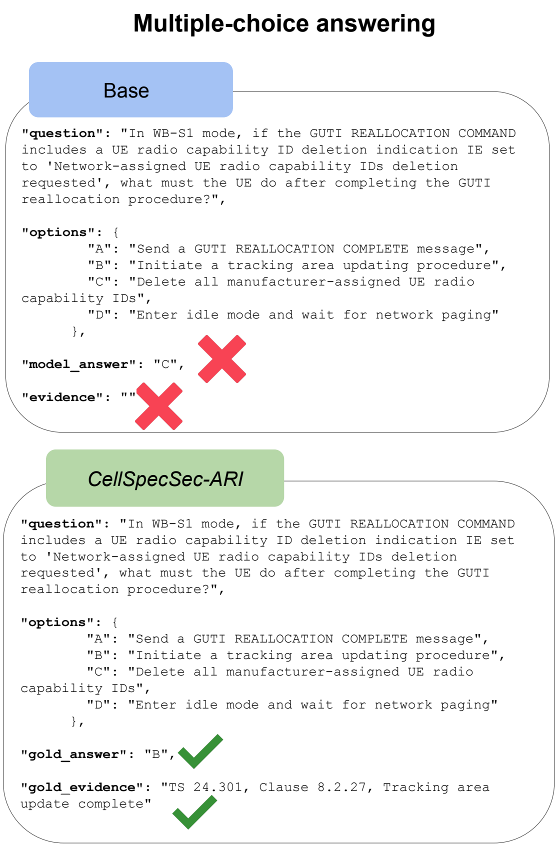

Figure 13: Multiple-choice answering

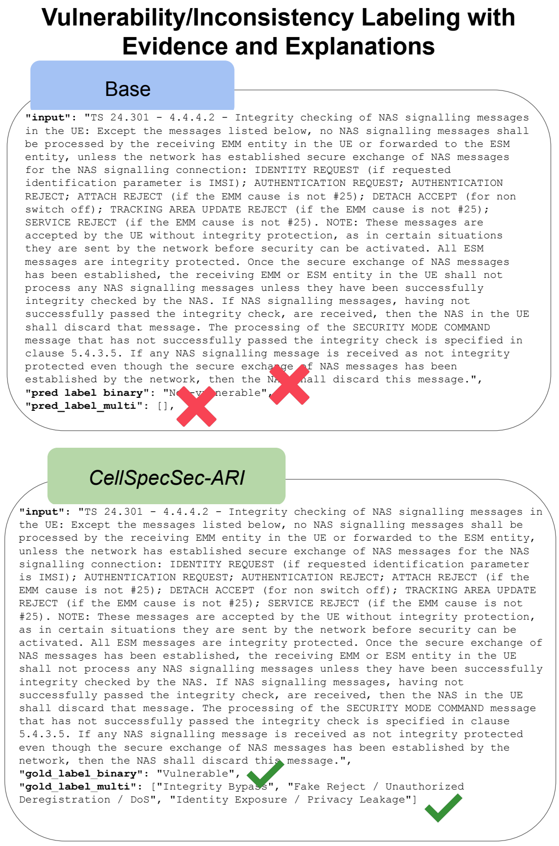

Figure 14: Vulnerability/Inconsistency Labeling with Evidence and Explanations (Example I)

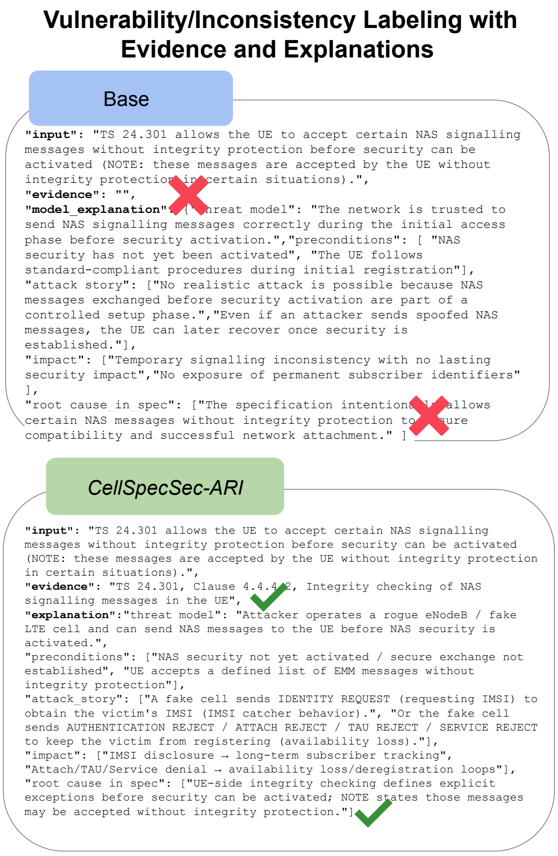

Figure 15: Vulnerability/Inconsistency Labeling with Evidence and Explanations (Example II)

H Details of Tasks in *CellularSpecSec-Bench*

H.1 Stage 1: Intra-Clause Specification Comprehension

Tasks in this stage are constructed so that the correct answer can be found within a single specification clause (including all multimodal contexts), without requiring reasoning across multiple clauses, tables, or figures. Stage 1 tasks focus on evaluating the simplest adaptation ability, which is fundamental to interpret 3GPP specifications. The following tasks are included in Stage 1:

Original Tasks. There are three types of tasks developed originally. **Extractive QA (EQA)**: It's an open-ended answering task. Models must identify an explicit answer that appears verbatim in a single clause from specifications, measuring literal comprehension of standards; **Abstractive QA (AQA)**: It's an open-ended answering task. Models must generate a concise, semantically coherent answer that synthesizes information from nearby normative text, measuring high-level understanding beyond span matching; **Multi-choice QA (MCQA)**: It's a multiple-choice reasoning task. Models need to select the answer that reflects the specification-compliant behavior under a given condition, state, or signaling action.

Verified and Corrected Tasks. We apply a quality screening and sample multiple-choice questions from TeleQnA (Maatouk et al., 2023) and QA questions from Tele-Eval (Maatouk et al., 2025) First, we incorporate multiple-choice questions from TeleQnA (Maatouk et al., 2023), which aggregates questions from heterogeneous sources (e.g., research publications/overviews, lexicons, standards overviews, and standards specifications). Since *CellularSpecSec-Bench* focuses on specification-grounded understanding, we restrict TeleQnA to the standards-related subsets, standards overview and standards specifications. We then apply a quality screening to remove ill-posed items and finally sample 500 MCQAs from the remaining pool to form our Stage 1 external MCQA subset.

Second, we construct a 500-question general QA subset from Tele-Eval (Maatouk et al., 2025). Tele-Eval aggregates questions derived from heterogeneous sources, including (i) scientific papers from arXiv, (ii) 3GPP standards, (iii) Wikipedia articles related to telecommunications, and (iv) telecommunications-related websites extracted from Common Crawl dumps. Since *CellularSpecSec-Bench* targets specification-grounded understanding, we restrict our use of Tele-Eval to questions originating from the *3GPP standards* subset. We then manually inspect the candidate questions to ensure that they are specification-relevant and fall within the scope of *CellularSpecSec-Bench*. After this filtering process, we verify and correct 500 high-quality open-ended answering questions that are fully aligned with our task construction criteria.

These Stage 1 verified and corrected tasks' results are summarized in the Appendix J Table 7.

H.2 Stage 2: Evidence-Grounded Answering

Original Tasks. Stage 2 introduces an explicit evidence grounding requirement to enforce verifiability of model outputs. In this stage, each prediction must be both correct and traceable to the authoritative specification source: the model outputs the answer together with the supporting clause identifier or table/figure identifiers. Evidence grounding prevents answers that are superficially plausible but unverifiable, and therefore measures retrieval-based reasoning in addition to factual correctness. Similar to Stage 1, Stage 2 maintains full coverage across TS 24.501 (3GP, 2022a), TS 38.331 (3GP, 2022b), and TS 24.301 (3GP, 2020). For each specification, Stage 2 includes 500 EQA with Evidence (EQA-E), 500 AQA with Evidence (AQA-E), and 500 MCQA with Evidence (MCQA-E), aligned with the task definitions in Section 5.1. In all cases, *CellSpecSec-ARI* must produce a correct answer and an explicit evidence reference, ensuring each prediction is grounded in 3GPP specifications.

Verified and Corrected Tasks. The external datasets incorporated in Stage 2 (see Appendix 6). We include the 3GPP specification portion of TSpec-LLM (Nikbakht et al., 2024) to assess robustness under increased question difficulty. Because TSpec-LLM does not publicly release its evaluation set, we reproduce its question-generation pipeline using the same GPT-4 API and generate 500 MCQAs grounded in our target specification scope. We also considered integrating Telco-DPR (Saraiva et al., 2024); however, manual inspection revealed substantial quality issues, including QA pairs whose answers cannot be located in the referenced 3GPP specifications. We therefore exclude Telco-DPR from the main benchmark to preserve evidence verifiability. These Stage 2 verified and corrected tasks' results are summarized in the Appendix J Table 7

H.3 Stage 3: Cross-Clause and Security-Aware Reasoning

Stage 3 consolidates the remaining highest-level reasoning requirements in *CellularSpecSec-Bench* by requiring models to integrate information across multiple specification clauses and perform security-aware interpretation.

Original Tasks. Stage 3 first includes open-ended answering tasks whose correct solutions require jointly interpreting information across multiple specification clauses and/or combining textual and tabular/figure content. It includes two task families: Cross-Clause Question Answering (CCQA) and Table-and-Figure Question Answering (TFQA). CCQAs are constructed only when the correct answer necessarily depends on the combined interpretation of multiple explicitly related clauses. We include both single-unit TFQAs, where the question can be answered using a single table or figure, and cross TFQAs, which require jointly interpreting multiple tables or multiple figures. Under these constraints, Stage 3 contains 222 CCQAs and 747 TFQAs. For all questions, models must generate an answer together with an evidence set consisting of clause numbers, when applicable, table identifiers and/or figure labels.

Stage 3 further introduces two security-analysis tasks: vulnerability/inconsistency labeling, and the evidence and explanations task. We first develop a vulnerability set, consolidating from prior works and manually verified (Shaik et al., 2015; Bassil et al., 2013; Kambourakis et al., 2011; Lee et al., 2009; Leong et al., 2014; Kim et al., 2019; Van Den Broek et al., 2015; Park et al., 2016; Chlosta et al., 2019; Yu and Chen, 2019; Michau and Devine, 2016; Hussain et al., 2019b; Borgaonkar et al., 2018; Cao et al., 2020; Chlosta et al., 2021; Hussain et al., 2019a; Al Ishtiaq et al., 2024; Hussain et al., 2018; Xie et al., 2025); non-vulnerable controls are sampled from ordinary normative text.

For the vulnerability/inconsistency labeling task, given a normative sentence, *CellSpecSec-ARI* determines whether it encodes a security-relevant vulnerability or a specification inconsistency and, if so, assigns one or more predefined vulnerability or inconsistency categories. Based on the vulnerability set, we include 27 vulnerability sentences in total (TS 24.501: 11; TS 38.331: 7; TS 24.301: 9) and provide 50 instances per specification by combining vulnerability cases with non-vulnerable controls. Each output consists of a binary vulnerability label and a multi-label vector over predefined vulnerability categories (e.g., DoS, replay, downgrade, privacy/tracking, spoofing, authentication bypass, etc.).

The evidence and explanation task, representing the highest reasoning level in *CellularSpecSec-Bench*, evaluates whether *CellSpecSec-ARI* can explain why a vulnerability arises and retrieve the complete set of supporting evidence across multiple clauses/tables/figures. Leveraging the vulnerability set, this task covers 31 vulnerability scenarios in total (TS 24.501: 15; TS 38.331: 9; TS 24.301: 7), and we provide 50 instances per specification by combining vulnerability cases with non-vulnerable controls. For a vulnerable case, *CellSpecSec-ARI* outputs (1) a concise explanatory summary and (2) an evidence set consisting of one or more normative clauses (and, when applicable, table identifiers or figure labels); for a non-vulnerable control, the correct output indicates that no vulnerability is present and the evidence set is empty.

Verified and Corrected Tasks. To complement our vulnerability and inconsistency labeling task, Stage 3 incorporates three datasets, each curated through our verification and correction. Results on these verified and corrected Stage 3 tasks are reported in Appendix L, Table 8.

First, 5GSC dataset (Karim et al., 2023) contains 2,401 sentences annotated into three classes: Non-Security, Security, and Undefined. Through careful manual inspection, we identify a substantial number of samples whose security labels cannot be inferred from the sentence content itself. For example, the sentence “IP-CAN Session Termination listed in this Annex” is labeled as Security in the original dataset, despite merely referring to an annex title without describing any security mechanism, threat, or protection property. To ensure label consistency and semantic clarity, we manually review all samples and remove sentences with insufficient contextual information, including section headers and other low-information fragments. After filtering, we retain 1,454 high-quality sentences.

ConTester (Chen et al., 2023) constructs a dataset by randomly sampling 500 sentences from the 4G NAS specification and forming 400 sentence pairs for semantic classification, reporting an overall accuracy of 97.25%. However, inspection of the publicly released dataset reveals that it contains only 366 sentence pairs, and that some annotations are inconsistent with standard semantic interpretations. For example, Sample 333 consists of the sentences “the attach attempt counter is equal to 5” and “the UE should set the attach attempt counter to 5”, which are labeled as class 1 (semantically equivalent). In fact, the former describes a system state, whereas the latter specifies an action, indicating a clear semantic distinction. To ensure label consistency and annotation quality, we manually reviewed the dataset and retained 320 sentence pairs.

Last, CellularLint (Rahman et al., 2024) provides a sentence-pair dataset derived from 3GPP specifications for Natural Language Inference (NLI). However, its annotation criteria deviate from widely accepted NLI standards. Under standard NLI definitions, entailment requires that the second sentence logically follows from the first, contradiction indicates an explicit conflict within the same context, and neutral denotes plausibility without entailment or contradiction. We therefore construct a new dataset of 55 sentence pairs sampled from 4G/5G NAS and RRC specifications, with all pairs annotated strictly according to standard NLI.

I Verified and Corrected Datasets included in Stage 1 and Stage 2

Table 6 summarizes the external datasets incorporated in Stage 1 and Stage 2, together with the corresponding filtering decisions and scope alignment considerations applied during task construction.

Dataset	Task	Kept	Issues Identified
TeleQnA	Multiple-choice reasoning (Stage 1)	500	Restricted to standards overview and standards specifications; removed questions derived from non-standard sources (e.g., lexicons, research papers); filtered ill-posed items
Tele-Eval	Open-ended answering (Stage 1)	500	Restricted to questions originating from 3GPP standards; excluded QA pairs derived from arXiv, Wikipedia, and Common Crawl sources; ensured specification-relevant scope
TSpec-LLM	Multiple-choice reasoning (Stage 2)	500	Reproduced question-generation pipeline due to unavailable evaluation set; generated MCQAs grounded in target 3GPP specifications
Telco-DPR	Multiple-choice reasoning (candidate)	0	Excluded due to unverifiable answers and hallucinated evidence not traceable to cited 3GPP clauses

Table 6: Verified and corrected datasets included in Stage 1 and Stage 2

J Results for Verified and Corrected Datasets in Stage 1 and Stage 2

Table 7 summarizes the evaluation results of *CellSpecSec-ARI* on external benchmarks included in Stage 1 and Stage 2. Note that external benchmarks contain incorrect entries as discussed before. Results reported for *CellSpecSec-ARI* are therefore computed on our verified and corrected subsets.

TeleQnA reports baseline accuracies for the standards/specifications category of 56.97% (GPT-3.5), 64.78% (GPT-4), and 69.84% (GPT-3.5 with standards context). On our verified TeleQnA MCQA subset, *CellSpecSec-ARI* achieves 94.20% accuracy.

For the Open-ended answering subset derived from Tele-Eval, we evaluate answers using a three-level rubric (2: fully correct; 1: partially correct; 0: incorrect), under which *CellSpecSec-ARI* attains 496/500 fully correct, 1/500 partially correct, and 3/500 incorrect. For comparability with Tele-LLMs (Tele-Eval), which adopts an LLM-as-a-judge binary protocol (Yes/No), we map score=2 to Yes and score=0,1 to No; under this aligned protocol, *CellSpecSec-ARI* achieves a corresponding +59.4% absolute improvement over the base model. Tele-LLMs reports an average improvement of ∼25% over base.

Reported results for TSpec-LLM show accuracies of 93% on easy questions, 66% on medium questions, and 65% on hard questions. Under our reproduced evaluation setting, *CellSpecSec-ARI* achieves 94% overall accuracy.

Stage	Benchmark	Task	Reported Baselines	*CellSpecSec-ARI*
1	TeleQnA	Multiple-choice reasoning	Std. specs: GPT-3.5 56.97%, GPT-4 64.78%, GPT-3.5+ctx 69.84%	94.20%
1	Tele-LLMs (Tele-Eval)	Open-ended answering	LLM-as-judge (Yes/No); reports +25% over base	+59.4% over base
2	TSpec-LLM	Multiple-choice reasoning	Easy 93%, Medium 66%, Hard 65%	Average 94.0%

Table 7: Results for verified and corrected datasets in Stage 1 and Stage 2

K Verified and Corrected Datasets included in Stage 3

Table 8 summarizes the external datasets included in Stage 3, together with the corresponding filtering decisions and data quality issues identified during curation.

Dataset	Task	Orig.	Kept	Issues Identified
5GSC	Sentence-level classification	2,401	1,454	Insufficient context (e.g., headers or annex references); label not inferable from sentence alone
ConTester	Sentence-pair semantic classification	366	320	Annotation inconsistency; state vs. action conflation (e.g., Sample 333)
CellularLint	NLI	55	55	Non-standard NLI criteria; re-annotated under standard NLI definitions

Table 8: Verified and corrected datasets included in Stage 3

L Results for verified and corrected tasks in Stage 3

On the verified and corrected 5GSC subset, *CellSpecSec-ARI* correctly classifies all instances across the three classes, achieving 100% accuracy. On the refined ConTester subset, *CellSpecSec-ARI* also achieves 100% accuracy. On the newly constructed CellularLint dataset aligned with standard definitions, *CellSpecSec-ARI* attains 100% accuracy.

M Evidence correctness

Stage 2 introduces explicit evidence grounding, and we report evidence correctness in addition to answer correctness. Evidence correctness is measured by the exact match between the predicted evidence set (clause numbers and/or table/figure identifiers) and the gold references. For tasks that require multiple evidence units, we consider the prediction correct only if it includes the complete gold set.

N LLM-as-judge

Figure 16 show details of the prompt template and scoring procedure used by the LLM-based judge for evaluating open-ended question answering tasks.

LLM-as-Judge

Inputs:
- Question Q
- Model-generated answer A
- Gold reference answer R

Judge Model:
- DeepSeek-V3.2 (vanilla)
- Temperature = 0

Grading Prompt:
Grade this answer.
Score 2 = Correct (covers essential points, no contradiction).
Score 1 = Partially correct (right direction but missing key detail).
Score 0 = Incorrect or contradicts spec.
Question:
{Q}
Gold answer:
{R}
Model answer:
{A}
Respond ONLY as JSON with keys "score" and "justification".

Notes:
- The judge does NOT have any external knowledge

Figure 16: *LLM-as-judge*

O Results for Stages 1–3 tasks

Tables 9, 10, and 11 present the evaluation results of the three *CellSpecSec-ARI* components (*SpecFusion*, *SpecRAG*, and *SpecReasoning*) across the three stages of *CellularSpecSec-Bench*.

Task	Metric	BASE	SpecFusion	*CellSpecSec-ARI*
Stage 1: Intra-Clause Specification Comprehension				
EQA	Score=2	36%	85.8%	97.75%
AQA	Score=2	34%	81.40%	97%
MCQA	Accuracy	76%	83%	100%
Stage 1: Verified & Corrected External Tasks				
TeleQnA	Accuracy	90%	94.20%	96%
Tele-Eval	Score=2	38%	97.4%	97.4%

Table 9: Results of Stage 1 tasks

Task	Metric	BASE	SpecRAG	*CellSpecSec-ARI*
Stage 2: Evidence-Grounded Answering				
EQA-E	Score=2 & Evidence Correct	0%	96%	97%
AQA-E	Score=2 & Evidence Correct	0%	96.40%	96.50%
MCQA-E	Accuracy & Evidence Correct	0%	100%	100%
Stage 2: Verified & Corrected External Tasks				
TSpec-LLM	Accuracy & Evidence Correct	76%	94%	94%

Table 10: Results of Stage 2 tasks

Task	Metric	BASE	*CellSpecSec-ARI*
Stage 3: Vulnerability/Inconsistency Labeling with Evidence and Explanations			
CCQA	Score=2 & Evidence Correct	0%	98%
TFQA	Score=2 & Evidence Correct	0%	96%
Vulnerability / Inconsistency Labeling	Binary F1 Multi Micro-F1 Multi Macro-F1	80% 39.02% 38.81%	93.05% 70.59% 89.33%
Evidence and Explanations	Evidence & Explanations Correctness	0%	88%
Stage 3: Verified & Corrected External Tasks			
5GSC	Accuracy	46%	100%
ConTester	Accuracy	96%	100%
CellularLint	Accuracy	36%	100%

Table 11: Results of Stage 3 tasks