Can Large Language Models Automate the Refinement of Cellular Network Specifications?

Anonymous ACL submission

Abstract

Cellular networks, *e.g.*, 4G/5G, rely on complex technical specifications to ensure correct functionality; however, these specifications often contain flaws or ambiguities. In this paper, we investigate the application of Large Language Models for *automated cellular network specification refinement*. We identify Change Requests, which record specification revisions, as a key source of domain-specific data and formulate refinement as three complementary sub-tasks. We introduce **CR-EVAL**, a benchmark of 200 security-related test cases, and evaluate 17 open-source and 14 proprietary models. The best-performing model, `GPT-o3-mini`, identifies weaknesses in over 127 test cases within five trials. We further study LLM specialization, showing that fine-tuning an 8B model can outperform advanced LLMs such as `DeepSeek-R1` and `Qwen3-235B`. Evaluations on 30 real-world cellular attacks demonstrate the practical impact and remaining challenges. The codebase and benchmark are available at `https://anonymous.4open.science/r/CR-Eval`.

1 Introduction

Recent advances in Large Language Models (LLMs) (Achiam et al., 2023; OpenAI, 2025; Guo et al., 2025) have sparked their remarkable applications across diverse domains, including finance (Wu et al., 2023), healthcare (Singhal et al., 2023), and mathematics (DeepMind, 2024). In this work, we investigate ***automated cellular network specification refinement***, a previously unthinkable yet now plausible concept with LLMs.

Cellular networks, *e.g.*, 4G/5G, rely on technical specifications to define protocol behavior and interoperability (GSMA, 2025a; 3GPP, 2025a). Despite their importance, these specifications often contain design flaws or ambiguities that can lead to security vulnerabilities or performance degradation (Shaik et al., 2016). Detecting such weaknesses has traditionally relied on manual expert analysis (Rupprecht et al., 2019), which is increasingly impractical as the standards rapidly expand[1]. Existing automated approaches, including formal verification (Hussain et al., 2019b) and NLP-based methods (Al Ishtiaq et al., 2024), provide partial automation but still require expert effort and do not scale well with evolving specifications. Consequently, the cellular networking community continues to call for more powerful and scalable automation to detect and mitigate specification weaknesses (3GPP, 2020; Nair, 2024). This motivates exploring LLMs for automated cellular specification refinement, given their strong language understanding and capable reasoning abilities.

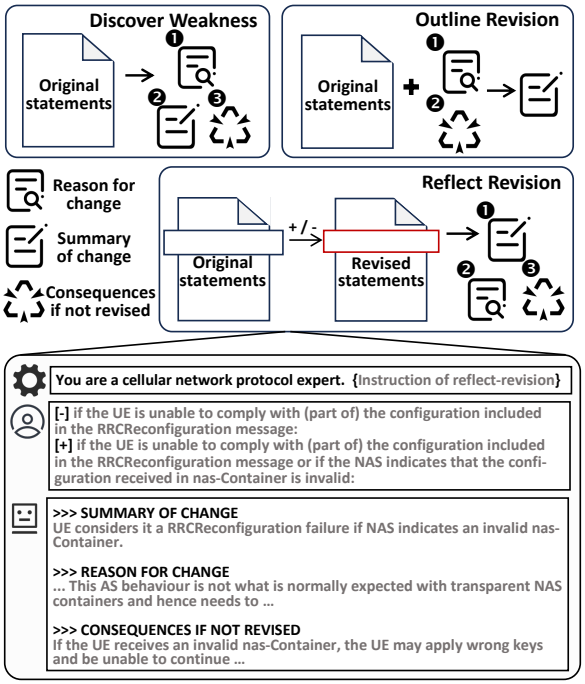

Figure 1: Illustration of task formulation on **CR-EVAL**.

To advance this, we identify and leverage

[1]Measured by PDF page count, 3GPP standards grew from 59,258 pages in Release 8 (LTE) to 117,951 pages in Release 15 (5G), and to 195,752 pages in Release 18 as of March 2025.

200,000+ approved Change Requests (CRs), which document historical specification revisions and necessary expert comments, as a data source. Based on CRs, we devise three complementary LLM-tractable domain tasks: *discover-weakness* (uncovering potential weaknesses in specifications), *outline-revision* (proposing necessary revisions given weaknesses), and *reflect-revision* (ensuring revisions address weaknesses). Under this framing, we propose **CR-EVAL**, which comprises 200 security-related test cases and serves as a proxy for the real-world application of LLMs. We extensively evaluate 17 open-source and 14 proprietary models, including `GPT-5`. In the hardest yet imperative *discover-weakness* task, the best-performing model `GPT-o3-mini` can discover weaknesses in over 127 out of 200 test cases within five trials.

We further explore domain specialization through fine-tuning. We introduce an effective three-stage training recipe combined with a novel rationale augmentation technique. The resulting domain-specialized 8B LLM nearly triples the performance of its base model (`LLaMA-3.1-8B`) on the *reflect-revision* task and even outperforms advanced models such as `Qwen3-235B` on the *discover-weakness* task. Further analysis of token-prediction behavior shows increased emphasis on security-related tokens and more precise technical terminology of cellular networks. We conduct extensive experiments to examine the scalability of the training recipe and its extensibility to stronger base models. In addition, we evaluate the domain-specialized LLM on 30 known cellular attacks, all of which are successfully detected.

As the incoming 6G technology integrates new features (Lin, 2024; 3GPP, 2025b), it inevitably drives the evolution of 3GPP standards and raises concerns about new specifications. We show that LLMs present a timely and effective opportunity for automated cellular specification refinement. Our main contributions are threefold:

- **New insight.** We pioneer LLM adoption for cellular specification refinement and strategically leverage change requests as domain data to form the foundation of a systematic study.
- **Evaluating LLMs' domain-specific ability.** We establish **CR-EVAL**, which enables the community to understand the domain-specific abilities of modern LLMs. Using **CR-EVAL**, we conduct an extensive measurement across 31 representative frontier LLMs.

SpecNumber	CRNum	RevNum	Current version	x.y.z
Title	A descriptive title		Date	Written date
Categorgy	e.g., ``F'' -> Correction		Release	e.g., Rel-18
Reason for change	Filled in free text			
Summary of change	Filled in free text			
Consequences if not approved	Filled in free text			
Original statements	Track changes →	Revised statements		

Figure 2: Structure of 3GPP Change Request coversheet (see Tables 13 and 14 in Appendix G for examples).

- **Towards domain-specialized LLMs.** We explore avenues for domain specialization, including an effective fine-tuning recipe. We test on known cellular attacks to identify areas for further improvements in steering LLMs.

2 Necessary Backgrounds

Largae Language Models (LLMs), for example, `GPT-5` (OpenAI, 2025), utilize the decoder-only Transformer architecture (Vaswani et al., 2017) and are trained on the next-token-prediction task (Radford et al., 2018) as:

$$P(x_1, \cdots, x_N) = \prod_{i=1}^{N} P(x_i | x_1, \cdots, x_{i-1}), \quad (1)$$

where $P(x_i|x_1, \cdots, x_{i-1})$ represents the probability of predicting token x_i given the preceding sequence $x_1, \cdots, x_{i-1}$, with tokens typically operating at the subword level. During inference, LLMs can respond to user queries provided as prompts.

Cellular specifications, standardized by the 3rd Generation Partnership Project (3GPP), define the operation of cellular network systems, ensuring interoperability across vendors (GSMA, 2025a; 3GPP, 2025a). As cellular networks evolve from 2G through 5G and beyond, specifications undergo updates through a structured process involving *technical specification groups (TSGs)* and industry stakeholders. Among these, many updates aim to address inherent security/privacy vulnerabilities discovered in cellular specifications (we refer curious readers to Appendix H for a brief introduction).

Change request. To manage specification updates, 3GPP employs a Change Request (CR) procedure to revise specifications for various purposes, including keeping consistent with a change in an earlier release (*A*), addition of feature (*B*), functional modification of feature (*C*), editorial modification (*D*), and correction (*F*) (3GPP, 2024). 3GPP individual members (*e.g.*, Qualcomm, Apple) raise CRs

using a template coversheet (3GPP, 2025c). As illustrated in Figure 2, each CR has several key blocks, including meta-information, expert rationales that explain the necessity of revisions, and the proposed clause modifications. The modifications are tracked by the word processor software's "revision mode" and surrounded by the proposer-decided specification clauses as context.

3 Benchmarking LLMs for Cellular Specification Refinement

3.1 Insight: Leveraging CRs as Data Source

Large-scale human labeling for the task of refining cellular specification is largely impractical due to the high demand for expertise. To address the challenge of domain data scarcity, we propose utilizing the approved change requests as valuable data sources. Specifically, CRs that correct existing statements are especially suited for our purpose as they inherently reflect specification weaknesses in earlier versions. Categories such as *F* (correction), *D* (editorial modification), and potentially others, encompass various specification weaknesses discussed in this work. Key elements of our focus include *expert rationales* R (*reason for change* R_r and *consequences if not revised* R_c), *summary of change* S'_{rev}, *original statements* S_{orig}, and *revised statements* S_{rev}. See Table 13 for a CR example.

3.2 Formalizing Specification Refinement Into LLM-Tractable Tasks

We devise three domain sub-tasks that mirror the real-world process of refining cellular specifications, as illustrated in Figure 1.

- **Discover Weakness** ($S_{orig} \rightarrow R$)**:** This task positions LLMs as expert reviewers, requiring them to discover potential weaknesses in given statements. This task is relatively challenging as models receive minimal contextual information.
- **Outline Revision** ($S_{orig} + R \rightarrow S'_{rev}$)**:** Once specification weaknesses are identified, the next step is revision. To simplify the task, we require the model to outline a revision plan.
- **Reflect Revision** ($|S_{rev} - S_{orig}| \rightarrow R$)**:** This task supports real-world scenarios where editors assess whether revisions exactly imply and thus address the identified weaknesses.

Our evaluation emphasizes scenarios where LLMs operate in a zero-shot setting, where LLMs receive a general task instruction without any case-dependent inductive information (see Prompt 1 of the *discover-weakness* task). This setting honestly reflects their intrinsic ability to handle the given task instance with minimal human intervention.

3.3 Benchmark Establishment

Each change request can be instantiated across the three domain tasks via predefined task templates. Concretely, each test case comprises a task instruction, a structured task-dependent input, and a reference answer. We provide examples of the *discover-weakness* task in Examples 1 to 3. While change requests are issued for various purposes, we focus primarily on those with security-related consequences and potentially severe implications. Using LLM-based security tagging, we select 200 security-related CRs to form **CR-EVAL**. We defer the dataset curation details to Section 3.5 to enable clearer side-by-side understanding of the training-set and benchmark processing (including decontamination).

We conduct a comprehensive structural analysis of the 200 test cases in Appendix E: The benchmark exhibits extensive release and specification coverage, progressive difficulty levels, and long-context complexity. Qualitatively, the test cases in **CR-EVAL** feature well-structured, focused specification clauses rich in cellular network terminology (*e.g.*, *AUTS*, *VLR/SGS*, and *synchronization failure message*). **CR-EVAL** serves as a holistic assessment of LLM capabilities, encompassing extensive domain knowledge, systematic reasoning, precise instruction following, effective long-context processing, a deep understanding of cellular specification weaknesses, and acute awareness of security-related vulnerabilities.

3.4 Automatic Evaluation

Following prior work (Zheng et al., 2023; Fang et al., 2024; Ullah et al., 2024), we use a reference-aware, point-wise LLM-as-a-Judge setting, where each LLM-generated answer is scored by comparing it to the reference answer. LLM-as-a-Judge evaluates responses using a 5-point Likert scale (Likert, 1932), where the positive two points indicate acceptance. This allows differentiation between varying degrees of acceptance. The detailed LLM-as-a-Judge prompt template is shown in Prompt 5, with minor task-specific variations. We instantiate the LLM-as-a-Judge with `GPT-4o` (OpenAI, 2024). For reproducibility, we prompt LLM-as-a-Judge to directly give back the scoring and greedily decode with temperature as 0.

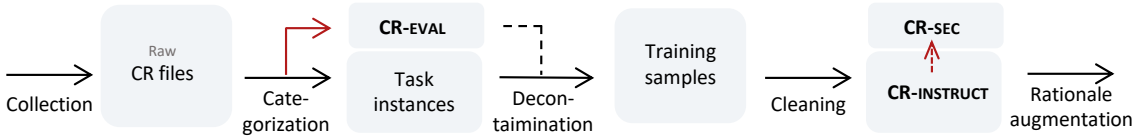

Figure 3: Overview of the data-processing pipeline with change requests. This sequentially includes categorization of security relevance, decontamination, cleaning, and rationale augmentation. See Appendix C.1 for details.

We validate the reliability of our final LLM-as-a-Judge setup through a human study, involving eight PhD students majoring in network security. The study includes two rounds: an alignment test and a judgment approval test. Detailed settings, labeling system snapshots, and results are provided in Appendix F. Key takeaways include:

- **Availability:** Manual checking is extremely labor-intensive, underscoring the need for efficient automatic methods. LLM-as-a-Judge is rather fast and accessible.
- **Conformity:** Human participants frequently disagree on certain LLM responses, while LLM-as-a-Judge typically yields agreements with the majority of participants.
- **Reliability:** Although human participants may have distinct judgment criteria, most LLM-as-a-Judge's evaluations are acceptable for them in the judgment approval test.

3.5 Curating Domain-Specifc Datasets

Before diving into training methods, we introduce four domain datasets. We illustrate the CR processing pipeline in Figure 3 and detail curation steps in Appendix C.1. These datasets include:

- **CR-EVAL** (benchmarking): 200 security-related CRs for evaluating domain-specific capabilities.
- **CR-MIX** ($\mathcal{D}_{\text{DACT}}$): A dataset for continual training on cellular specs and general knowledge.
- **CR-INSTRUCT** ($\mathcal{D}_{\text{TST}}$): CR-converted data for fine-tuning LLMs on domain tasks.
- **CR-SEC** ($\mathcal{D}_{\text{SCT}}$): Security-related CR data for enhancing LLMs' focus on security weaknesses.

4 Domain Specialization via Fine-Tuning

In this section, we resort to fine-tuning to achieve domain specialization. We have also explored prompting-based methods in Appendix B.

In the following, we propose a three-stage training framework that mirrors human expert development, as illustrated in Figure 4. We also propose rationale augmentation in Section 4.2 for converting raw CRs into high-quality training data.

4.1 Three-Stage Training Framework

Stage 1: Domain-Adaptive Continual Training (DACT). As we cannot assume that foundation models have adequately acquired domain knowledge during their initial pre-training, we refine an LLM's learned distribution through continual pre-training on domain data (Gururangan et al., 2020; Zhou et al., 2023; Ghosh et al., 2024). This is modeled as follows:

$$\mathcal{L}_{\text{DACT}}(\theta) = -\mathbb{E}_{x\sim\mathcal{D}_{\text{DACT}}}\left[\sum_{i=1}^{n}\log p_\theta(x_i|x_{<i})\right], \tag{2}$$

where we instantiate $\mathcal{D}_{\text{DACT}}$ with **CR-MIX**.

Stage 2: Task-Specialized Tuning (TST). This stage is designed to help LLM master the basic ability to analyze cellular specifications. Fine-tuning during this stage utilizes our **CR-INSTRUCT** dataset, which encompasses all CR data, and relies on labeled samples:

$$\mathcal{L}_{\text{TST}}(\theta) = -\mathbb{E}_{(x,y)\sim\mathcal{D}_{\text{TST}}}\left[\log p_\theta(y \mid x)\right], \tag{3}$$

where $\mathcal{D}_{\text{TST}}$ represents the **CR-INSTRUCT** dataset. The **CR-INSTRUCT** dataset incorporates diverse task formulations, enabling the model to learn each CR through multiple contexts. This multi-task learning paradigm encourages the model to generalize reasoning skills across tasks.

Stage 3: Security-Centric Tuning (SCT). To tackle **CR-EVAL**, we expect the model to analyze specifications from the security perspective. Inspired by He and Vechev (2023), we frame security-centric analysis as a style-controlled text generation problem. This approach leverages security-related CRs, which reveal real-world security issues, to shape security-centric analysis, enhancing operational feasibility. The loss is defined as:

$$\mathcal{L}_{\text{SCT}}(\delta\theta) = -\mathbb{E}_{(x,y)\sim\mathcal{D}_{\text{SCT}}}\left[\log p_{\theta+\delta\theta}(y \mid x)\right], \tag{4}$$

where $\mathcal{D}_{\text{SCT}}$ denotes security-related task instances of the target task from **CR-SEC**. The parameter $\delta\theta$ corresponds to the additional adapter implemented

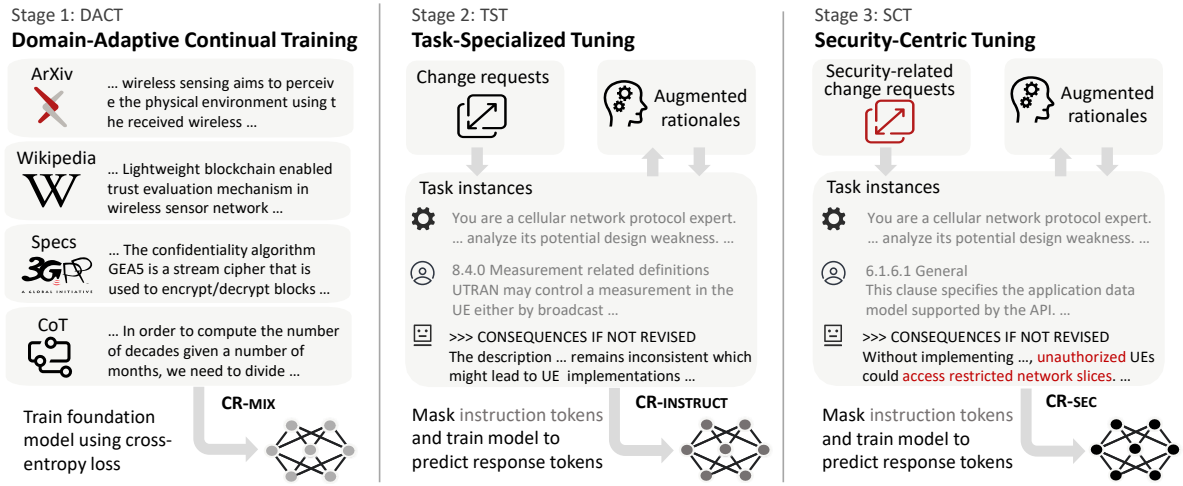

Figure 4: High-level overview of the training framework for domain specialization.

using LoRA (Hu et al., 2022), which allows for efficient adaptation and preserves most of the model's original capabilities (Biderman et al., 2024).

4.2 Rationale Augmentation

Manual inspection reveals a limitation of training samples derived from CRs: rationales R written by human experts in CRs are mainly concise declarative statements, rather than detailed reasoning (see Appendix Figure 8 for an example). This gap hinders effective LLM training, as insufficient rationales lead models to memorize answers instead of developing problem-solving skills (Chung et al., 2024; Kim et al., 2023; Yue et al., 2024).

To address this issue, we introduce rationale augmentation, generating refined, rationale-rich responses for LLM training. Following prior work on training with rationales (Rajani et al., 2019; Zelikman et al., 2022; Kim et al., 2023), we adopt a backward-rationalization strategy: A rationale generator P_o processes a complete task instance—comprising task instruction T, test case Q, and original answer A—and applies backward reasoning to produce a rationale-augmented answer A^*, following augmentation principles C, as formulated by $A^* \leftarrow P_o(C \mid T \oplus Q \oplus A)$. We enforce pedagogically oriented principles C to enhance instructional effectiveness while preserving answer consistency (see Prompt 2).

5 Experiments

5.1 Experimental Setup

Metrics. We evaluate LLM performance on **CR-EVAL** using pass@k (Chen et al., 2021a). The pass@k metric measures the success rate by allowing k independent attempts and considering the best result among the k completions. Given $n \geq k$ completions, where $c \leq n$ completions are correct (*i.e.*, accepted by the LLM-as-a-Judge), the unbiased pass@k score is computed as: $\text{pass@}k := 1 - \frac{\binom{n-c}{k}}{\binom{n}{k}}$ Specifically, we report the cumulative pass@k score over all test cases, with a maximum of 200. Following established practices (Roziere et al., 2023; Chen et al., 2021a; Gu et al., 2024), we set the sampling temperature to 0.8 and top-p to 0.95. Balancing reliability and cost, we sample $n = 10$ completions.

Models. Our evaluation captures the currently highest achievable performance of LLMs, including 17 open-source models and 14 closed-source models. For open-source models, we use the official chat templates. For reasoning models, we set the reasoning efforts to medium. Table 6 details the models.

Rationale augmentation. In this work, we employ `LLaMA-3.1-70B` and `GPT-4o` for rationale augmentation of **CR-INSTRUCT** and **CR-SEC**, respectively. We prompt the rationale generators with a temperature of 0.8 and a top-p of 0.95 to encourage rationale diversity. Our default rationale number per instance is three for TST and five for SCT. Note that we do not augment the reference answers on **CR-EVAL**, avoiding biased evaluation.

Training configurations. Constrained by resource limit, our experiments are primarily on `LLaMA-3.1-8B` (Dubey et al., 2024), a representative 8B model at the inception of the project. Meanwhile, we acknowledge and have verified that fine-tuning on stronger base models can achieve better domain abilities, as explored in Appendix D.6. We fine-tune all parameters for the first two stages,

Table 1: LLMs' performance in **CR-EVAL** across the three domain tasks. We highlight the best performance within open-source and closed-source LLMs, respectively.

Model	Discover Weakness $S_{orig} \rightarrow R$			Outline Revision $S_{orig} + R \rightarrow S'_{rev}$			Reflect Revision $\lvert S_{rev} - S_{orig} \rvert \rightarrow R$		
	pass@1	pass@3	pass@5	pass@1	pass@3	pass@5	pass@1	pass@3	pass@5
Open-Source LLMs									
GLM-4-9B	14.5	23.3	27.8	172.7	188.3	191.0	28.7	50.1	61.3
GLM-4.5	16.0	23.0	26.4	180.2	186.6	188.0	101.6	117.6	122.4
Mistral-7B-v0.3	9.1	16.1	19.8	163.4	182.5	186.5	26.0	44.9	54.9
InternLM-2.5-7B	12.9	25.9	33.5	158.9	185.3	190.8	21.9	42.4	54.7
Qwen-2.5-7B	13.9	24.9	29.9	175.5	189.3	191.6	32.0	55.7	68.2
Qwen-2.5-14B	17.6	27.4	30.8	183.6	193.9	196.0	85.8	119.1	130.1
Qwen-2.5-32B	18.0	28.8	33.7	183.2	190.6	192.6	77.2	106.7	116.6
Qwen-2.5-72B	15.2	22.4	25.7	186.2	195.5	197.7	79.4	105.5	114.1
Qwen3-8B	15.4	25.8	30.8	188.1	195.6	197.2	58.0	85.4	95.6
Qwen3-14B	23.3	37.6	44.5	185.1	194.4	196.5	81.3	112.5	122.7
Qwen3-32B	21.5	35.6	42.1	188.0	195.4	197.2	96.1	130.4	142.5
Qwen3-30B-A3B	18.5	31.8	38.2	188.1	194.2	195.2	72.8	102.3	114.6
Qwen3-235B-A22B	23.5	36.4	41.9	**192.9**	196.7	197.6	111.1	**144.2**	**155.1**
DeepSeek-V3	8.4	13.8	16.7	188.4	195.1	197.0	95.6	121.5	128.8
DeepSeek-R1	9.2	15.8	19.4	192.0	**197.2**	**198.3**	**119.2**	143.7	151.3
LLaMA-3.1-70B	7.4	13.4	16.4	144.4	168.9	174.8	40.5	64.3	76.1
LLaMA-3.1-8B	6.1	13.2	18.1	126.4	164.2	174.0	27.4	48.3	59.8
CRITIC-LLaMA-3.1-8B	**27.2**	**42.3**	**57.8**	160.5	182.4	186.7	106.4	137.9	148.4
Δ *(Relative Increase)*	**+345.9%**	**+220.5%**	**+219.3%**	**+27.0%**	**+11.1%**	**+7.3%**	**+288.3%**	**+185.5%**	**+148.2%**
Closed-Source LLMs									
Doubao-seed-1-6-flash	21.3	33.3	39.1	159.8	173.9	178.0	88.5	117.6	127.9
Doubao-seed-1-6-lite	18.3	28.9	33.7	171.4	182.0	184.6	124.4	153.5	161.3
Doubao-seed-1-6	24.6	41.2	48.4	190.7	197.5	198.5	144.4	169.5	175.1
Doubao-seed-1-6-thinking	42.0	63.0	72.9	189.7	196.0	197.1	158.9	177.7	182.2
Claude-Sonnet-3.5	9.5	16.2	19.3	172.6	182.5	184.9	77.7	106.3	118.1
Gemini-2.0-flash-thinking	79.0	114.8	127.3	166.8	177.6	179.2	139.8	164.0	169.5
Gemini-2.5-flash	61.8	92.1	106.4	181.4	192.0	194.7	158.1	178.0	182.6
Gemini-2.5-pro	62.7	93.0	106.0	185.3	195.6	197.8	172.7	184.8	187.2
GPT-3.5-turbo	11.2	20.1	24.2	146.2	166.3	170.7	42.2	63.6	71.7
GPT-4o-mini	18.2	27.5	31.2	173.0	182.3	183.5	52.4	74.0	81.8
GPT-4o	16.0	25.3	29.2	176.8	186.3	188.0	88.0	113.5	122.6
GPT-o3-mini	**89.0**	**116.6**	**127.9**	186.8	192.5	194.0	132.5	154.4	162.0
GPT-5-mini	73.9	105.4	118.9	**199.7**	**200.0**	**200.0**	171.6	181.0	183.7
GPT-5	75.3	109.8	123.9	196.3	199.0	199.7	**182.7**	**190.2**	**192.6**

DACT and TST. Then, we introduce different LoRA adapters (Hu et al., 2022) ($r = 256$, $\alpha = 512$) for the three domain tasks in SCT. The training ends up with **CRITIC**-LLaMA-3.1-8B[2]. We provide a more detailed list of our training choices in Appendix C.2.

5.2 Evaluation Results

General performance. As shown in Table 1, the three tasks vary in difficulty, from the easiest *outline-revision* to the hardest *discover-weakness*. The *outline-revision* task, primarily a summarization task, is well-handled by most models, with some smaller models (`Qwen-2.5-7B`, `GLM-4-9B`) even outperforming closed-source counterparts (*e.g.*, `GPT-4o`). In contrast, the *reflect-revision* task reveals a significant gap between models, highlighting the challenge of identifying implicit specification weaknesses even when given revisions as hints. The *discover-weakness* task emerges as an extremely challenging task, particularly for open-source models. Proprietary reasoning models (`GPT-o3-mini`, `Gemini-2.0-flash-thinking`) demonstrate superior performance, particularly in the *reflect-revision* and *discover-weakness* tasks. This suggests that reasoning may be a key enabler of strong analysis of sophisticated specifications, which is also emphasized by our rationale augmentation design.

Gap between general and domain-specific capabilities. Several widely recognized LLMs, such as `DeepSeek-R1` and `Claude-3.5-Sonnet`, perform poorly on *discover-weakness*. We also observe inverse scaling (McKenzie et al., 2023) in the `Qwen-2.5` family, where the large 72B model performs the worst in the *discover-weakness* task compared to its smaller-sized cousins. These under-

[2]We name it **CRITIC**, for the model is trained to act as a **critic** for cellular network specifications, and its power can be attributed to **C**hange **R**equests.

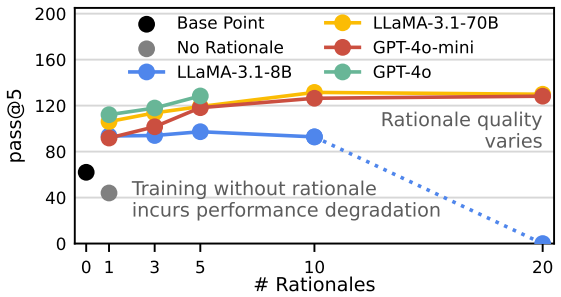

Figure 5: Rationale-dimension scalability.

scores *a potential gap between general-purpose LLMs and domain-specific task requirements* while emphasizing the importance of **CR-EVAL** in helping practitioners identify models with the strongest domain-specific capabilities.

5.3 Performance Analysis of Fine-Tuning

Performance of the domain-specialized LLM. **CRITIC**-LLaMA-3.1-8B nearly triples pass@5 scores of its base model in the *reflect-revision* (+219.3%) and *discover-weakness* (+148.2%) tasks. Notably, it outperforms its **contemporary** proprietary models like `GPT-4o`, solving almost twice as many *discover-weakness* test cases. Despite these advancements, we have to acknowledge that state-of-the-art LLMs that are published more recently, particularly the reasoning models, continue to improve at an unprecedented pace. At the time of writing, closed-source reasoning models like `GPT-o3-mini` have surpassed our domain-specialized medium-sized model, but the fine-tuned model still surpasses open-source reasoning models in *discover-weakness*. Overall, these results highlight the impact of domain-specific fine-tuning in bridging the gap between general LLM capabilities and domain-specific requirements.

Impact and scalability of rationale augmentation. Rationale augmentation offers an additional scaling axis to enhance performance. We evaluate this on **CR-SEC** in the *reflect-revision* task with various rationale generators. The landscape of scaling up rationales is illustrated in Figure 5. *Training without rationales degrades performance, confirming that raw task instances offer limited learning value,* underscoring the necessity of rationale augmentation. Typically, the benefits of incorporating more rationales show evident gains and then reach a plateau. More capable rationale generators yield greater performance gains, likely due to improved knowledge distillation (Gou et al., 2021). An exception is observed with `LLaMA-3.1-8B`, where training with highly diverse rationales (*e.g.*, 20) generated by itself leads to model collapse (see Appendix D.3 for a preliminary explanation).

Table 2: Analysis of token prediction behavior in **CRITIC**-LLaMA-3.1-8B on **CR-EVAL**. Note: '_' denotes the blank character in tokens. A more detailed version is provided in Table 11.

Token	Ratio	Token	Ratio
_safeguard	138.70×	_degrade	79.21×
_improper	58.78×	Failure	61.47×
_mistakenly	39.49×	_challenges	47.90×
_interception	32.64×	_interruptions	24.07×
_inadvertently	21.85×	_reuse	16.60×
_misuse	13.98×	_operational	16.38×
_susceptible	12.34×	_cryptographic	10.98×
_legal	10.49×	_degraded	10.07×

5.4 Analyzing Model Behaviors

We investigate why **CRITIC**-LLaMA-3.1-8B excels on **CR-EVAL** by analyzing its next-token prediction behavior on the *reflect-revision* task. For each test case, we collect softmax-normalized next-token distributions and apply hierarchical aggregation: averaging predictions across tokens within each sample, then across all samples. See Appendix D.2 for more operational details. This yields a single distribution $P_{\text{LLM}} \in \mathbb{R}^{|V|}$ that summarizes the model's overall token-level behavior on **CR-EVAL**. We compare **CRITIC**-LLaMA-3.1-8B with its base model, LLaMA-3.1-8B, highlighting representative differences in Table 2. **CRITIC**-LLaMA-3.1-8B assigns higher probabilities to security-related tokens (*e.g.*, "_safeguard") and shifts toward more specific terminology (*e.g.*, "_interception"), reflecting increased domain specialization.

◆ **Additional experiments.** In Appendix D, we conduct more experiments to (1) ablate the three training stages, (2) scale training data, (3) fine-tune additional backbones (*e.g.*, `GPT-4o-mini`), and (4) study transfer across **CR-EVAL** tasks.

6 Case Study: Can LLMs Replace Humans in Discovering Weaknesses?

To understand this, we test LLMs in discovering 3G/4G/5G vulnerabilities that were previously reported in top-tier security conferences.

Experimental setup. Following Al Ishtiaq et al. (2024), we examine a total of 36 types of existing attacks (see Table 17), collecting relevant specification statements, threat models, and attack descriptions for each. This process is labor-intensive,

Table 3: Examples of **CRITIC**-LLaMA-3.1-8B analyses for the *Incarceration with RRCRELEASE attack* (Hussain et al., 2019b).

Accepted CRITIC-LLaMA-3.1-8B response
The current specification allows the RRCRelease message to be sent unprotected under certain conditions. This poses a significant security risk, particularly if these messages contain sensitive information or instructions that could be intercepted or modified by malicious entities. The inclusion of optional fields such as de-prioritisation information and cell reselection priorities in the RRCRelease message further exacerbates this risk, as these fields can convey critical operational data that, if leaked, could disrupt network performance or lead to unauthorized access.
Rejected CRITIC-LLaMA-3.1-8B response
The current specification allows the RRCRelease message to be sent unprotected, which can lead to significant security vulnerabilities. Since the RRCRelease message may contain sensitive information, such as redirected carrier information or suspend configuration details, sending it unprotected could allow unauthorized entities to intercept this data.

costing around five working days of two students with extensive experience in cellular security. Specification statements were successfully located for 30 attacks, while the remaining were excluded due to reliance on additional implementation or configuration flaws. The examination is conducted as follows: 1) **CRITIC**-LLaMA-3.1-8B (*discover-weakness*) analyzes potential weaknesses across 10 trials. 2) `GPT-4o` evaluates each analysis according to Prompt 6, determining whether combining the discovered weakness with the corresponding threat model sufficiently derives the final attack. 3) We manually validate accepted analyses for reliability.

Quantitative results. **CRITIC**-LLaMA-3.1-8B detects all 30 attack types. In contrast, the conventional formal analysis method Hermes (Al Ishtiaq et al., 2024) detects 19. Due to space constraints, full results are reported in Appendix Table 17.

Qualitative examples. Representative examples of **CRITIC**-LLaMA-3.1-8B's responses are shown in Table 3. The domain-specialized model exhibits the ability to reason about potential weaknesses diversely, which we attribute to the incorporation of multiple rationales during training. Remarkably, even the rejected responses provide valuable insights, unveiling other negative consequences with the unprotected *RRCRelease* message.

Failure analysis. While these results are promising, we also identify several challenges, as discussed in Appendix A, *e.g.*, low calibration due to potential hallucination (Zhang et al., 2023) and requirements for additional preparations.

7 Related Works

LLMs for cellular network. The human-like intelligence of modern LLMs has catalyzed numerous studies on their potential applications in cellular networks. Existing research predominantly investigates whether LLMs can comprehend domain knowledge through question-answering tasks. For example, GSMA has officially launched the Open-Telco LLM Benchmark project (GSMA, 2025b) to evaluate LLMs on interacting with complex standards. Similar efforts include SPEC5G (Karim et al., 2023), TSpec-LLM (Nikbakht et al., 2024), and TeleQnA (Maatouk et al., 2023). Beyond knowledge comprehension, Wen et al. (2024) employ LLMs to detect and explain runtime anomalies in the O-RAN data plane, while Kotaru (2023) investigate their potential for analyzing 5G operator network data. In this work, we stress-test LLMs in a productive setting, evaluating LLMs in refining cellular specifications.

NLP for analyzing cellular network specifications. NLP techniques have been adopted to uncover specification flaws (Chen et al., 2021b, 2022, 2023; Rahman et al., 2024; Al Ishtiaq et al., 2024). Atomic (Chen et al., 2021b) applies textual entailment to detect risky descriptions in 3GPP standards. Several approaches fine-tune encoder models (*e.g.*, RoBERTa (Liu et al., 2019)) for different purposes: CREEK (Chen et al., 2022) identifies security-related CRs, CellularLint (Rahman et al., 2024) detects inconsistencies, and Hermes (Al Ishtiaq et al., 2024) constructs state machines for formal analysis. These methods focus on information extraction rather than direct vulnerability detection.

8 Conclusion

In this work, we pioneer the adoption of LLMs for automated cellular specification refinement. To advance it, we tackle the domain data scarcity challenge by transforming change requests of 3GPP standards into utilizable task instances and formulate three domain tasks. We establish the **CR-EVAL** benchmark, enabling the community to assess the domain-specific capabilities of the rapidly advancing LLMs. What's more, we enhance LLM domain specialization by contributing effective training recipes. Our case studies on 30 known cellular attacks reveal the current status in achieving fully automated cellular specification refinement. This study sheds light on the potential of LLMs and provides a foundation for future advancements.

9 Limitations

Coverage of models. Due to cost constraints, our measurement study does not cover all existing LLMs; instead, we focus on recent state-of-the-art models as representative examples. Future work may include newly released LLMs. For fine-tuning, we mainly focus on LLaMA-3.1-8B, and also experiment with LLaMA-3.1-70B and GPT-4o-mini. These models are all chat-style models. Although we do not explicitly train reasoning models, we hypothesize that they may achieve better performance after domain specialization.

Absence of human baselines. As pinpointed in Appendix E, **CR-EVAL** involves test cases that are related to as many as 74 distinct specifications. Among these, the longest specification can span more than 1,600 PDF pages (*e.g.*, 3GPP TS 38.331 v18.0.2 for 5G RRC). In fact, these specifications are typically studied and updated by different experts or editors. This renders it challenging to recruit experts who can grasp all the involved specifications. However, as test cases are converted from expert-issued change requests, the ground truths can be seen as an ensemble of human performance.

Full completeness of test instances. In this work, we directly convert change requests into test instances. This means that the provided context for LLMs only contains those specification clauses that are directly related to the weakness and thus cannot ensure guarantees of self-inclusiveness. Actually, this is a deliberate choice: As mentioned in Section 3.3, the evaluated capabilities by **CR-EVAL** are not simply limited to the reasoning ability for specification analysis, but extend to domain knowledge about cellular specifications.

Optimal fine-Tuning choices. Our systematic exploration consumed over 32,120 H800 GPU hours, but computational constraints prevented us from exploring other promising directions, *e.g.*, scaling to giant LLMs. Nevertheless, our experiments in Section 5.3 demonstrate the feasibility of extending domain specialization to stronger base models.

Alternative domain specialization techniques. This work mainly explores fine-tuning and prompting to enhance LLMs' domain-specific capabilities. Alternative approaches such as reinforcement learning and agentic AI may further boost domain specialization, which we leave for future work.

Scope of weakness types. Additionally, while our focus is on the security aspects of cellular specifications, our future work could extend LLM-driven specification refinement to address other types of weaknesses following similar methodologies. Meanwhile, this work mainly focuses on cellular specifications, which is a subject of crucial importance. However, our exploration can also be extended to other protocols, *e.g.*, IoT protocol, DNS protocol, or even proprietary protocols.

Cross-lingual evaluation. In this work, we mainly focus on evaluating LLMs in English, as the 3GPP committee uses English as the working language for both specifications and change requests most of the time. An interesting future work is to investigate the domain-specific performance of LLMs in a cross-lingual setting.

10 Ethical Considerations

Our research faithfully adheres to the ethical guidelines established by the Association for Computational Linguistics (ACL)[3]. Our use of AI assistants in this manuscript is limited to writing polishing under full human monitoring.

All change requests and specifications are openly accessible, which we use simply for research purposes. All experiments were performed on publicly available models or API services, ensuring compliance with relevant terms of service.

In this work, our evaluation primarily focuses on existing CRs and known specification weaknesses, both of which have been previously disclosed to relevant stakeholders (e.g., 3GPP councils) and are open to the public. We conducted all experiments and explorations in isolated environments, without allowing autonomy to exploit the weaknesses. Our human study for the LLM-as-a-Judge validity does not contain offensive content, resulting in no harm to the annotators.

Broader implications. This research aims to understand the possibility of applying large language models for automated cellular network specification refinement, which is beneficial to relevant stakeholders and practitioners. We also recognize that the advancement of AI techniques presents a double-edged sword: while offering noteworthy benefits, they may also pose unprecedented threats to human society (Bengio et al., 2024). We should also envision the possible situation where AI systems are given more autonomy. In this context, our work also serves as a systematic and rigorous assessment of the dangerous red-line capabilities

[3] https://aclrollingreview.org/responsibleNLPresearch/

of automated AI systems, specifically discovering and leveraging vulnerabilities within cellular specifications. From another aspect, the application of LLMs for automated cellular specification refinement is not to replace relevant stakeholders but to provide assistance for human experts.

We will release our codebase and benchmark to ensure reproducibility of our results. Meanwhile, it is worth noting that transitioning from a small-scale study to a tool that can be used in the real world requires additional research to ensure the safety, reliability, and efficacy of the technology. Finally, we remind the readers that any techniques introduced in this paper should be applied ethically and within appropriate research contexts.

References

3GPP. 2020. Study on security aspects of the 5G Service Based Architecture (SBA). Technical Report 33.855, 3GPP. Version 16.1.0.

3GPP. 2024. Technical Specification Group working methods. Technical Report 21.900, 3GPP. Version 18.2.0.

3GPP. 2025a. 3gpp standard.

3GPP. 2025b. 6g scenarios and performance requirements.

3GPP. 2025c. Change requests - step-by-step.

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Mujtahid Akon, Tianchang Yang, Yilu Dong, and Syed Rafiul Hussain. 2023. Formal analysis of access control mechanism of 5g core network. In *CCS*.

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 *USENIX Security*.

Zhangir Azerbayev, Hailey Schoelkopf, Keiran Paster, Marco Dos Santos, Stephen Marcus McAleer, Albert Q. Jiang, Jia Deng, Stella Biderman, and Sean Welleck. 2024. Llemma: An open language model for mathematics. In *ICLR*.

Sangwook Bae, Mincheol Son, Dongkwan Kim, CheolJun Park, Jiho Lee, Sooel Son, and Yongdae Kim. 2022. Watching the watchers: Practical video identification attack in {LTE} networks. In *USENIX Security*.

David Basin, Jannik Dreier, Lucca Hirschi, Saša Radomirovic, Ralf Sasse, and Vincent Stettler. 2018. A formal analysis of 5g authentication. In *CCS*.

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. IEEE.

Yoshua Bengio, Geoffrey Hinton, Andrew Yao, Dawn Song, Pieter Abbeel, Trevor Darrell, Yuval Noah Harari, Ya-Qin Zhang, Lan Xue, Shai Shalev-Shwartz, and 1 others. 2024. Managing extreme ai risks amid rapid progress. *Science*.

Nathaniel Bennett, Weidong Zhu, Benjamin Simon, Ryon Kennedy, William Enck, Patrick Traynor, and Kevin RB Butler. 2024. Ransacked: A domain-informed approach for fuzzing lte and 5g ran-core interfaces. In *CCS*.

Dan Biderman, Jacob Portes, Jose Javier Gonzalez Ortiz, Mansheej Paul, Philip Greengard, Connor Jennings, Daniel King, Sam Havens, Vitaliy Chiley, Jonathan Frankle, Cody Blakeney, and John Patrick Cunningham. 2024. LoRA learns less and forgets less. *TMLR*.

Evangelos Bitsikas and Christina Pöpper. 2021. Don't hand it over: Vulnerabilities in the handover procedure of cellular telecommunications. In *ACSAC*.

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

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, and 1 others. 2020. Language models are few-shot learners. In *NeurIPS*.

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*, pages 170–195.

Lichang Chen, Shiyang Li, Jun Yan, Hai Wang, Kalpa Gunaratna, Vikas Yadav, Zheng Tang, Vijay Srinivasan, Tianyi Zhou, Heng Huang, and 1 others. 2024a. Alpagasus: Training a better alpaca with fewer data. In *ICLR*.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, and 1 others. 2021a. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Min-Yue Chen, Yiwen Hu, Guan-Hua Tu, Chi-Yu Li, Sihan Wang, Jingwen Shi, Tian Xie, Ren-Chieh Hsu, Li Xiao, Chunyi Peng, and 1 others. 2024b. Taming the insecurity of cellular emergency services (9-1-1):

From vulnerabilities to secure designs. *IEEE/ACM Transactions on Networking*.

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 *USENIX Security*.

Yi Chen, Di Tang, Yepeng Yao, Mingming Zha, XiaoFeng Wang, Xiaozhong Liu, Haixu Tang, and Dongfang Zhao. 2022. Seeing the forest for the trees: Understanding security hazards in the {3GPP} ecosystem through intelligent analysis on change requests. In *USENIX Security*.

Yi Chen, Yepeng Yao, XiaoFeng Wang, Dandan Xu, Chang Yue, Xiaozhong Liu, Kai Chen, Haixu Tang, and Baoxu Liu. 2021b. Bookworm game: Automatic discovery of lte vulnerabilities through documentation analysis. In *S&P*.

Merlin Chlosta, David Rupprecht, Thorsten Holz, and Christina Pöpper. 2019. Lte security disabled: misconfiguration in commercial networks. In *WiSec*.

Merlin Chlosta, David Rupprecht, Christina Pöpper, and Thorsten Holz. 2021. 5g suci-catchers: Still catching them all? In *WiSec*.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, and 1 others. 2024. Scaling instruction-finetuned language models. *JMLR*.

Tri Dao. 2024. Flashattention-2: Faster attention with better parallelism and work partitioning. In *ICLR*.

DeepMind. 2024. Ai solves imo problems at silver medal level.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Simon Erni, Martin Kotuliak, Patrick Leu, Marc Roeschlin, and Srdjan Capkun. 2022. Adaptover: adaptive overshadowing attacks in cellular networks. In *MobiCom*.

Chongzhou Fang, Ning Miao, Shaurya Srivastav, Jialin Liu, Ruoyu Zhang, Ruijie Fang, Ryan Tsang, Najmeh Nazari, Han Wang, Houman Homayoun, and 1 others. 2024. Large language models for code analysis: Do {LLMs} really do their job? In *USENIX Security*.

Wikimedia Foundation. 2024. Wikimedia downloads.

Sreyan Ghosh, Chandra Kiran Reddy Evuru, Sonal Kumar, S Ramaneswaran, Deepali Aneja, Zeyu Jin, Ramani Duraiswami, and Dinesh Manocha. 2024. A closer look at the limitations of instruction tuning. In *ICML*.

Jianping Gou, Baosheng Yu, Stephen J Maybank, and Dacheng Tao. 2021. Knowledge distillation: A survey. *International Journal of Computer Vision*.

GSMA. 2025a. Gsma intelligence.

GSMA. 2025b. Gsma open-telco llm benchmarks.

Alex Gu, Baptiste Roziere, Hugh James Leather, Armando Solar-Lezama, Gabriel Synnaeve, and Sida Wang. 2024. CRUXEval: A benchmark for code reasoning, understanding and execution. In *ICML*.

Suriya Gunasekar, Yi Zhang, Jyoti Aneja, Caio César Teodoro Mendes, Allie Del Giorno, Sivakanth Gopi, Mojan Javaheripi, Piero Kauffmann, Gustavo de Rosa, Olli Saarikivi, and 1 others. 2023. Textbooks are all you need. *arXiv preprint arXiv:2306.11644*.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, and 1 others. 2025. Deepseek-r1 incentivizes reasoning in llms through reinforcement learning. *Nature*, 645(8081):633–638.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A Smith. 2020. Don't stop pretraining: Adapt language models to domains and tasks. In *ACL*.

Jingxuan He and Martin Vechev. 2023. Large language models for code: Security hardening and adversarial testing. In *CCS*.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, and 1 others. 2022. Training compute-optimal large language models. In *NeurIPS*.

Byeongdo Hong, Sangwook Bae, and Yongdae Kim. 2018. Guti reallocation demystified: Cellular location tracking with changing temporary identifier. In *NDSS*.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *ICLR*.

Syed Hussain, Omar Chowdhury, Shagufta Mehnaz, and Elisa Bertino. 2018. Lteinspector: A systematic approach for adversarial testing of 4g lte. In *NDSS*.

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. In *NDSS*.

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 *CCS*.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.

Imtiaz Karim, Kazi Samin Mubasshir, Mirza Masfiqur Rahman, and Elisa Bertino. 2023. Spec5g: A dataset for 5g cellular network protocol analysis. In *ACL (Findings)*.

Hongil Kim, Jiho Lee, Eunkyu Lee, and Yongdae Kim. 2019. Touching the untouchables: Dynamic security analysis of the lte control plane. In *S&P*.

Seungone Kim, Se Joo, Doyoung Kim, Joel Jang, Seonghyeon Ye, Jamin Shin, and Minjoon Seo. 2023. The cot collection: Improving zero-shot and few-shot learning of language models via chain-of-thought fine-tuning. In *EMNLP*.

Daniel Klischies, Moritz Schloegel, Tobias Scharnowski, Mikhail Bogodukhov, David Rupprecht, and Veelasha Moonsamy. 2023. Instructions unclear: undefined behaviour in cellular network specifications. In *USENIX Security*.

Katharina Kohls, David Rupprecht, Thorsten Holz, and Christina Pöpper. 2019. Lost traffic encryption: fingerprinting lte/4g traffic on layer two. In *WiSec*.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. In *NeurIPS*.

Manikanta Kotaru. 2023. Adapting foundation models for operator data analytics. In *HotNets*, pages 172–179.

Martin Kotuliak, Simon Erni, Patrick Leu, Marc Röschlin, and Srdjan Čapkun. 2022. {LTrack}: Stealthy tracking of mobile phones in {LTE}. In *USENIX Security*.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *SOSP*.

Gyuhong Lee, Jihoon Lee, Jinsung Lee, Youngbin Im, Max Hollingsworth, Eric Wustrow, Dirk Grunwald, and Sangtae Ha. 2019. This is your president speaking: Spoofing alerts in 4g lte networks. In *MobiSys*, pages 404–416.

Chi-Yu Li, Guan-Hua Tu, Chunyi Peng, Zengwen Yuan, Yuanjie Li, Songwu Lu, and Xinbing Wang. 2015. Insecurity of voice solution volte in lte mobile networks. In *CCS*.

Rensis Likert. 1932. A technique for the measurement of attitudes. *Archives of Psychology*.

Xingqin Lin. 2024. 3gpp evolution from 5g to 6g: A 10-year retrospective. *arXiv preprint arXiv:2412.21077*.

Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024. Lost in the middle: How language models use long contexts. *TACL*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Shayne Longpre, Le Hou, Tu Vu, Albert Webson, Hyung Won Chung, Yi Tay, Denny Zhou, Quoc V Le, Barret Zoph, Jason Wei, and 1 others. 2023. The flan collection: Designing data and methods for effective instruction tuning. In *ICML*.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *ICLR*.

Norbert Ludant and Guevara Noubir. 2021. Sigunder: A stealthy 5g low power attack and defenses. In *WiSec*.

Norbert Ludant, Pieter Robyns, and Guevara Noubir. 2023. From 5g sniffing to harvesting leakages of privacy-preserving messengers. In *S&P*.

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. *arXiv preprint arXiv:2310.15051*.

Ian R. McKenzie, Alexander Lyzhov, Michael Martin Pieler, Alicia Parrish, Aaron Mueller, Ameya Prabhu, Euan McLean, Xudong Shen, Joe Cavanagh, Andrew George Gritsevskiy, Derik Kauffman, Aaron T. Kirtland, Zhengping Zhou, Yuhui Zhang, Sicong Huang, Daniel Wurgaft, Max Weiss, Alexis Ross, Gabriel Recchia, and 7 others. 2023. Inverse scaling: When bigger isn't better. *Transactions on Machine Learning Research*.

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)*.

Niklas Muennighoff, Alexander Rush, Boaz Barak, Teven Le Scao, Nouamane Tazi, Aleksandra Piktus, Sampo Pyysalo, Thomas Wolf, and Colin A Raffel. 2023. Scaling data-constrained language models. In *NeurIPS*.

Suresh P. Nair. 2024. 3gpp security assurance (scas) specifications.

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

OpenAI. 2024. GPT-4o. https://openai.com/index/hello-gpt-4o/.

OpenAI. 2025. GPT-5 System Card. https://openai.com/index/gpt-5-system-card/.

CheolJun Park, Sangwook Bae, BeomSeok Oh, Jiho Lee, Eunkyu Lee, Insu Yun, and Yongdae Kim. 2022. {DoLTEst}: In-depth downlink negative testing framework for {LTE} devices. In *USENIX Security*.

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.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. *OpenAI blog*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, and 1 others. 2019. Language models are unsupervised multitask learners. *OpenAI blog*.

Mirza Masfiqur Rahman, Imtiaz Karim, and Elisa Bertino. 2024. {CellularLint}: A systematic approach to identify inconsistent behavior in cellular network specifications. In *USENIX Security*.

Nazneen Fatema Rajani, Bryan McCann, Caiming Xiong, and Richard Socher. 2019. Explain yourself! leveraging language models for commonsense reasoning. In *ACL*.

Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. 2020. Zero: Memory optimizations toward training trillion parameter models. In *SC*.

Muhammad Taqi Raza, Yunqi Guo, Songwu Lu, and Fatima Muhammad Anwar. 2021. On key reinstallation attacks over 4g lte control-plane: Feasibility and negative impact. In *ACSAC*.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. 2024. Gpqa: A graduate-level google-proof q&a benchmark. In *COLM*.

Baptiste Roziere, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Tal Remez, Jérémy Rapin, and 1 others. 2023. Code llama: Open foundation models for code. *arXiv preprint arXiv:2308.12950*.

David Rupprecht, Kai Jansen, and Christina Pöpper. 2016. Putting {LTE} security functions to the test: A framework to evaluate implementation correctness. In *WOOT*.

David Rupprecht, Katharina Kohls, Thorsten Holz, and Christina Pöpper. 2019. Breaking lte on layer two. In *S&P*.

David Rupprecht, Katharina Kohls, Thorsten Holz, and Christina Pöpper. 2020a. Call me maybe: Eavesdropping encrypted {LTE} calls with {ReVoLTE}. In *USENIX Security*.

David Rupprecht, Katharina Kohls, Thorsten Holz, and Christina Pöpper. 2020b. Imp4gt: Impersonation attacks in 4g networks. In *NDSS*.

Thomas Scialom, Tuhin Chakrabarty, and Smaranda Muresan. 2022. Fine-tuned language models are continual learners. In *EMNLP*.

A Shaik, R Borgaonkar, N Asokan, V Niemi, and J Seifert. 2016. Practical attacks against privacy and availability in 4g/lte mobile communication systems. In *NDSS*.

Altaf Shaik, Ravishankar Borgaonkar, Shinjo Park, and Jean-Pierre Seifert. 2019. New vulnerabilities in 4g and 5g cellular access network protocols: exposing device capabilities. In *WiSec*.

Karan Singhal, Shekoofeh Azizi, Tao Tu, S Sara Mahdavi, Jason Wei, Hyung Won Chung, Nathan Scales, Ajay Tanwani, Heather Cole-Lewis, Stephen Pfohl, and 1 others. 2023. Large language models encode clinical knowledge. *Nature*.

Guan-Hua Tu, Yuanjie Li, Chunyi Peng, Chi-Yu Li, Hongyi Wang, and Songwu Lu. 2014. Control-plane protocol interactions in cellular networks. *SIGCOMM*.

Saad Ullah, Mingji Han, Saurabh Pujar, Hammond Pearce, Ayse Coskun, and Gianluca Stringhini. 2024. Llms cannot reliably identify and reason about security vulnerabilities (yet?): A comprehensive evaluation, framework, and benchmarks. In *S&P*.

Fabian Van Den Broek, Roel Verdult, and Joeri De Ruiter. 2015. Defeating imsi catchers. In *CCS*, pages 340–351.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NeurIPS*.

Maurice Weber, Daniel Y. Fu, Quentin Gregory Anthony, Yonatan Oren, Shane Adams, Anton Alexandrov, Xiaozhong Lyu, Huu Nguyen, Xiaozhe Yao, Virginia Adams, Ben Athiwaratkun, Rahul Chalamala, Kezhen Chen, Max Ryabinin, Tri Dao, Percy Liang, Christopher Re, Irina Rish, and Ce Zhang. 2024. Redpajama: an open dataset for training large language models. In *NeurIPS Datasets and Benchmarks Track*.

Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V Le. 2022a. Finetuned language models are zero-shot learners. In *ICLR*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, and 1 others. 2022b. Chain-of-thought prompting elicits reasoning in large language models. In *NeurIPS*.

Haohuang Wen, Prakhar Sharma, Vinod Yegneswaran, Phillip Porras, Ashish Gehani, and Zhiqiang Lin. 2024. 6g-xsec: Explainable edge security for emerging openran architectures. In *HotNets*.

Shijie Wu, Ozan Irsoy, Steven Lu, Vadim Dabravolski, Mark Dredze, Sebastian Gehrmann, Prabhanjan Kambadur, David Rosenberg, and Gideon Mann. 2023. Bloomberggpt: A large language model for finance. *arXiv preprint arXiv:2303.17564*.

Jiarong Xing, Sophia Yoo, Xenofon Foukas, Daehyeok Kim, and Michael K Reiter. 2024. On the criticality of integrity protection in 5g fronthaul networks. In *USENIX Security*.

Hojoon Yang, Sangwook Bae, Mincheol Son, Hongil Kim, Song Min Kim, and Yongdae Kim. 2019. Hiding in plain signal: Physical signal overshadowing attack on {LTE}. In *USENIX Security*.

Jiayi Ye, Yanbo Wang, Yue Huang, Dongping Chen, Qihui Zhang, Nuno Moniz, Tian Gao, Werner Geyer, Chao Huang, Pin-Yu Chen, Nitesh V Chawla, and Xiangliang Zhang. 2024. Justice or prejudice? quantifying biases in LLM-as-a-judge. In *Neurips Safe Generative AI Workshop*.

Chuan Yu and Shuhui Chen. 2019. On effects of mobility management signalling based dos attacks against lte terminals. In *IPCCC*.

Xiang Yue, Xingwei Qu, Ge Zhang, Yao Fu, Wenhao Huang, Huan Sun, Yu Su, and Wenhu Chen. 2024. MAmmoTH: Building math generalist models through hybrid instruction tuning. In *ICLR*.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. 2022. Star: Bootstrapping reasoning with reasoning. In *NeurIPS*.

Yue Zhang, Yafu Li, Leyang Cui, Deng Cai, Lemao Liu, Tingchen Fu, Xinting Huang, Enbo Zhao, Yu Zhang, Yulong Chen, and 1 others. 2023. Siren's song in the ai ocean: a survey on hallucination in large language models. *arXiv preprint arXiv:2309.01219*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, and 1 others. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. In *NeurIPS*.

Chunting Zhou, Pengfei Liu, Puxin Xu, Srinivasan Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, and 1 others. 2023. Lima: Less is more for alignment. In *NeurIPS*.

A Future Works

- **Requirements for effective calibration:** While LLMs' ability to produce diverse interpretations can be useful, it inevitably increases false positives. Their inherent hallucination issues (Zhang et al., 2023) further exacerbate this, making blind reliance infeasible. For weakness verification, current practices primarily delegate this responsibility to human analysts (Chen et al., 2021b; Hussain et al., 2018, 2019b). Extending this practice to LLM-based analysis would be impractical due to the sheer volume of generated weaknesses. Crucially, we argue that even with full autonomy, decision-making should not be ceded to LLMs. Rather than blaming LLMs, the focus should be on developing effective calibration mechanisms to reduce human effort. Without them, unverified proposals risk overwhelming analysts instead of aiding them.
- **Completeness of analyzed clauses:** Our manually curated set of attack-related specification clauses provides an idealized benchmark, containing sufficient context for analysis. This partially explains why **CRITIC**-LLaMA-3.1-8B achieves perfect detection of known attacks despite its limitations in addressing all test cases in **CR-EVAL**. However, practical challenges emerge when refining active specifications, particularly in identifying vulnerabilities arising from complex interactions across multiple sources.

B Exploring Prompting Methods

B.1 Why Fine-Tuning?

In this work, we explore fine-tuning methods to improve LLMs in cellular specification refinement. Several compelling reasons motivate our resorting to fine-tuning: ❶ Refining cellular specifications is reasoning-heavy, knowledge-intensive, and expertise-driven. There exists a gap between LLMs' general-purpose training objectives and the task's specialized requirements. ❷ As we cannot assume that general-purpose LLMs inherently possess all the fundamental components necessary for expert-level analysis, we demand an approach that enables building these capabilities from the ground up. ❸ The cellular security community continuously evolves with ongoing research and new discoveries. This necessitates a scalable approach capable of knowledge ingestion. Fine-tuning effectively fulfills all the expectations outlined above (Chung et al., 2024; Longpre et al., 2023; Wei et al., 2022a). ❹ Furthermore, for domain-specific tasks, fine-tuned models may achieve either higher performance at fixed cost or lower cost at fixed performance. These benefits align with the success of specialized LLMs in various fields, including medicine (Singhal et al., 2023), finance (Wu et al., 2023), and math (Azerbayev et al., 2024).

B.2 Prompting Methods

Another potential method to achieve domain specialization is prompting, which involves crafting well-suited instructions to steer general-purpose LLMs. Prompting is lightweight, as it requires no additional model training. However, it also suffers from limitations, including model capacity bottleneck, lack of systematic methodology, reliance on human expertise, limited performance scalability, restricted transferability across models, and sometimes practical policy constraints.

We conduct experiments to assess the effectiveness of various prompting methods on **CR-EVAL**. Due to space limit, we present details in Appendix B and summarize the main results here: (1) We ask one author to manually rephrase instructions or query `GPT-4o` to refine instructions. These types of prompt engineering yield limited performance improvements, as shown in Table 4. (2) We further explore advanced prompting techniques, including zero-shot CoT (Kojima et al., 2022) and few-shot CoT (Wei et al., 2022b). While zero-shot CoT enhances reasoning density and offers slight improvements (up to 4% in the best cases), these gains remain modest. Few-shot CoT can even degrade performance, particularly for `GPT-4o-mini`, likely due to increased context length and the *lost-in-the-middle* effect (Liu et al., 2024). (3) We also investigate prompting in a human-in-the-loop scenario to assess whether LLMs effectively leverage expert guidance in the *discover-weakness* task:

- **Distilled references:** Emulating expert guidance, we use `GPT-4o` to condense reference answers into single-sentence root cause analyses without weakness disclosure.
- **Enumerable directions:** From 1,922 common root causes of specification weaknesses (*e.g.*, "`poor failure management`"), `GPT-4o` selects the five most relevant as guidance.

We present examples and corresponding results

Table 4: Impact of prompt refinement on *reflect-revision*. "Inst." represents Instruction.

	LLaMA-3.1-8B		CRITIC-LLaMA-3.1-8B	
	pass@1	pass@5	pass@1	pass@5
Default Inst.	27.4	59.8	106.4	148.4
Manual Inst.	22.0	49.8	106.3	146.3
GPT Inst. 1	18.5	45.2	106.4	150.2
GPT Inst. 2	22.0	48.3	105.3	143.7

Distilled references (pass@5: 103.2)

The potential root cause that incurs problems lies in mismatched authentication methods or unsupported features across devices and network components.

Enumerable hints (pass@5: 81.4)

The potential specification issues are: inadequate authentication, authentication mechanisms, protocol misalignment, authentication flaws, specification updates.

Figure 6: Results and examples of incorporating expertise. We test on the *discover-weakness* task, and the pass@5 of the tested checkpoint with no hint is 57.8.

in Figure 6. The findings show that LLM like **CRITIC**-LLaMA-3.1-8B can benefit significantly from additional guidance, achieving up to a 78.5% improvement in the challenging *discover-weakness* task and making the augmented LLM comparable to reasoning models. We hypothesize that expert knowledge serves as external hints, whereas reasoning models generate such hints by themselves, enabling more directed reasoning about specification weaknesses. It is worth noting that while the distilled reference approach assumes access to preliminary high-quality expert analysis, enumerable directions remain easily accessible in production environments. These results highlight another promising pathway for enhancing domain specialization in LLMs by incorporating expert knowledge.

B.3 Why Prompting Methods Fall Short

As for the usage of LLMs, the naive method is to prompt advanced LLMs to solve target tasks in either zero-shot or few-shot manners. However, this approach suffers from several critical limitations. ❶ Prompting remains an art instead of a systematic science, making it challenging to easily craft effective prompts. ❷ Crafting prompts is not effort-free, while the effectiveness heavily relies on human expertise. This contradicts our objective of automated analysis. ❸ Prompts are typically model- and case-dependent, limiting their scalability and transferability across different scenarios and different models. Moreover, the effectiveness of even well-crafted prompts fundamentally depends on the underlying model's capabilities. As such, model limitations will bottleneck outcomes. Our fine-tuning methods directly enhance model abilities. ❹ For more practical considerations, weakness analysis is a sensitive topic, and utilizing third-party LLM services introduces the risk of information leakage. Besides, LLM service providers often enforce strict regulations and may restrict or block security-related queries. This calls for the development and local deployment of specialized LLMs.

B.4 What If Using Prompting Methods

Prompt baselines. As we cannot enumerate all possibilities of prompts, we empirically show the performance of several representative prompt settings. First, we ask one project member to rephrase the default instruction to test the impact of phrasing variance on LLM performance. Second, we utilize `GPT-4o` to refine the default instruction provided in **CR-EVAL** tasks by requesting more LLM-friendly variants, generating two stronger prompt baselines. We follow default testing configurations with only the task instruction altered. The results are shown in Table 4. Comparing different prompt settings, we observe the sufficiency of the default instruction, which simply describes the task plainly. We also notice that the performance of **CRITIC**-LLaMA-3.1-8B is not strongly dependent on the default instruction, which **CRITIC**-LLaMA-3.1-8B encounters in the training stage. Besides the powerful ability obtained through domain-adaptive fine-tuning, another implicit benefit is that it eliminates the need for users to engage in extensive prompt engineering to achieve optimal performance. This is evidenced by the smaller variance of **CRITIC**-LLaMA-3.1-8B's performance across multiple prompt settings.

CoT prompting. We explore the impact of recognized reasoning-enhancing techniques. Specifically, we evaluate two representative approaches, few-shot CoT (Wei et al., 2022b) and zero-shot CoT (Kojima et al., 2022), applying them to `LLaMA-3.1-8B`, `GPT-4o-mini`, and **CRITIC**-LLaMA-3.1-8B. We instantiate few-shot CoT with three randomly sampled training samples, each with augmented rationales. As shown in Figure 7, the incorporation of zero-shot CoT enhances reasoning density, leading to higher pass rates across all three models. We observe that introducing a few-shot CoT may adversely impact the performance of models. The task instances typically span long context, *e.g.*, the 3-shot setting additionally costs 8,287 tokens of `GPT-4o-mini`. We hypothesize that

Table 5: Training configurations for **CRITIC**-LLaMA-3.1-8B.

	Stage 1 (DACT)	Stage 2 (TST)	Stage 3 (SCT)
Corpus	**CR-MIX**	**CR-INSTRUCT**	**CR-SEC**
Training method	Pre-training	Supervised	Supervised
Learnable parameters	Full parameters	Full parameters	LoRA ($r = 128, \alpha = 256$)
Learning rate	2e-6	2e-5	1e-4
Global batch size	256	128	64
Weight decay	0%	10%	0%
Gradient clipping	1.0	1.0	1.0
Training epoch	1	1	1
Parameter precision	BF16 + TF32	BF16 + TF32	BF16 + TF32
Warmup ratio	10%	3%	3%
Scheduler type	Cosine	Cosine	Cosine
Max sample length	512	12,000	12,000

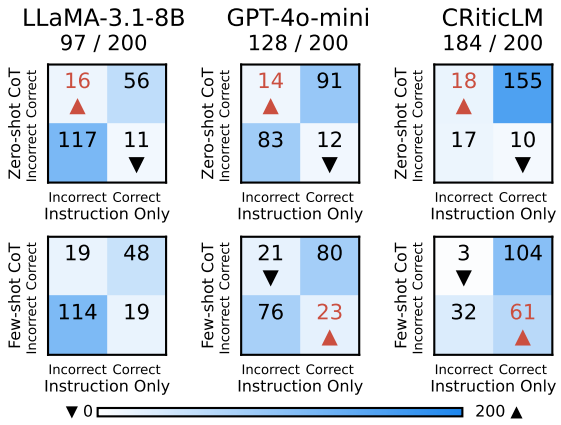

Figure 7: Impact of CoT prompting, conducted on *reflect-revision*. We use pass@10 to ease the sample-wise comparison and report the overall performance under diverse CoT settings for model-level comparison.

the performance degradation stems from the *lost-in-the-middle* phenomenon inherent in LLMs (Liu et al., 2024), where long or irrelevant context will lead LLMs to behave worse. Besides, the inference cost also increases with additional shots.

C Complementary Experimental Setup

C.1 Curating Domain-Specifc Datasets

C.1.1 CR-EVAL

Collection. We first query the official database[4] to obtain a complete list of CRs, filtering only those approved by *TSGs* to ensure content reliability. A parallel crawler queries the CR search service[5], retrieves FTP paths, and downloads raw CR files from the 3GPP FTP server[6], yielding 205,374 valid CRs. Revisions within *doc/docx* files are tracked via Office Word's *Track Changes* mode. Despite format evolution, 3GPP has maintained standardized CR coversheets. We implemented a parsing script to extract key elements, discarding change requests that failed to process. This resulted in 189,904 structured change requests.

Annotating security relevance. We implement an LLM-based process to annotate the security relevance of CRs based on expert rationales R. Using `LLaMA-3.1-70B` with the instruction in Prompt 3, this approach identifies 4,869 security-related CRs. To ensure precision, we cross-reference these annotations with security-related CRs documented by (Chen et al., 2022), yielding 529 intersecting cases. This is followed by a manual verification, as a small portion of CRs prove unsuitable as task instances. For example, CRs proposing entirely new statements have empty original statements S_{orig}, rendering them invalid for the *discover-weakness* task. Ultimately, we curate a set of 200[7] high-quality security-related CRs for **CR-EVAL**.

C.1.2 CR-INSTRUCT and CR-SEC

After obtaining security-related CRs for evaluation, we have 185,035 security-unrelated CRs and 4,669 security-related CRs for the training set. We convert all the change requests into task instances based on task formatting templates, which we introduce in Section 3.2.

[4] https://www.3gpp.org/ftp/Information/Databases/

[5] http://netovate.com/cr-search/

[6] https://www.3gpp.org/ftp

[7] The success of compact benchmarks like HumanEval (Chen et al., 2021a) (164 examples) and GPQA (Rein et al., 2024) (448 examples) demonstrates that small benchmarks can be effective, with faster and cheaper evaluation as an additional advantage.

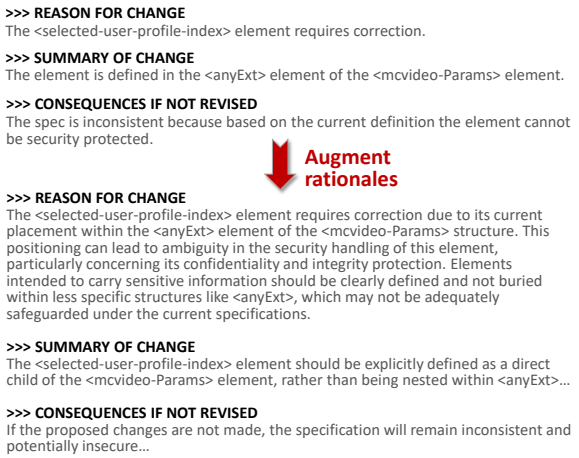

Figure 8: Example of rationale augmentation.

Decontamination. To precisely reflect the benefits of domain-adaptive fine-tuning, we try our best to minimize leakage of test cases in **CR-EVAL**. Following established practices (Brown et al., 2020; Achiam et al., 2023; Radford et al., 2019), we employ a rigorous and proactive decontamination strategy at the level of task instances. We exclude training samples that exhibit 20-gram overlaps with any test case answers, where a gram is defined as a lowercase, whitespace delimited word. This approach prevents both direct test case leakage and the occurrence of suspicious task instances. Furthermore, we remove training samples associated with existing attacks discussed in Section 6 using the same 20-gram matching criterion. From this point on, the task instances of **CR-EVAL** are frozen and isolated.

Cleaning. Invalid task instances, as discussed in processing **CR-EVAL**, are also present in the training set. To address this, we down-sample task instances through a two-step filtering process. First, we exclude invalid instances based on heuristic rules (*e.g.*, extremely short queries and missing task placeholders). Second, inspired by (Gunasekar et al., 2023; Dubey et al., 2024; Chen et al., 2024a), we implement another semantic filtering to remove low-quality samples and those irrelevant to specification weaknesses. We use `LLaMA-3.1-70B` to evaluate their educational value for specification analysis, following Prompt 4. Instances deemed to lack educational value are removed, and the remaining samples constitute our **CR-INSTRUCT** dataset. We clone security-related samples to create **CR-SEC**, comprising three subsets, each aligned with a specific domain task.

C.1.3 CR-MIX

We incorporate **3GPP standards** to enhance the LLM's comprehension of cellular networks. Concretely, we utilize the *python-docx* library to extract the main body of 2,445 specifications from the TSpec-LLM dataset (Nikbakht et al., 2024). These specifications, spanning the 21 to 55 series and ranging from Release 8 to Release 19, cover essential aspects of cellular networks. We retain tables and figure captions while omitting figures due to intractability. We also borrow a general-domain reasoning enhancement dataset, the **CoT collection dataset** (Kim et al., 2023). To mitigate catastrophic forgetting (Scialom et al., 2022), we include the **Wikipedia dataset** (Foundation, 2024) and the **ArXiv split** from the RedPajama dataset (Weber et al., 2024). We filter these general-domain datasets using keyword-based heuristics to identify documents specifically relevant to cellular networks and security, ensuring focused domain adaptation.

C.2 Training Configurations

We list main training configurations in Table 5. We choose a high rank $r = 256$ for the LoRA adapter and set $\alpha = 2r$ as recommended by Biderman et al. (2024). We apply LoRA adapters to all linear layers in the model. All three training stages employ AdamW optimizer (Loshchilov and Hutter, 2019) with β_1 as 0.9 and β_2 as 0.999. The settings of learning rates are 2×10^{-6} for DAPT, 2×10^{-5} for TST, and 1×10^{-4} for SCT. For batch size, we employ 256 for DAPT, 128 for TST, and 64 for SCT. All the training stages consume one epoch with the analogous learning scheduler: the learning rate is linearly warmed up for several training steps, and then cosine-decreases to $1/20$ of the peak learning rate. Gradient accumulation is adopted to achieve large batch sizes with constrained GPU memory.

Compute infrastructure. All experiments were conducted on a server running Ubuntu 20.04.5 LTS operating system. The machine is equipped with an Intel Xeon Platinum 8468V processor (96 cores, 192 threads), 2 TB of system memory, and 8 NVIDIA H800 GPUs with 80 GB of VRAM each.

Software. Our project is implemented based on Python 3.12, CUDA 11.8, PyTorch 2.4.0, and HuggingFace's transformer library. To accelerate training, we achieve data parallel through DeepSpeed (Rajbhandari et al., 2020): we adopt ZeRO stage-2 with a world size of 4 for 8B models and

Table 6: Models evaluated in this work.

Model	Model size	Open-source	Context window
GLM-4-9B	9B	Y	128K
GLM-4.5	MoE	Y	128K
Mistral-7B-v0.3	7B	Y	128K
InternLM-2.5-7B	7B	Y	1M
Qwen-2.5-7B	7B	Y	128K
Qwen-2.5-14B	14B	Y	128K
Qwen-2.5-32B	32B	Y	128K
Qwen-2.5-72B	72B	Y	128K
Qwen3-8B	8B	Y	128K
Qwen3-14B	14B	Y	128K
Qwen3-32B	32B	Y	128K
Qwen3-30B-A3B	MoE	Y	128K
Qwen3-235B-A22B	MoE	Y	128K
DeepSeek-V3	MoE	Y	128K
DeepSeek-R1	MoE	Y	128K
LLaMA-3.1-70B	70B	Y	128K
LLaMA-3.1-8B	8B	Y	128K
Doubao-seed-1-6-flash	Unknown	N	256K
Doubao-seed-1-6-lite	Unknown	N	256K
Doubao-seed-1-6	Unknown	N	256K
Doubao-seed-1-6-thinking	Unknown	N	256K
Claude-3.5-Sonnet	Unknown	N	200K
Gemini-2.0-flash-thinking	Unknown	N	1M
Gemini-2.5-flash	Unknown	N	1M
Gemini-2.5-pro	Unknown	N	1M
GPT-3.5-turbo	Unknown	N	16,385
GPT-4o-mini	Unknown	N	128K
GPT-4o	Unknown	N	128K
GPT-o3-mini	Unknown	N	200K
GPT-5-mini	Unknown	N	400k
GPT-5	Unknown	N	400k

Table 7: Module-level lines of code counted using *cloc*.

Component	**LoC**
Data collection and processing	4,946
Training	3,419
CR-EVAL	2,203
User study	2,241
Total	12,809

stage-3 with a world size of 8 for 70B models. We use Flash-Attention 2 (Dao, 2024) to improve throughput and use gradient checkpointing to reduce memory requirements. For evaluation, we deploy inference endpoints using vLLM (Kwon et al., 2023). The entire project consumes around 12,800 lines of code, decomposed in Table 7.

D Additional Experiment Results

D.1 Correlation between CR-EVAL Tasks

We explore the correlation between the **CR-EVAL** tasks by evaluating whether the knowledge acquired by training on the source task can transfer to the target task. We train base models with **CR-SEC** of the source task and evaluate the trained model in the target task. The results are shown in Table 8. We notice that the knowledge is clearly transferable between tasks, which substantiates the efficacy of our multi-task learning design in the TST stage.

Table 8: Cross-task performance: models are trained on source tasks with **CR-SEC** and evaluated on target tasks using pass@5.

		Source		
		Outline Revision	**Reflect Revision**	**Discover Weakness**
Target	Outline Revision	177.9	183.6	169.8
	Reflect Revision	77.8	124.2	74.4
	Discover Weakness	31.9	27.2	33.5

D.2 Tracking Model Behaviors

We study why **CRITIC**-LLaMA-3.1-8B can excel in cellular specification refinement and, consequently, on **CR-EVAL**. We analyze the model's behavior by collecting next-token predictions during processing the *reflect-revision* task of **CR-EVAL**. Formally, we obtain a set of softmax-normalized next-token prediction probabilities denoted as $p_i^j \in \mathbb{R}^{|V|}, i \in [1, S_j], j \in [1, N]$, where N denotes the number of test cases in **CR-EVAL**, S_j represents the sequence length of predicted tokens for the j-th test case under greedy search, and $|V|$ is the vocabulary size. To mitigate varying completion lengths, we perform hierarchical aggregation: computing the mean prediction distribution within each sample ($\frac{1}{S_j}\sum_{i=1}^{S_j} p_i^j$), and then averaging across all N samples. This processing condenses the LLM's behavior on **CR-EVAL** into a single probability distribution $P_{\text{LLM}} \in \mathbb{R}^{|V|}$, where each dimension represents the model's averaged behavior for a vocabulary token. Due to the huge vocabulary size, *e.g.*, $|V| = 131,072$ for `LLaMA-3.1` models, we focus on tokens with probabilities higher than $\frac{1}{|V|}$, which represent frequently used vocabulary in LLM outputs. We conduct a comparative analysis between $P_{\textbf{CRITIC}-LLaMA-3.1-8B}$ and $P_{\text{LLaMA-3.1-8B}}$, with key observations presented in Table 11. Significantly, **CRITIC**-LLaMA-3.1-8B yields higher probabilities for security-related tokens, and notably transitions from employing generic descriptions (*e.g.*, "_errors" and "_risks") to more specific terminologies (*e.g.*, "_interception" and "_confidentiality"). As the P_{LLM} is normalized, an increase in certain token probabilities inevitably results in the reduction of others. This transition aligns with our objective of developing a more domain-specialized model.

D.3 Exploring Diversity Gain of Rationales

Building upon our scalability analysis rationales in Section 5.3, we study why training with more

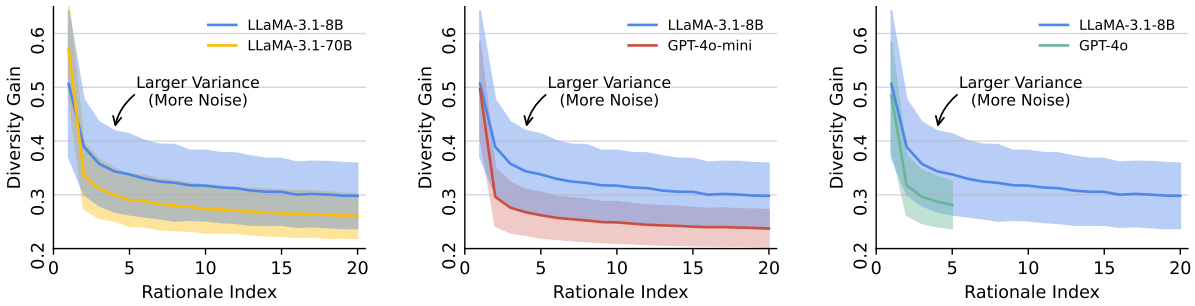

Figure 9: Patterns of diversity gain when augmenting rationales using different models: `LLaMA-3.1-8B`, `GPT-4o-mini`, `GPT-4o`, and `LLaMA-3.1-70B`. The rationales correspond to the *reflect-revision* task. We use Euclidean distance to measure the similarity between rationales while the sample-level diversity gain is measured as the minimal distance of the i-th rationale against the set of the previous $(i-1)$ rationales plus the original answer. We plot the variance across different task instances.

rationale-augmented answers can benefit **CRITIC**-LLaMA-3.1-8B and why this improvement finally plateaus. We analyze the semantic differences between rationales by measuring their distances in the embedding space. We employ a feature extractor (*i.e.*, OpenAI's text-embedding-3-large) to project each rationale into the embedding space. We then observe the diversity gain brought by progressively adding new rationales. The diversity gain corresponding to the i-th rationale of the j-th task instance is defined as the minimum distance between the new rationale and the union of existing rationales and the original answer, formalized as: $\min_{k<i} |r_{i,j} - r_{k,j}|_2 \cup |r_{i,j} - a_j|_2$ where $r_{i,j}$ denotes the i-th rationale for the j-th task instance, a_j is the original answer, and $|\cdot|_2$ represents the Euclidean distance in the embedding space. Our analysis in Figure 9 reveals that the marginal diversity gain per new rationale diminishes as the number of rationales increases. This trend correlates with the observed improvement of **CRITIC**-LLaMA-3.1-8B as we increase the rationale number. We stipulate that this convergence occurs because additional rationales fail to introduce new insights, and the remaining diversity gains stem from the altered wordings. Notably, when using `LLaMA-3.1-8B` as the rationale generator, we observe diversity gains with both a higher mean and significantly greater variance compared to other models. This observation partially explains the inferior performance of `LLaMA-3.1-8B` as the rationale generator and provides insights for choosing rationale generators.

D.4 Ablating Training Stages

To assess the contributions of each training stage, we conduct an ablation study. As shown in Table 9, each stage positively contributes to **CRITIC**-LLaMA-3.1-8B's performance on **CR-EVAL** tasks. The SCT stage, closely aligned with security-centric analysis, yields the most significant improvement. Training with security-irrelevant samples in TST also enhances performance by improving generalization in addressing specification weaknesses. The DACT stage, designed to complement LLMs with domain knowledge, provides modest gains, likely because base LLMs (*e.g.*, `LLaMA-3.1-8B`) already possess relevant knowledge (Zhou et al., 2023).

Table 9: Impact of training stages, with rigorous decontamination to minimize memorization.

	Reflect Revision		Discover Weakness	
	pass@1	pass@5	pass@1	pass@5
LLaMA-3.1-8B	27.4	59.8	6.1	18.1
+ SCT	73.3	124.2	12.7	33.5
+ TST + SCT	95.8	145.3	25.3	49.9
+ DACT + TST + SCT	106.4	148.4	27.2	57.8

D.5 Scaling Along the Data Dimension

We examine how model performance evolves with increasing training data volume. We focus on the *reflect-revision* task, as it is the most distinguishing task in **CR-EVAL**. Training data scales approximately logarithmically, with each data point representing a full training run using default hyperparameters. As illustrated in Figure 10, results reveal two key trends: 1) performance consistently improves with more data, and 2) performance gains exhibit diminishing returns, aligning with observations of established scaling laws (Kaplan et al., 2020; Hoffmann et al., 2022; Dubey et al., 2024).

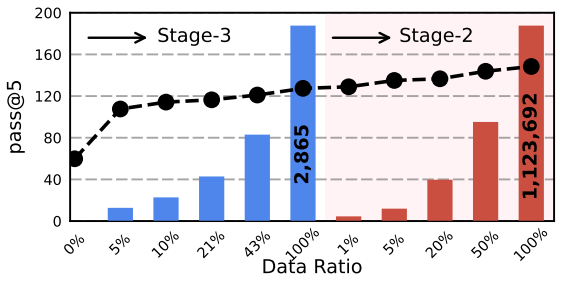

Figure 10: Data-dimension scalability.

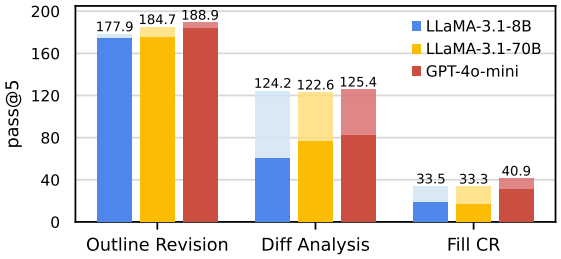

Figure 11: Extensibility to advanced base models.

D.6 Extensibility to Advanced Base Models

In our experiments, we primarily focus on the LLM with a "small" parameter count, LLaMA-3.1-8B. We extend the domain-adaptive fine-tuning to closed-sourced `GPT-4o-mini`[8] and the `LLaMA-3.1-70B` with a larger parameter count, using the **CR-SEC**. As shown in Figure 11, domain-adaptive fine-tuning yields performance improvements across all models. Interestingly, `LLaMA-3.1-8B` and `LLaMA-3.1-70B` converge to similar levels, indicating that domain data quality, rather than model size, is the primary bottleneck in certain cases. Meanwhile, `GPT-4o-mini` consistently maintains superior performance, particularly on the *discover-weakness* task. This suggests that applying our training methodology to advanced foundation models could further specialize LLMs.

D.7 Examining Existing Attacks

Considered threat models. The settings of the threat model strictly follow their corresponding original papers. In gross, we consider both passive and active attacker models.

- **Passive adversary:** This attacker can eavesdrop on over-the-air radio broadcast channels, such that they can analyze and deduce information from intercepted messages.
- **Active adversary:** This attacker can establish and operate a rogue base station to inject malicious traffic directed at UEs. While they are assumed to have full knowledge of the protocol specifications, they lack access to cryptographic keys, except for public keys.

For certain scenarios, we suppose the attacker also knows some identity information of the victim UE, like the C-RNTI. Alternatively, the attacker might have a hypothesis about the victim's identity information and seek to verify it.

[8]Fine-tuning GPT models is officially accessible via `https://platform.openai.com/finetune`. Considering cost affordability, we select GPT-4o-mini as a representative example of closed-source models.

E Structural Analysis of Datasets

We demonstrate the representativeness of **CR-EVAL** for evaluating cellular specification refinement. We also provide statistics about the training dataset size.

Sample length of CR-EVAL. We analyze the sample length distribution of test cases in **CR-EVAL** at the token level, as demonstrated in Figure 12. The test cases of **CR-EVAL** typically contain thousands of or even tens of thousands of tokens, presenting rigorous challenges that specifically test models' long-context capabilities. Moreover, LLMs suffer the problem of *lost-in-the-middle*, meaning that the models claiming long context cannot effectively leverage the information given in their context (Liu et al., 2024). Models without sufficiently effective long context (less than the claimed maximum token number) cannot tackle the test cases in a single inference.

Release coverage of CR-EVAL. We provide the statistics of the target releases of CRs in **CR-EVAL**, which is illustrated in Figure 13. **CR-EVAL** demonstrates extensive release coverage and excellent diversity, with its involved CRs spanning from Release 5 to Release 17. Covering CRs of old releases in **CR-EVAL** is necessary as historical security vulnerabilities can provide insights for refining contemporary cellular specifications. Intriguingly, we observed more security-related CRs during Release 8 and Release 15, which correspond to the introductions of LTE and 5G, respectively (Chen et al., 2022). This underscores the importance of automated cellular specification refinement methods, particularly during major technological transitions. To ensure rigorous identification of security-relevant CRs, we cross-referenced our annotations with those from Chen et al. (2022). While this early decision enhanced the reliability of our security relevance annotations, it also brought about an unexpected consequence, specifically, **CR-EVAL**

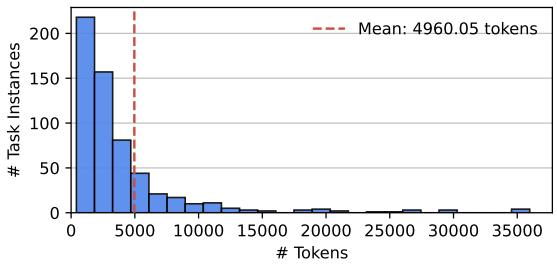

Figure 12: Distribution of token counts of test cases in **CR-EVAL**, with three tasks combined.

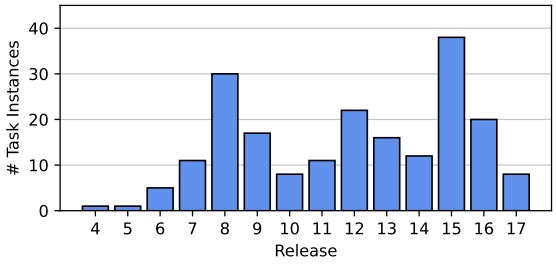

Figure 13: Release distribution of CRs in **CR-EVAL**.

	# Samples	# Tokens	# Response tokens
CR-EVAL (*outline-revision*)	200	898,521	15,642
CR-EVAL (*reflect-revision*)	200	1,194,454	73,239
CR-EVAL (*discover-weakness*)	200	883,054	73,239
CR-MIX (shared)	3,729,713	1,433,683,482	1,433,683,482
CR-INSTRUCT (shared)	1,123,692	4,892,550,046	661,498,363
CR-SEC (*outline-revision*)	13,860	44,085,131	3,087,016
CR-SEC (*reflect-revision*)	14,325	56,114,617	5,427,157
CR-SEC (*discover-weakness*)	15,370	53,544,878	6,305,663

Table 10: Token statistics of the datasets at both the sample and token level, based on the tokenizer of the `LLaMA-3.1` herds (Dubey et al., 2024).

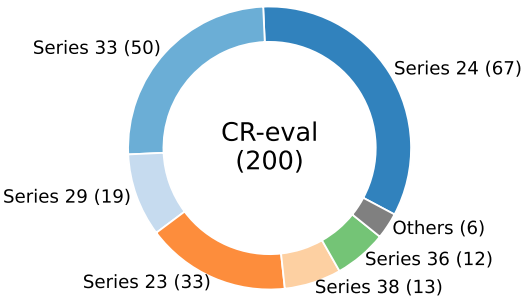

Figure 14: Distribution of CRs in **CR-EVAL** across different specification series.

currently excludes test cases from Release 18 onward. Future work will address this by developing an extended version of **CR-EVAL** incorporating more recent CRs.

Specification coverage of CR-EVAL. CR-EVAL encompasses 200 CRs distributed across 74 distinct specifications, demonstrating its extensive scope. We provide a coarse-grained summary of the specification distribution according to the belonging to standard series in Appendix E. Unlike previous works that primarily focus on a limited set of specifications such as NAS and RRC, **CR-EVAL** provides comprehensive coverage across the 3GPP ecosystem. Yet the broad coverage makes it impractical for us to establish a human-level baseline on **CR-EVAL**.

Token number of all datasets. We present the dataset decomposition in Appendix E. Note that we count the token number of data after rationale augmentation. As the auto-regressive LLMs are trained through the next token prediction task (Radford et al., 2018), the dataset size at the token level can more precisely show how much the LLMs can learn from the training. For the continual pre-training paradigm, *e.g.*, DACT in our framework, all tokens are learnable while only the response tokens can be learned for the supervised fine-tuning paradigm. Scaling law (Kaplan et al., 2020; Hoffmann et al., 2022; Muennighoff et al., 2023) demonstrates that LLMs can consistently gain benefits through continual training investment. An implicit side of the scaling law is what the training dataset teaches the model. That's the rationale behind our finding that a limited number of security-related domain data **CR-SEC** contributes significantly to the performance improvement on **CR-EVAL**. This underscores the crucial importance of developing high-quality domain datasets closely relevant to our target task, cellular specification refinement.

Difficulty of CR-EVAL. The difficulty of test cases can be naturally measured through their *global solv-*

ing rate, defined as the proportion of model trials capable of solving them. We reuse the model predictions in Section 5.2 and aggregate the solving rates across all model trials (10 trials per model). We provide the statistics in Figure 18. The final **CR-EVAL** aligns with our design principle of progressive challenge levels. At a macro level, the three tasks exhibit distinct difficulty tiers, as evidenced by their mean solving rates. For example, the *discover-weakness* task presents the highest challenge by providing the model with minimal information while demanding an in-depth understanding of potential weaknesses within the specification clauses. At a micro level, each task comprises test cases of varying difficulty, as substantiated by our demonstrations in Section 5.2. While all test cases passed our manual verification process, ensuring that they provide sufficient information for task completion, they incorporate different implicit confounding factors, *e.g.*, the provided context specification statements and the expected response quality. Among the three tasks, both the *reflect-revision* and *discover-weakness* are challenging enough to differentiate LLMs' domain-specific abilities, despite rapid advances in LLM development.

F Human Study for LLM-as-a-Judge

We conducted the human study with eight PhD students specializing in network security. Their research experience in the field ensures the quality of our evaluation. All participants volunteered and were willing to contribute their annotations to the community. The human study concerning the reliability of LLM-as-a-Judge consists of two rounds, whose system snapshots are presented in Figure 19 and Figure 20: 1) Alignment test: Participants were presented with 25 samples, each consisting of an LLM response and the corresponding reference answer. Participants are tasked to *accept* or *reject* the LLM responses based on their alignment with the reference answers. This round evaluates Human-as-a-Judge under the fair setting with LLM-as-a-Judge, aiming to test their alignment. 2) Judgment approval test: Participants are additionally presented with the LLM-as-a-Judge's judgment and its posterior explanation. Participants are asked to *approve* or *disapprove* the LLM-as-a-Judge's judgment. This round aimed to calibrate the rigor degree of humans and show the acceptableness of LLM-as-a-Judge's decisions from the perspective of human annotators.

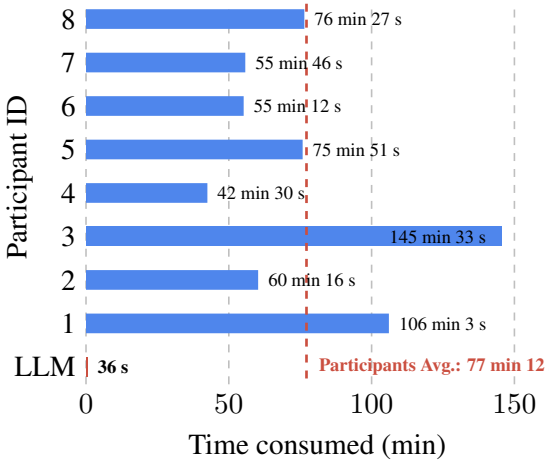

Figure 15: Time consumed (in minutes) by the LLM and each participant during the study.

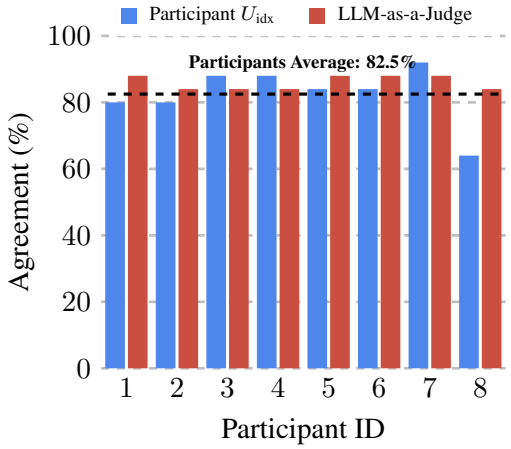

Figure 16: Results of the leave-one-annotator-out experiments comparing agreement between individual annotators and majority decisions. For each test case, we evaluate the agreement between a single annotator (either a human participant or LLM-as-a-Judge) and the majority vote of the remaining $N-1$ participants. The majority vote serves as the consolidated judgment from the excluded annotators.

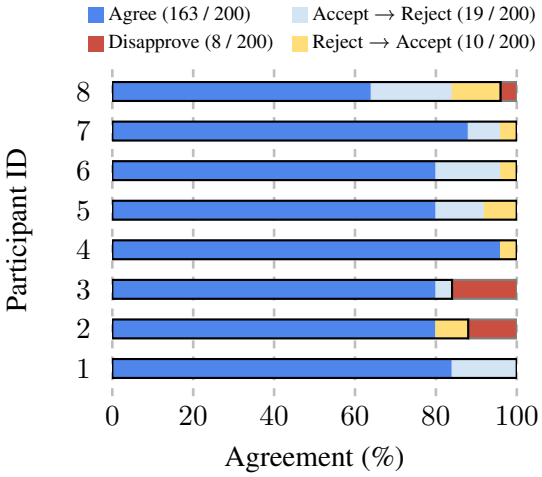

Figure 17: Participants' agreement with the LLM before and after receiving the LLM's explanations.

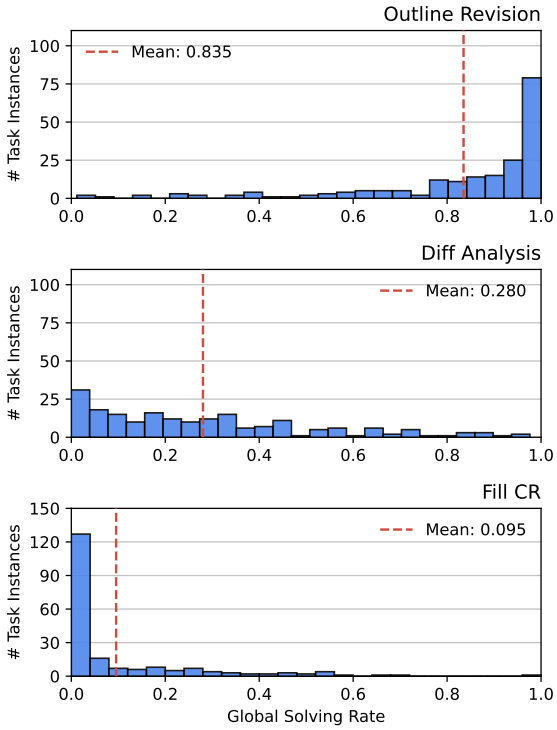

Figure 18: Solving rates of all models on the three tasks.

The 25 LLM responses used in the human study belong to `GPT-4o` and a preview version of **CRITIC**-LLaMA-3.1-8B for *reflect-revision* and *discover-weakness*. We randomly sample 12 acceptable and 13 unacceptable samples based on the LLM-as-a-Judge to ensure representativeness. We collected a total of $8 * 25 * 2 = 400$ responses, which are presented in Table 12. The time consumed by the LLM and each participant during the study is shown in Figure 15. The LLM completed the task in 36 seconds, significantly faster than the participants, who averaged 77 minutes and 12 seconds. This suggests the unavailability of a large-scale study with human judgments and highlights the LLM-as-a-Judge's potential for time-efficient automation in evaluating LLM responses (Zheng et al., 2023; Chen et al., 2024a; Ye et al., 2024). The percentage of agreement between each participant and the LLM-as-a-Judge with the remaining participants is shown in Figure 16. For example, we compare Participant 1 and LLM-as-a-Judge with the consensus of Participants 2-8. The LLM generally achieved a higher agreement rate, comparable to the participants' average of approximately 82.5%. This indicates that the LLM's judgments align closely with participant consensus, supporting its reliability as an automated judge. Figure 17 presents participants' agreement with the LLM before and after receiving its explanations. Of 200 decisions, 163 consistently align with the LLM. Notably, 19 cases shifted from approval to rejection after the explanation, 10 shifted the opposite way, and 8 remained disapproved. These patterns suggest that human participants and the LLM-as-a-Judge may hold different judgment criteria, which are effectively calibrated through the explanations provided in the judgment approval test. These findings demonstrate our finally instantiated LLM-as-a-Judge's reliability in automating the evaluation of model answers at a notable expert level.

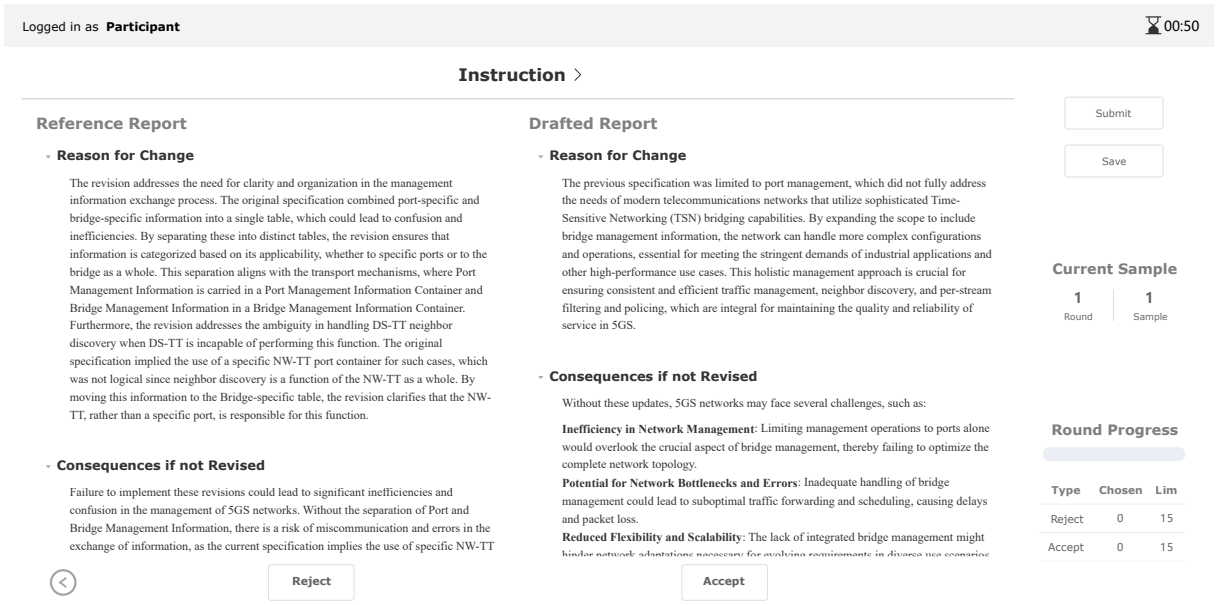

Figure 19: System snapshot of the first round of human study, Alignment Test.

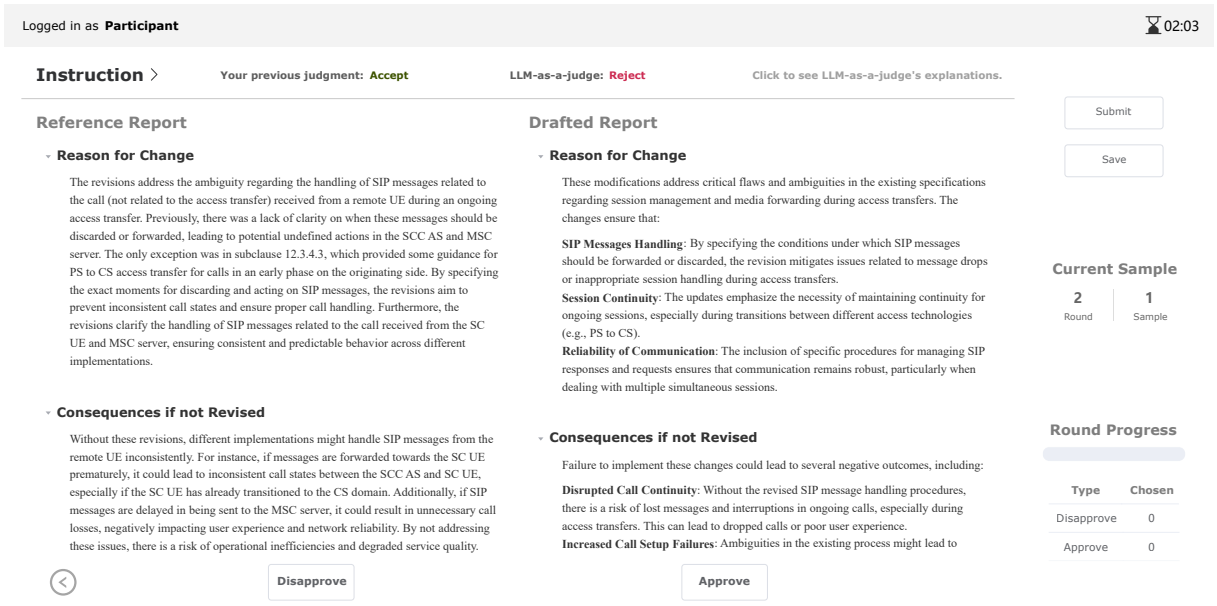

Figure 20: System snapshot of the second round of human study, Judgment Approval Test.

Table 11: Full list of **CRITIC**-LLaMA-3.1-8B's token prediction behavior on **CR-EVAL**. The ratios are relative to the base model (LLaMA-3.1-8B). Note: '_' denotes the blank character in tokens.

Token	Ratio	Token	Ratio
_safeguard	138.70×	_degrade	79.21×
_improper	58.78×	Failure	61.47×
_mistakenly	39.49×	_challenges	47.90×
_interception	32.64×	_interruptions	24.07×
_inadvertently	21.85×	_reuse	16.60×
_misuse	13.98×	_operational	16.38×
_susceptible	12.34×	_cryptographic	10.98×
_legal	10.49×	_degraded	10.07×
_disrupt	6.90×	_misunderstanding	6.5×
_unintended	6.77×	_privacy	5.43×
_fail	4.66×	_protecting	4.66×
_unprotected	4.31×	_risk	3.82×
_ambiguity	3.70×	_invalid	3.73×
_spoof	3.41×	_trust	3.62×
_reliability	3.29×	_disruptions	3.10×
_incorrectly	2.88×	_confidentiality	2.96×
_legitimate	2.84×	_protected	2.86×
_unauthorized	2.62×	_compromise	2.83×
_integrity	2.62×	_dereg	2.56×
_leaks	2.51×	_ambigu	2.36×
_breaches	2.35×	_compliance	2.34×
_intercepted	2.27×	_manipulation	2.32×
_authenticity	2.14×	_inconsistency	2.23×
_disruption	1.97×	_threats	2.11×
_rejection	1.85×	_degradation	1.95×
_securely	1.77×	_vulnerability	1.77×
_lack	1.67×	_ambiguous	1.70×
_safety	1.63×	_authenticated	1.66×
_failures	1.62×	_robust	1.61×
_attacks	1.56×	_interoper	1.54×
_compromising	1.48×	_inability	1.52×
_authorized	1.46×	_malicious	1.47×
_failed	1.40×	_intended	1.46×
_unable	1.32×	_Privacy	1.34×
_incorrect	1.27×	_consistency	1.23×
_availability	1.21×	_compromised	1.22×
_authenticate	1.15×	_confusion	1.14×
_secure	1.11×	_authentication	1.14×
_security	1.07×	_consistent	1.05×
_predictable	0.96×	_unexpected	0.97×
_attacker	0.91×	_certificate	0.91×
_inconsistencies	0.86×	_disabled	0.88×
_authorization	0.81×	_error	0.81×
_vulnerabilities	0.81×	_inconsistent	0.80×
_vulnerable	0.72×	_errors	0.71×
_risks	0.64×	_comply	0.54×
_unclear	0.53×	_reliable	0.52×
_barred	0.45×	_compatibility	0.43×
_flexibility	0.42×	_negative	0.41×
_damage	0.40×	_difficulties	0.20×

Table 12: The raw data of human study. Each column corresponds to one human annotator. Each row corresponds to one sample and we cluster the samples based on LLM-as-a-Judge's decisions for readability.

Idx	LLM-as-a-Judge	1	2	3	4	5	6	7	8
1	Accept	Accept	R→ A	Reject	R→ A	Accept	Accept	R→ A	R→ A
4	Accept	Accept	Accept	Accept	Accept	R→ A	Accept	Accept	R→ A
5	Accept	Accept	Accept	Accept	Accept	Accept	Accept	Accept	Accept
7	Accept	Accept	Accept	Accept	Accept	Accept	Accept	Accept	Accept
9	Accept	Accept	Accept	Accept	Accept	Accept	Accept	Accept	Accept
11	Accept	Accept	R→ A	Reject	Accept	Accept	Accept	Accept	Accept
15	Accept	Accept	Accept	Accept	Accept	Accept	Accept	Accept	R→ A
16	Accept	Accept	Accept	Accept	Accept	Accept	Accept	Accept	Accept
18	Accept	Accept	Accept	Accept	Accept	Accept	R→ A	Accept	Accept
22	Accept	Accept	Accept	Accept	Accept	Accept	Accept	Accept	Accept
24	Accept	Accept	Accept	Accept	Accept	R→ A	Accept	Accept	Reject
25	Accept	Accept	Accept	Accept	Accept	Accept	Accept	Accept	Accept
2	Reject	Reject	Reject	Reject	Reject	Reject	Reject	Reject	Reject
3	Reject	Reject	Reject	Reject	Reject	Reject	Reject	Reject	A→ R
6	Reject	Reject	Reject	Reject	Reject	Reject	Reject	Reject	Reject
8	Reject	Reject	Reject	Reject	Reject	Reject	Reject	Reject	Reject
10	Reject	A→ R	Reject	Reject	Reject	A→ R	A→ R	A→ R	Reject
12	Reject	A→ R	Reject	Reject	Reject	Reject	Reject	Reject	A→ R
13	Reject	A→ R	Reject	Reject	Reject	Reject	Reject	Reject	Reject
14	Reject	Reject	Accept	Accept	Reject	Reject	Reject	Reject	Reject
17	Reject	Reject	Accept	Reject	Reject	Reject	A→ R	Reject	A→ R
19	Reject	Reject	Reject	Accept	Reject	A→ R	A→ R	A→ R	A→ R
20	Reject	Reject	Reject	Reject	Reject	Reject	Reject	Reject	Reject
21	Reject	Reject	Reject	Reject	Reject	Reject	Reject	Reject	A→ R
23	Reject	A→ R	Accept	A→ R	Reject	A→ R	A→ R	Reject	Reject

G Prompts Used and Example Artifacts

Prompt 1: Discover Weakness

You are a cellular network protocol expert. Given a segment of the 3GPP specifications, you should envision what bad things may happen when following the statements, and analyze its potential design weakness. Then, you prepare a change request, which should include:
1. REASON FOR CHANGE: Explain why the identified flaws need to be addressed.
2. SUMMARY OF CHANGE: Provide a summary of the necessary changes to the specifications.
3. CONSEQUENCES IF NOT REVISED: Describe potential negative impacts if the proposed changes are not made.

You should avoid missing important statements and try your best to return detailed responses rich in reasoning.

Prompt 2: Rationale Augmentation

You will be given a task instance composed of TASK INSTRUCTION, USER QUERY, and ASSISTANT RESPONSE. Your task is to revise the ASSISTANT RESPONSE by adding reasoning contents to it. The reasoning contents should explain how the response was generated and act as chain of thoughts for reaching the responses.
Note that
- The task will be related to network protocols, and you should leverage your knowledge in this domain.
- The revised response should be coherent with the original response.
- The revised response should perfectly fit the TASK INSTRUCTION and USER QUERY.
- The revised response should be informative and helpful to the user.
- The revised response should be rich in thoughts and smooth in logic.
- The revised response should be fruitful in educating other assistants.
- You should not alter the original response format.
- You should only return the revised response, which can directly replace the original response.

TASK INSTRUCTION
{}
USER QUERY
{}
ASSISTANT RESPONSE
{}

Prompt 3: Evaluating Security Relevance of CR

You will be given a reasoning segment concerning analyzing problems of cellular network protocol. You should determine whether the implied problem is high-risk, meaning that it strongly relates to security, user privacy, attacks, or any threats to normal service. You should meticulously analyze the given task instance and end up with a judgment. If the problem discussed by the instance is high-risk, you should finally respond with 'High-Risk'; otherwise, respond with 'Low-Risk'.

>>> REASON FOR CHANGE
{}
>>> CONSEQUENCES IF NOT REVISED
{}

Prompt 4: Distinguishing Educational Task Instances

You will be given a task instance concerning analyzing problems of cellular network protocol. The sample is composed of three parts, SYSTEM INSTRUCTION, USER QUERY, and ASSISTANT RESPONSE. You should determine its educational value for a student whose goal is to learn reasoning about the potential flaws in cellular network protocols. You should consider the following aspects:

1. The ASSISTANT RESPONSE is self-contained, meaning that it does not depend on other external information too much and is easy to understand without additional context.
2. The USER QUERY is meaningful and is a suitable analytic target under the SYSTEM INSTRUCTION.
3. The ASSISTANT RESPONSE is logically coherent and conforms well to the requirements of the SYSTEM INSTRUCTION.
4. The ASSISTANT RESPONSE is informative and provides a detailed analysis of the USER QUERY.

You should analyze the given task instance and end up with a judgement. DO NOT make your response too verbose. If it is educational, meaning that it is helpful for a student to learn reasoning about the potential flaws in cellular network protocols, you should finally respond with 'Educational'; otherwise, respond with 'Non-educational'.

TASK INSTRUCTION
{}
USER QUERY
{}
ASSISTANT RESPONSE
{}

Prompt 5: Evaluating Discover Weakness

You are given two reports concerning a weakness analysis of the 3GPP protocol: the Reference Report (which should considered **acceptable** in quality of understanding the protocol problems) and the Drafted Report. Both reports aim to explain the weaknesses and reasons for certain revisions. Your task is to evaluate the claim the Drafted Report identifies the hidden problems in the protocol, compared to the Reference Report.

Conclude your evaluation with a judgment score (s) from:

* **-2 (Strongly Disagree):** The Drafted Report contains significant misunderstandings, misrepresents the issues, or omits critical weaknesses.
* **-1 (Weakly Disagree):** The Drafted Report shows a partial understanding but includes notable errors or omissions in identifying the weaknesses.
* **0 (Neutral):** The Drafted Report identifies the main weaknesses correctly but lacks depth or accuracy in some areas.
* **1 (Weakly Agree):** The Drafted Report largely understands the weaknesses but has minor discrepancies.
* **2 (Strongly Agree):** The Drafted Report demonstrates a near-perfect understanding of the potential problem, with only trivial deviations from the Reference Report.

Note that:

* The Reference Report provides a basic understanding of the protocol problems. The Reference Report is not perfect. So the Drafted Report does not need to match exactly with the Reference Report.
* You should focus on the protocol problems only. Ignore information unrelated to protocol problems in Reference Report, *e.g.* reference to other documents.
* Superficial reports, those with speculative analysis, and those that lack focus should be rated lower. In contrast, reports that are decisive, informative, and facilitate further investigation by human experts are preferred.
* Focus on the content and the understanding of the protocol issues, not on the presentation or formatting.

Do not do anything else other than scoring. Only the final score (x) should be returned in the form of 's: x'.

Reference Report
{}
Drafted Report
{}

Prompt 6: Weakness-to-Attack Verification

You are an expert in verifying the correctness of a vulnerability analysis. You must be extremely rigorous and thorough in your verification process. Analyze and verify whether a given vulnerability analysis (V) enables an attacker, operating within a specified threat model (T), to execute the described attack (A). In other words, whether V + T -> A. Your verification must be comprehensive and consider:
1. Logical completeness - Are all necessary steps and conditions accounted for?
2. Technical accuracy - Are the technical details precise and correct?
3. Exploitability - Can the vulnerability be exploited within the constraints of the threat model?
4. Attack feasibility - Does the analysis conclusively demonstrate the attack's viability?

Note that both T and A are trustworthy and you should evaluate the vulnerability analysis based on them.

1. Vulnerability Analysis (V): Identified weaknesses from the specification
2. Threat Model (T): Attacker capabilities and assumptions
3. Attack Description (A): Attack procedure and implications

ANALYSIS SECTIONS
———————-
Vulnerability Analysis
{}

Threat Model
{}

Attack Description
{}

OUTPUT FORMAT
————————
Please provide your detailed analysis in the following XML format:
<result>
<correctness_reason>
Provide a thorough analysis addressing:
1. Logical completeness of the vulnerability analysis
2. Technical accuracy of all claims and assumptions
3. Whether the vulnerability can be exploited given the threat model constraints
4. Clear demonstration of attack feasibility
5. Any gaps or inconsistencies found

Support your conclusion with specific references to the input components.
</correctness_reason>
<correctness>true/false</correctness>
</result>

Note: Default to false if there is ANY uncertainty or gaps in the logical chain from vulnerability to attack.

Example 1: Discover Weakness Task in CR-EVAL (Index: 29, CR: S3-190376, Design Flaw), Part: 1/2

Task Instruction

You are a cellular network protocol expert. Given a segment of the 3GPP specifications, you should envision what bad things may happen when following the statements, and analyze its potential design weakness. Then, you prepare a change request, which should include:
1. REASON FOR CHANGE: Explain why the identified flaws need to be addressed.
2. SUMMARY OF CHANGE: Provide a summary of the necessary changes to the specifications.
3. CONSEQUENCES IF NOT REVISED: Describe potential negative impacts if the proposed changes are not made.

You should avoid missing important statements and try your best to return detailed responses rich in reasoning.

Test Case

>>> Original Specification Statements:

6.3.3 Authentication and key agreement
The purpose of this procedure is to authenticate the user and establish a new pair of cipher and integrity keys between the VLR/SGSN and the USIM. During the authentication, the USIM verifies the freshness of the authentication vector that is used.

Figure 8: Successful UMTS Authentication and Key Agreement
The VLR/SGSN invokes the procedure by selecting the next unused authentication vector from the ordered array of authentication vectors in the VLR/SGSN database. Authentication vectors in a particular node are used on a first-in / first-out basis. The VLR/SGSN sends to the USIM the random challenge RAND and an authentication token for network authentication AUTN from the selected authentication vector.
Upon receipt the user proceeds as shown in Figure 9.

Figure 9: User authentication function in the USIM
Upon receipt of RAND and AUTN the USIM first computes the anonymity key AK = f5K (RAND) and retrieves the sequence number SQN = (SQN AK) AK.
Next the USIM computes XMAC = f1K (SQN || RAND || AMF) and compares this with MAC which is included in AUTN. If they are different, the user sends an authentication failure message back to the VLR/SGSN with an indication of the cause and the user abandons the procedure. In this case, VLR/SGSN shall initiate an Authentication Failure Report procedure towards the HLR as specified in section 6.3.6. VLR/SGSN may also decide to initiate a new identification and authentication procedure towards the user, cf. TS 24.008 [35].
Next the USIM verifies that the received sequence number SQN is in the correct range.
If the USIM considers the sequence number to be not in the correct range, it sends synchronisation failure back to the VLR/SGSN including an appropriate parameter, and abandons the procedure.
The synchronisation failure message contains the parameter AUTS. It is AUTS = Conc(SQNMS) || MACS. Conc(SQNMS) = SQNMS f5*K(RAND) is the concealed value of the counter SQNMS in the MS, and MACS = f1*K(SQNMS || RAND || AMF) where RAND is the random value received in the current user authentication request. f1* is a message authentication code (MAC) function with the property that no valuable information can be inferred from the function values of f1* about those of f1, ... , f5, f5* and vice versa. f5* is the key generating function used to compute AK in re-synchronisation procedures with the property that no valuable information can be inferred from the function values of f5* about those of f1, f1*, f2, ... , f5 and vice versa.

The AMF used to calculate MACS assumes a dummy value of all zeros so that it does not need to be transmitted in the clear in the re-synch message. The construction of the parameter AUTS in shown in the following Figure 10:

Figure 10: Construction of the parameter AUTS
If the sequence number is considered to be in the correct range however, the USIM computes RES = f2K (RAND) and includes this parameter in a user authentication response back to the VLR/SGSN. Finally the USIM computes the cipher key CK = f3K (RAND) and the integrity key IK = f4K (RAND). Note that if this is more efficient, RES, CK and IK could also be computed earlier at any time after receiving RAND. If the USIM also supports conversion function c3, it shall derive the 64-bit GSM cipher key Kc from the UMTS cipher/integrity keys CK and IK. UMTS keys are sent to the MS along with the derived 64-bit GSM key for UMTS-GSM interoperability purposes. USIM shall store original CK, IK until the next successful execution of AKA.
Upon receipt of user authentication response the VLR/SGSN compares RES with the expected response XRES from the selected authentication vector. If XRES equals RES then the authentication of the user has passed. The SGSN shall compute the 128-bit GSM ciphering key Kc128 according to annex B.5 if it is to use a 128-bit GSM ciphering algorithm. The VLR/MSC shall compute the 128-bit GSM ciphering key Kc128 according to annex B.5 if it signals a 128-bit GSM ciphering algorithm as a permitted GSM ciphering algorithm to the BSS. The VLR/SGSN also selects the appropriate cipher key CK and integrity key IK from the selected authentication vector. If XRES and RES are different, VLR/SGSN shall initiate an Authentication Failure Report procedure towards the HLR as specified in section 6.3.6. VLR/SGSN may also decide to initiate a new identification and authentication procedure towards the user, cf. TS 24.008 [35].

To be continued in the next page ↘

Example 2: Discover Weakness Task Instance in CR-EVAL (Index: 29, CR: S3-190376, Design Flaw), Part: 2/2

↘ Continued

Re-use and re-transmission of (RAND, AUTN)
The verification of the SQN by the USIM will cause the MS to reject an attempt by the VLR/SGSN to re-use a quintet to establish a particular UMTS security context more than once. In general therefore the VLR/SGSN shall use a quintet only once.
There is one exception however: in the event that the VLR/SGSN has sent out an authentication request using a particular quintet and does not receive a response message (authentication response or authentication failure) from the MS, it may re-transmit the authentication request using the same quintet. However, as soon as a response message arrives no further re-transmissions are allowed. If after the initial transmission or after a series of re-transmissions no response arrives, retransmissions may be abandoned. If retransmissions are abandoned then the VLR/SGSN shall delete the quintet. At the MS side, in order to allow this re-transmission without causing additional re-synchronisation procedures, the ME shall store for the PS domain (and optionally the CS domain) the last received RAND as well as the corresponding RES, CK and IK. If the USIM returned SRES and Kc (for GSM access), the ME shall store these values. When the ME receives an authentication request and discovers that a RAND is repeated, it shall re-transmit the response. The ME shall delete the stored values RAND, RES and SRES (if they exist) as soon as the 3G security mode command or the GSM cipher mode command is received by the ME or the connection is aborted. If the ME can handle the retransmission mechanism for CS domain then it shall be able to handle the retransmission for both PS and CS domain simultaneously.

6.3.5 Re-synchronisation procedure
A VLR/SGSN may send two types of authentication data requests to the HE/AuC, the (regular) one described in subsection 6.3.2 and one used in case of synchronisation failures, described in this subsection.
Upon receiving a synchronisation failure message from the user, the VLR/SGSN sends an authentication data request with a "synchronisation failure indication" to the HE/AuC, together with the parameters:
- RAND sent to the MS in the preceding user authentication request, and
- AUTS received by the VLR/SGSN in the response to that request, as described in subsection 6.3.3.

An VLR/SGSN will not react to unsolicited "synchronisation failure indication" messages from the MS.
The VLR/SGSN does not send new user authentication requests to the user before having received the response to its authentication data request from the HE/AuC (or before it is timed out).
When the HE/AuC receives an authentication data request with a "synchronisation failure indication" it acts as follows:
1. The HE/AuC retrieves SQNMS from Conc(SQNMS) by computing Conc(SQNMS) f5*K(RAND).
2. The HE/AuC checks if SQNHE is in the correct range, i.e. if the next sequence number generated SQNHE using would be accepted by the USIM.
3. If SQNHE is in the correct range then the HE/AuC continues with step (6), otherwise it continues with step (4).
4. The HE/AuC verifies AUTS (cf. subsection 6.3.3).
5. If the verification is successful the HE/AuC resets the value of the counter SQNHE to SQNMS.
6. The HE/AuC sends an authentication data response with a new batch of authentication vectors to the VLR/SGSN. If the counter SQNHE was not reset then these authentication vectors can be taken from storage, otherwise they are newly generated after resetting SQNHE. In order to reduce the real-time computation burden on the HE/AuC, the HE/AuC may also send only a single authentication vector in the latter case.

Whenever the VLR/SGSN receives a new batch of authentication vectors from the HE/AuC in an authentication data response to an authentication data request with synchronisation failure indication it deletes the old ones for that user in the VLR/SGSN.
The user may now be authenticated based on a new authentication vector from the HE/AuC. Figure 12 shows how re-synchronisation is achieved by combining a user authentication request answered by a synchronisation failure message (as described in section 6.3.3) with an authentication data request with synchronisation failure indication answered by an authentication data response (as described in this section).

Figure 12: Resynchronisation mechanism

Reference Answer

>>> REASON FOR CHANGE

The calculation of the keystream for concealing SQNMS in the case of re-sychronisation only has RAND as a non-key input and hence the same keystream may be used to protect different SQNMSs if an attacker replays an authentication challenge. This leaks at least the xor between the different SQNMS values and it has been shown how this leak can be used to derive the LSBs of SQNMS.

>>> SUMMARY OF CHANGE

Add MAC-S as an input parameter to claculate the keystream at re-synchronisation in order to get a differeent keystream for each SQNMS value.

>>> CONSEQUENCES IF NOT REVISED

Keystream is repeated which should be against security design properties.

Example 3: Discover Weakness Task Instance in CR-EVAL (Index: 145, CR: C1-193185, Under-specification)

Task Instruction

For display brevity, the repetitive instruction is omitted here.

Test Case

>>> Original Specification Statements:
5.4.1.2.4.2 EAP message reliable transport procedure initiation by the network
In order to initiate the EAP message reliable transport procedure, the AMF shall create an AUTHENTICATION REQUEST message.
The AMF shall set the EAP message IE of the AUTHENTICATION REQUEST message to the EAP-request message to be sent to the UE. The AMF shall set the ngKSI IE of the AUTHENTICATION REQUEST message to the ngKSI value selected in subclause 5.4.1.2.2.2 or subclause 5.4.1.2.3.1. In this release of specification, the AMF shall set the ABBA IE of the AUTHENTICATION REQUEST message with the length of ABBA IE to 2 and the ABBA contents to be 2 octets in length with value 0000H as described in subclause 9.11.3.10.
The AMF shall send the AUTHENTICATION REQUEST message to the UE, and the AMF shall start timer T3560 (see example in figure 5.4.1.2.4.2.1).
Figure 5.4.1.2.4.2.1: EAP message reliable transport procedure Upon receipt of an AUTHENTICATION REQUEST message with the EAP message IE, the UE handles the EAP message received in the EAP message IE and the ABBA of the AUTHENTICATION REQUEST message.

5.4.1.3.2 Authentication initiation by the network
The network may initiate a 5G AKA based primary authentication and key agreement procedure for a UE in 5GMM-CONNECTED mode at any time. For restrictions applicable after handover or inter-system change to N1 mode in 5GMM-CONNECTED mode, see subclause 5.4.1.2.3.
The network initiates the 5G AKA based primary authentication and key agreement procedure by sending an AUTHENTICATION REQUEST message to the UE and starting the timer T3560 (see example in figure 5.4.1.3.2.1). The AUTHENTICATION REQUEST message shall contain the parameters necessary to calculate the authentication response (see 3GPP TS 33.501 [24]). This message shall include the ngKSI that will be used by the UE and AMF to identify the KAMF and the partial native security context that is created if the authentication is successful. This message shall also include the ABBA parameter. In this release of specification, the network shall set the length of ABBA IE to 2 and the ABBA contents to be 2 octets in length with value 0000H as described in subclause 9.11.3.10.
If an ngKSI is contained in an initial NAS message during a 5GMM procedure, the network shall include a different ngKSI value in the AUTHENTICATION REQUEST message when it initiates a 5G AKA based primary authentication and key agreement procedure.
Figure 5.4.1.3.2.1: 5G AKA based primary authentication and key agreement procedure

9.11.3.10 ABBA
The purpose of the ABBA information element is to enable the bidding down protection of security features.
The ABBA information element is coded as shown in figure 9.11.3.10.1 and table 9.11.3.10.1.
The ABBA is a type 4 information element with a minimum length of 4 octets.
8 7 6 5 4 3 2 1
ABBA IEI octet 1
Length of ABBA contents octet 2
ABBA contents octet 3 octet n
Figure 9.11.3.10.1: ABBA information element
Table 9.11.3.10.1: ABBA information element
ABBA contents (octet 3-n): indicate set of security features defined for 5GS as described in 3GPP TS 33.501 [24].

Reference Answer

>>> REASON FOR CHANGE
CT1 sent an LS to SA3 (see C1-191686) asking about the UE behavior when an ABBA parameter with a non-zero value, or with a length that is more than 2 octets, is received in Release 15. This document aims to specify the UE behavior for this case following the LS response from SA3.

>>> SUMMARY OF CHANGE
Specify that in case the UE receives an ABBA parameter with - a length of more than 2 octets, - a non-zero value the UE shall use the ABBA as received from the network. Interoperability analysis1) UE compliant with the previous specification version with an AMF that is compliant with this CRIf the UE gets an ABBA parameter with a value that is different from 0000H, the UE will send a 5GMM STATUS message. However, the AMF already handles a 5GMM STATUS message.2) UE compliant with this CR with an AMF that is compliant with the previous specification version the UE gets an ABBA parameter with a value that is different from 0000H, the UE will use the ABBA parameter as it is received. If for some reason the KAMF at the UE and the network is not the same, the integrity check fails during the security mode procedure. However, handling integrity check failures already exists and is not introduced by this CR. If the KAMF at the UE and the network is the same, then no errors will occur. This CR is backwards compatible.

>>> CONSEQUENCES IF NOT REVISED
The UE uses an ABBA that is different from what the network has used leading to security failure.

Table 13: Meta-information and expert rationales for the example Change Request (C1-172658[9])

24.301	2871	2	**Current version**	14.3.0
Title	Correction of Handling of MO Detach without Integrity Protection		**Date**	2017-05-19
Category	F (Correction)		**Release**	Rel-14
Reason for change	In Rel-8, when CT1 specified the rules for the handling of NAS messages that are not intergrity protected or fail the integrity check by the receiver, CT1 decided that a mobile originated Detach Request without integrity protection was to be treated by the MME, because it is one of the messages which "in certain situations ... are sent by the UE before security can be activated". An additional justification was at that time that it did not appear very likely that someone would take the efforts to listen in on the NAS signalling in a cell and operate a manipulated UE just for the purpose of detaching other subscribers. Moreover, in the worst case this kind of DoS attack, which would prevent the UE from receiving MT services, would be detected and alleviated when the UE performed the next normal or periodic TAU (or RAU) or when the UE requested some MO service. Since then, the situation has changed, because *e.g.* for UEs used for MTC/ CIoT, it may take longer to detect and repair the issue, as - periodic update timer values up to 14 days can be negotiated between UE and network, and - some of these devices send MO user data with a frequency of once every few weeks, but on the other hand the UEs (*e.g.* certain metering devices) may be required to stay attached in order to be reachable for the application server. Additionally, due to the higher density of devices per cell (and higher number per MME), it has become easier to perform the attack successfully even by picking the S-TMSI values at random. As there are not so many cases where a UE might rightfully send a Detach Request without integrity protection, we suggest to modify the requirements for the MME: the MME should authenticate the subscriber if possible. If the authentication is not performed, *e.g.* because the detach is due to "switch-off", or for any other reason, the MME may ignore the Detach Request and remain in state EMM-REGISTERED. For this case the MME can attempt to apply additional criteria before marking the subscriber as deregistered, *e.g.* the MME may wait whether the UE is still performing periodic updating or whether it is still responding to paging when an MT user data packet arrives. (We found the following cases where a UE might rightfully send a Detach Request without integrity protection: 1) the UE is attached for emergency bearer services and there is no shared EPS security context available, *e.g.* due to lack of roaming agreement; 2) due to user interaction an attach procedure is cancelled before the secure exchange of NAS messages has been established; 3) a NAS COUNT wrap around occurred so that the current EPS security context can no longer be used. In principle it should be possible for the MME to determine whether any of these cases can apply when a Detach Request message failing the integrity check is received.)				
Summary of change	Rules for the handling of a DETACH REQUEST message failing the integrity check are modified for the case when a current EPS security context exists and the secure exchange of NAS messages has not yet been established: - If it is not a detach request due to switch off, and the MME can initiate an authentication procedure, the MME should authenticate the subscriber before processing the detach request any further. - If it is a detach request due to switch off, or the MME does not initiate an authentication procedure for any other reason, the MME may ignore the detach request and remain in state EMM-REGISTERED. (The network can attempt to use additional criteria before marking the UE as EMM-DEREGISTERED.)			
Consequences if not approved	Risk of a DoS attack. UEs using an extended periodic update timer can become unreachable for paging for a long time, if Detach Request without integrity protection is always accepted when secure exchange of NAS messages has not yet been established.			

[9] https://www.3gpp.org/ftp/tsg_ct/WG1_mm-cc-sm_ex-CN1/TSGC1_104_Zhangjiajie/Docs/C1-172658.zip

Table 14: Specification revisions for the example Change Request (C1-172658)

	4.4.4.3 Integrity checking of NAS signalling messages in the MME
	Except the messages listed below, no NAS signalling messages shall be processed by the receiving EMM entity in the MME or forwarded to the ESM entity, unless the secure exchange of NAS messages has been established for the NAS signalling connection:
	- EMM messages:
	- ATTACH REQUEST;
	- IDENTITY RESPONSE (if requested identification parameter is IMSI);
	- AUTHENTICATION RESPONSE;
	- AUTHENTICATION FAILURE;
	- SECURITY MODE REJECT;
	- DETACH REQUEST;
	The remaining unchanged clauses are omitted for brevity.
[+]	NOTE 2: The DETACH REQUEST message can be sent by the UE without integrity protection, *e.g.* if the UE is attached for emergency bearer services and there is no shared EPS security context available, or if due to user interaction an attach procedure is cancelled before the secure exchange of NAS messages has been established. For these cases the network can attempt to use additional criteria (*e.g.* whether the UE is subsequently still performing periodic tracking area updating or still responding to paging) before marking the UE as EMM-DEREGISTERED.
	All ESM messages are integrity protected except a PDN CONNECTIVITY REQUEST message if it is sent piggybacked in ATTACH REQUEST message and NAS security is not activated.
	Once a current EPS security context exists, until the secure exchange of NAS messages has been established for the NAS signalling connection, the receiving EMM entity in the MME shall process the following NAS signalling messages, even if the MAC included in the message fails the integrity check or cannot be verified, as the EPS security context is not available in the network:
	- ATTACH REQUEST;
	- IDENTITY RESPONSE (if requested identification parameter is IMSI);
	- AUTHENTICATION RESPONSE;
	- AUTHENTICATION FAILURE;
	- SECURITY MODE REJECT;
[-]	- DETACH REQUEST (if sent before security has been activated);
[+]	- DETACH REQUEST;
	- DETACH ACCEPT;
	- TRACKING AREA UPDATE REQUEST;
	- SERVICE REQUEST;
	- EXTENDED SERVICE REQUEST;
	- CONTROL PLANE SERVICE REQUEST.
	NOTE 3: These messages are processed by the MME even when the MAC fails the integrity check or cannot be verified, as in certain situations they can be sent by the UE protected with an EPS security context that is no longer available in the network.
	If an ATTACH REQUEST message fails the integrity check and it is not an attach request for emergency bearer services, the MME shall authenticate the subscriber before processing the attach request any further. For the case when the attach procedure is for emergency bearer services see subclause 5.5.1.2.3 and subclause 5.4.2.5.
[+]	If a DETACH REQUEST message fails the integrity check, the MME shall proceed as follows:
[+]	- If it is not a detach request due to switch off, and the MME can initiate an authentication procedure, the MME should authenticate the subscriber before processing the detach request any further.
[+]	- If it is a detach request due to switch off, or the MME does not initiate an authentication procedure for any other reason, the MME may ignore the detach request and remain in state EMM-REGISTERED.
	NOTE 4: The network can attempt to use additional criteria (*e.g.* whether the UE is subsequently still performing periodic tracking area updating or still responding to paging) before marking the UE as EMM-DEREGISTERED.
[+]	The remaining unchanged clauses are omitted for brevity.

Table 15: Reported attacks in cellular networks.

Attack Effects	Related Works
IMSI/SUPI cracking	Hussain et al. (2019a)
Traffic decryption	Rupprecht et al. (2020a)
User tracking	Kohls et al. (2019); Hong et al. (2018); Kotuliak et al. (2022)
User presence identification	Shaik et al. (2016); Hussain et al. (2019a); Ludant et al. (2023); Erni et al. (2022)
Device fingerprinting	Shaik et al. (2019); Kotuliak et al. (2022); Park et al. (2022)
Message/Service spoofing	Kim et al. (2019); Lee et al. (2019); Rupprecht et al. (2020b); Park et al. (2022) Rupprecht et al. (2019)
Traffic fingerprinting	Kohls et al. (2019); Bae et al. (2022)
Denial of service	Shaik et al. (2016); Yu and Chen (2019); Kim et al. (2019); Hussain et al. (2019a); Bitsikas and Pöpper (2021); Ludant and Noubir (2021); Erni et al. (2022); Akon et al. (2023); Chen et al. (2024b); Xing et al. (2024); Bennett et al. (2024)
Downgrading to insecure versions	Shaik et al. (2016)
Key re-installation	Raza et al. (2021)
Malicious message injection	Yang et al. (2019); Ludant and Noubir (2021); Erni et al. (2022); Kotuliak et al. (2022)
Eavesdropping data communication	Rupprecht et al. (2016); Kim et al. (2019); Rupprecht et al. (2020a); Park et al. (2022)
Exposing device capabilities	Shaik et al. (2019)
Content Fingerprinting	Kohls et al. (2019); Bae et al. (2022)
Illegitimate access to services	Rupprecht et al. (2020b); Akon et al. (2023)
Unauthorized entry to secrets	Akon et al. (2023)
Phishing legitimate users	Kim et al. (2019)
BTS resource depletion	Kim et al. (2019)
Free data service	Li et al. (2015); Chen et al. (2024b)
Signaling storm	Xing et al. (2024)

H Cellular Specification Weaknesses

Cellular specifications suffer from weaknesses, ranging from minor ambiguities and undefined behaviors to fundamental design flaws. While these weaknesses may go unnoticed under typical conditions, they can become threatening in specific scenarios. A large number of CRs aim to address weaknesses in the specifications, motivating our research in this work. Broadly speaking, these weaknesses can lead to various negative consequences, including performance degradation, interoperability failures, and security vulnerabilities. In this work, we focus primarily on those weaknesses that pose security risks, among which the severest ones may be exploited by malicious entities to disrupt normal service operations. While we focus on specification-level weaknesses, their implications are far-reaching. Design flaws within specifications lead to vulnerabilities in compliant implementations and thus propagate through the whole cellular network system. Issues like underspecification lead to implementations and configurations that fail to meet essential requirements. However, it would be unfair to place blame solely on specification drafters, particularly when observing the immense volume and complexity of cellular specifications. Current refinement practices depend on human experts to identify weaknesses and propose CRs, a labor-intensive approach that lacks a systematic evaluation framework. These challenges highlight the critical need for automated tools capable of refining cellular specifications.

We provide a comprehensive survey of common specification weaknesses (Table 16) reported by previous academic works and associated attack vectors (Table 15) in cellular networks. Our survey reveals that numerous attacks against cellular networks have been proposed by exploiting unsafe designs and ambiguous drafts. This demonstrates that specification weaknesses, if exploitable, can make significant impacts on cellular networks. However, it would be unfair to place blame solely on specification drafters, given the immense volume and complexity of the cellular network specification system. Rather, it underscores the importance of systematic weakness analysis and motivates automatic tools that help refine cellular specifications.

Table 16: Common issues in cellular specifications.

Specification weaknesses	Related works
Design Flaws	Tu et al. (2014); Shaik et al. (2016); Yu and Chen (2019); Shaik et al. (2019); Kim et al. (2019); Lee et al. (2019); Hussain et al. (2019a); Kotuliak et al. (2022); Ludant and Noubir (2021); Chen et al. (2021b); Bitsikas and Pöpper (2021); Bae et al. (2022); Borgaonkar et al. (2018)
Underspecification	Shaik et al. (2016); Hong et al. (2018); Basin et al. (2018); Rupprecht et al. (2020a); Park et al. (2022); Akon et al. (2023); Xing et al. (2024)
Undefined Behaviors	Park et al. (2022); Klischies et al. (2023)
Inconsistencies	Park et al. (2022); Chen et al. (2024b); Rahman et al. (2024)

Table 17: Evaluation of known attacks using **CRITIC**-LLaMA-3.1-8B (C). Hermes results (H) are self-reported in Al Ishtiaq et al. (2024). We indicate the version where the flawed specification was identified. We use symbols Ⓘ (implementation flaw), Ⓒ (configuration flaw), and Ⓝ (non-deterministic).

ID	Attack	Protocol	H	C
1	AUTH REJECT Attack (Yu and Chen, 2019)	4G NAS (15.0.0)	✓	✓
2	Blind DoS Attack (Kim et al., 2019)	4G RRC (14.2.2)	✗	✓
3	Cutting off the Device (Hussain et al., 2019b)	5G NAS (16.2.0)	✗	✓
4	Deletion of allowed CAG list (Al Ishtiaq et al., 2024)	5G NAS (17.8.0)	✓	✓
5	DoS with RRCSetupRequest attack (Hussain et al., 2019b)	5G RRC (15.5.1)	✗	✓
6	Denying all network services (Shaik et al., 2016)	4G NAS (12.8.0)	✓	✓
7	Denying selected service (Shaik et al., 2016)	4G NAS (12.8.0)	✗	✓
8	DETACH REQUEST attack (Hussain et al., 2018)	4G NAS (12.8.0)	✓	✓
9	Downgrade to non-LTE services (Shaik et al., 2016)	4G NAS (12.8.0)	✓	✓
10	Downgrade via ATTACH REJECT (Shaik et al., 2016)	4G NAS (12.8.0)	✓	✓
11	Energy Depletion with RRCSETUP (Al Ishtiaq et al., 2024)	5G RRC (17.0.0)	✓	✓
12	Exposing NAS Sequence Number (Hussain et al., 2019b)	5G NAS (16.0.2)	✓	✓
13	Exposure of SQN (Borgaonkar et al., 2018)	3G AKA (15.0.0)	✓	✓
14	IMSI Catching (Van Den Broek et al., 2015)	4G NAS (12.7.0)	✓	✓
15	IMSI Cracking (Hussain et al., 2019a)	4G RRC (15.0.0)	✗	✓
16	IMSI Cracking (Hussain et al., 2019a)	5G NAS (15.0.0)	✗	✓
17	Incarceration with RRCRELEASE (Hussain et al., 2019b)	5G RRC (15.5.1)	✓	✓
18	Installing Null Cipher/Integrity (Hussain et al., 2019b)	5G RRC (15.5.1)	✓	✓
19	Lullaby Attack (Hussain et al., 2019b)	5G RRC (15.5.1)	✓	✓
20	Measurement report (Shaik et al., 2016)	4G RRC (12.3.0)	✗	✓
21	NAS COUNT update attack (Al Ishtiaq et al., 2024)	5G NAS (16.4.0)	✓	✓
22	NAS Counter Reset (Hussain et al., 2019b)	5G NAS (16.0.2)	✓	✓
23	Neutralizing TMSI Refreshment (Hussain et al., 2019b)	5G NAS (16.2.0)	✗	✓
24	Paging channel hijacking (Hussain et al., 2018)	4G RRC (12.5.0)	✗	✓
25	SERVICE REJECT attack (Shaik et al., 2016)	4G NAS (12.8.0)	✓	✓
26	Signaling DoS Attack (Bassil et al., 2013)	4G NAS (16.8.0)	✓	✓
27	SUCI Catching Vulnerability (Chlosta et al., 2021)	5G NAS (15.0.0)	✗	✓
28	Synchronization Failure Attack (Yu and Chen, 2019)	4G NAS (15.0.0)	✗	✓
29	Uplink NAS Counter Desync (Hussain et al., 2019b)	5G NAS (16.0.2)	✓	✓
30	5G AKA DoS Attack (Cao et al., 2020)	5G NAS (15.2.0)	✓	✓
31	AKA Bypass (Kim et al., 2019)	5G RRC (Ⓘ)	✗	-
32	EMM Information Vulnerability (Park et al., 2016)	4G NAS (Ⓘ)	✓	-
33	Impersonation attack (Chlosta et al., 2019)	4G NAS (Ⓒ)	✗	-
34	Malformed Identity Request (Michau and Devine, 2016)	4G NAS (Ⓘ)	✗	-
35	RLF report (Shaik et al., 2016)	5G RRC (Ⓘ)	✓	-
36	S-TMSI Catching (Kim et al., 2019)	4G NAS (Ⓝ)	✓	-