The Impact of Context Window Constraints on ReAct Agents in Cryptographic CTF Challenges

Performance, Efficiency, and Failure Modes of ReAct Agents under Context Constraints

Yusuf Baris Kose[1]

Supervisor(s): Dr. Zeki Erkin[1]

[1]EEMCS, Delft University of Technology, The Netherlands

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
January 25, 2026

Name of the student: Yusuf Kose
Final project course: CSE3000 Research Project
Thesis committee: Dr. Zeki Erkin, Dr. Mitchell Olsthoorn

Abstract

While Large Language Model (LLM) agents are increasingly capable in specialized domains, the impact of Context Window Constraints on reasoning stability remains under-explored. In this paper we investigate how strictly controlling context size influences agent performance in solving multi-step cryptographic Capture The Flag (CTF) challenges. We adapted an agentic environment where models attempt to solve crypto challenges under four distinct context length constraints (8k–64k), managed dynamically as a tunable hyperparameter. Our results reveal a non-linear performance curve with a clear saturation threshold between 16k and 32k tokens, beyond which additional context offers negligible benefit. We observe a distinct shift in failure modes: tight constraints lead to context starvation (hallucination), while unconstrained windows allow the accumulation of error traces, where the presence of prior failed attempts biases the agent toward repetitive action loops which in turn inflates its operating costs and decreases operational efficiency. These findings demonstrate that unconstrained context is not only expensive, but leads to regression in the reasoning stability within logic-heavy domains. We conclude that future benchmarks must explicitly distinguish between capability failures and context-induced failures. Furthermore, our findings suggest that engineering strategies should prioritize dynamic context management over the reliance on static, maximized windows.

1 Introduction

1.1 Motivation: The "More is Better" Fallacy in Context Windows

The past few years have been defined by fierce competition in the LLM landscape. Amidst other architectural shifts, a primary trend observed to beat the competition has been the aggressive expansion of context windows, now exceeding one million tokens in state-of-the-art (SOTA) models [11]. This rapid scaling has fueled a prevalent industry assumption: that maximizing available context inevitably improves agentic performance. Consequently, the prevailing engineering logic suggests that by feeding an autonomous agent its entire interaction history, code execution logs, and environmental state, we minimize information loss and maximize decision quality.

In this thesis we challenge that assumption. We argue that for complex reasoning tasks such as cryptographic exploitation, "Infinite Context" is not a feature, but a liability. Recent studies contradicts the linear scaling hypothesis. Modarressi et al. (2025), in their *NOLIMA* benchmark, demonstrated that model performance degrades as context grows, particularly for tasks requiring latent reasoning rather than simple literal retrieval. They observed that 11 out of 13 SOTA models suffered a reasoning performance drop of over 50% when the context window reached just 32,000 tokens. This indicates that excessive history does not serve as memory; it acts as noise, actively diluting the model's ability to identify relevant signals.

Consequently, the reliance on massive context windows introduces a "Triple Penalty" for autonomous agents, offering no return on investment:

1. **Performance Degradation:** As evidenced by NOLIMA, pushing context beyond specific thresholds (e.g. 16k) causes the model to lose track of latent associations, effectively lowering its reasoning fidelity.
2. **Latency Bottlenecks:** Processing massive inputs creates significant inference latency (Time To First Token), delaying agent responsiveness in time-sensitive security environments.
3. **Unjustified Cost:** Even with modern context caching - where providers store and reuse frequently accessed prefix data to bypass the expensive prefill phase- reducing the financial impact of static prefixes, the computational cost of attending to hundreds of thousands of tokens remains significant.

Given this penalty, the engineering imperative is to reduce context size. However, simply truncating history risks removing vital information. This creates a critical tension between minimizing the "Triple Penalty" and maintaining agent viability—a trade-off that remains unexplored in cryptographic domains.

1.2 Research Gap

Existing benchmarks in this domain, such as *AICrypto* and *InterCode-CTF*, focus primarily on raw capability within unconstrained environments. They answer whether a model can solve a task, but neglect operational efficiency and stability under strict resource constraints. While context degradation is well-documented in general Natural Language Processing (NLP)—specifically the "Lost-in-the-Middle" phenomenon, where model performance peaks at the start and end of a context window but dips for information located in the center —there is no data on how these constraints manifest in logic-heavy cryptographic tasks. Consequently, the field lacks a defined "safe operating floor"—the minimum context size required for crypto-agents to function before systemic breakdown occurs.

1.3 Research Questions

The central question of this research, therefore, shifts the focus from *capability maximization* to *constraint optimization*. We ask: **How do systematically varying context-window constraints influence the reasoning performance, efficiency and failure modes of ReAct-style agents when solving cryptographic CTF challenges?** By isolating context length as an independent variable, we aim to determine if aggressive truncation acts as a stabilizing function for the ReAct (Reason + Act) loop —filtering out the noise that triggers reasoning degradation—or if it introduces new fragilities [19].

To address this gap, this thesis investigates the impact of context truncation on the solvability of cryptographic CTF challenges. We pose the following research questions:

1. **RQ1:** At what context-window threshold do LLM agents achieve functionally-equivalent performance to unconstrained baselines on multi-step cryptographic challenges?
2. **RQ2:** Does increasing context capacity yield proportional gains in task success, or do agents exhibit a saturation threshold beyond which additional context provides negligible benefit?
3. **RQ3:** How do failure mode distributions (hallucination, cognitive thrashing, timeout) shift as context constraints are relaxed from 8k to 64k?

1.4 Contributions

This research provides an experimental analysis of agentic behavior under resource constraints. Our contributions are listed as;

- **Dynamic Context Evaluation:** To address the lack of granular control in existing benchmarks, we implemented a custom Context Management Layer on top of the AICrypto framework. This experimental module allows for the precise, per-turn truncation of agent memory, enabling a controlled comparison of model performance across various thresholds.
- **Agentic Stability:** We provide a systematic comparison of SOTA models within a ReAct loop. Unlike standard benchmarks that measure static accuracy, our analysis isolates "stability" as a dependent variable, determining the specific context thresholds where reasoning models transition from effective problem-solving to repetitive failure.
- **Agentic Cognitive Thrashing:** We adapt the concept of thrashing"—defined in operating systems by Denning (1968) [4] and applied to human attentional control by Posner—to the domain of LLM agents. While prior literature identifies generic failures like step repetition" (Zhang et al., 2025), we distinguish **Agentic Cognitive Thrashing** as a deterministic failure mode driven by context constraints. In this state, the agent consumes most of the resources managing context without advancing the problem state, resembling the resource-swapping latency defined in the original Operating Systems metaphor.

1.5 Thesis Structure

The remainder of this thesis is organized as follows. Section 2 reviews the relevant background on LLM context limitations and the ReAct paradigm. Section 3 details our experimental methodology and the implementation of the dynamic context middleware. Section 4 presents the results and analysis regarding performance thresholds and failure modes, which are further interpreted in the discussion in Section 5. Finally, Section 6 addresses ethical considerations and sustainability, followed by conclusions and future work in Section 7.

2 Background and Related Work

In this section we review the architectural constraints of LLMs regarding context processing, define the ReAct agentic paradigm used in our experimental setup, and detail the environment our research agents operate within.

2.1 Effective vs. Available Context

While modern LLMs rely on the Transformer architecture [16], the effective processing of long contexts remains a primary architectural limitation [7]. The cost of processing these longer contexts is rooted in the self-attention mechanism, which calculates the relevance of every token in a sequence to every other token to construct semantic representations. Theoretically, the computational complexity and memory footprint of standard self-attention scale quadratically ($O(N^2)$) with the sequence length N [16]. This scaling penalty is incurred primarily during the **prefill phase** [10], where the model must process the entire conversation history before generating a single new token. For an autonomous agent executing recursive loops, this creates a compounding economic penalty: as the context grows, the cost and latency of every subsequent step increase non-linearly. While recent optimizations such as FlashAttention [3] have mitigated raw latency, they do not eliminate the fundamental resource cost of attending to massive sequences.

Crucially, architectural capacity (e.g., 1M tokens) does not equate to reasoning capacity. While Liu et al. [7] originally identified that retrieval accuracy degrades in the middle of long sequences, recent work [8] suggests this degradation is far more severe for the *latent reasoning* required in cryptography. Unlike simple retrieval tasks where a model must extract a literal string, cryptographic challenges require synthesizing associative links across the context window. As demonstrated by the NOLIMA benchmark [8], this reasoning instability often leads to failure at thresholds (e.g., 32k tokens) far lower than the model's theoretical limit, effectively rendering the tail of the context window detrimental to agentic reliability.

2.2 Agentic Reasoning

We define an Agent not merely as a chatbot, but as an LLM augmented with a cyclic feedback loop that permits the execution of external tools to manipulate the environment it is provided.

The ReAct Framework

This study utilizes the ReAct paradigm proposed by Yao et al. [19]. Unlike Act-only policies that blindly execute commands, ReAct enforces an interleaved structure: `Thought` $\rightarrow$ `Action` $\rightarrow$ `Observation`. The model must explicitly articulate its reasoning trace (Thought) before generating a tool call (Action). This verbalized reasoning serves as a chain-of-thought grounding, theoretically reducing hallucination by forcing the model to justify its next step based on the previous `Observation`.

Context as State and Failure Modes

In the absence of external databases, the context window serves as the agent's entire working memory or state [18]. It contains the interaction history, including past code execution errors and file system outputs. Consequently, the agent's ability to "learn" from a failed attempt is entirely dependent on its ability to attend to the specific history of that failure. If this

history is truncated or is suppressed by noise, the agent effectively becomes amnesic, losing the state necessary to correct its course.

Current literature identifies two primary failure modes in ReAct loops: Hallucination and Cyclic Repetition. While hallucination is well-documented, recent analysis highlights "Fragile Execution" loops, where agents under high cognitive load repeatedly invoke the same invalid tool [13]. We map this behavior to **Cognitive Thrashing** (defined in Section 1.4), positing that it is a symptom of context overload where the model loses the attention span required to integrate the Error observation into its next Thought.

2.3 LLMs in Cybersecurity

The evaluation of LLMs in cybersecurity has evolved around Capture The Flag (CTF) competitions—gamified exercises that simulate real-world vulnerabilities.

The Deterministic Execution Gap

Cryptographic challenges differ fundamentally from general coding tasks. They require *symbolic exactness* rather than *semantic approximation*. A creative writing prompt tolerates variance; a decryption script fails if a single bit is incorrect. As noted by recent benchmarking efforts, LLMs often struggle with the latent reasoning required to identify which specific mathematical property (e.g., RSA small exponent attack) applies to a given puzzle, even if they can generate syntactically correct Python code.

Work Related to CTF Benchmarks

Recent years have seen the emergence of several benchmarks tailored to security reasoning. The **NYU CTF Bench** [15] and the **EnIGMA** framework [1] have been pivotal in establishing baselines for general offensive capabilities. These benchmarks aggregate challenges across diverse categories such as reverse engineering, forensics, and binary exploitation, shifting evaluation from simple syntax verification to multi-step state exploration.

This study, however, builds upon the **AICrypto** benchmark [18]. Distinct from general web-exploitation datasets, AICrypto specifically targets the intersection of LLM reasoning and cryptographic complexity. Consisting of 150 static and dynamic challenges across domains like RSA, Lattice-based cryptography, and Elliptic Curve Cryptography (ECC), it provides a rigorous testbed for the specific 'dynamic reasoning' and large-number arithmetic required for modern cryptography—skills where current models struggle significantly despite their general coding proficiency.

2.4 Gap Analysis: Capability vs. Reliability

While the benchmarks discussed above establish baselines for unconstrained performance, recent NLP literature signals a critical fragility in these metrics. Modarressi et al. [8] demonstrate in *NOLIMA* that latent reasoning capabilities degrade non-linearly as context expands, often collapsing well before the model's advertised limit. Similarly, Hosseini et al. [5] provide evidence that long context windows can actively dilute attention in complex analytical tasks.

This suggests that current security benchmarks may be masking a fundamental inverse relationship: that more context is not merely expensive, but functionally detrimental to precise cryptographic reasoning. By extrapolating these NLP-focused findings to the agentic domain, we identify the need to investigate whether context truncation serves as a necessary stabilization mechanism rather than just a cost-saving measure.

3 Methodology

This section details the experimental framework designed to evaluate the impact of context truncation on agentic reasoning. To answer our research questions regarding the relationship between context capacity and reasoning stability, we adopted a constructive experimental approach. We specifically chose to implement a dynamic simulation environment rather than a static evaluation because the phenomenon of cognitive thrashing (Section 1.4) is an emergent property of the agentic feedback loop that cannot be observed in single-turn benchmarks.

To ensure reproducibility and comparability with prior baselines, we adopted the *AICrypto* benchmark framework as our primary experimental framework, modifying it with a novel context intervention layer.

3.1 Experimental Setup

The testing environment consists of two distinct components: the base agentic framework (adapted from AICrypto) and our custom Context Management Middleware.

Base Agent Framework (AICrypto Adaptation)

We utilized the open-source agentic implementation provided by the *AICrypto* benchmark[18]. This framework instantiates a ReAct-style agent within a containerized Linux environment, providing the necessary tooling for cryptographic tasks.

For this experiment, the agent was configured with the following capabilities:

- **Tooling:** The agent was granted access to a standard Linux shell equipped with general programming tools (`python3`, `gcc`) and specialized cryptographic utilities including `openssl`, `sagemath`, `yafu`, and `flatter`. File I/O operations were handled via specific `create_file` and `read_file` actions.
- **Execution Loop:** We enforced a maximum limit of 20 turns (iterations) per challenge and a maximum runtime of 30 minutes per task to prevent infinite loops from consuming excessive resources. These constraints were calibrated based on preliminary empirical trials; furthermore, they also serve to penalize *computationally infeasible strategies* (e.g., brute-force enumeration) that models may propose without calibrated runtime intuition.
- **Observation Space:** The agent received the standard output (`stdout`) and standard error (`stderr`) of executed commands as raw text observations.

Novel Contribution: Context Management Middleware
To enable the controlled testing of context size—a variable not natively adjustable in the original AICrypto framework—we implemented a custom **Context Manager** middleware. This layer sits between the agent's reasoning engine and the LLM API endpoint, acting as a filter function.

Formal Definition of Context Truncation: Let $H_t = \langle m_0, m_1, \ldots, m_t \rangle$ denote the ordered sequence of messages at turn t, where m_0 is the fixed System Prompt. Let $\mathcal{T}(m)$ denote the token count of a message m.

Given an experimental token budget B, the middleware applies a truncation function $\Phi(H_t, B) \to H'_t$ defined as:

$$H'_t = \langle m_0 \rangle \oplus \langle m_{t-k+1}, \ldots, m_t \rangle$$

where $\oplus$ denotes sequence concatenation, and k is the maximum integer satisfying the constraint:

$$\mathcal{T}(m_0) + \sum_{j=0}^{k-1} \mathcal{T}(m_{t-j}) \leq B$$

This formulation ensures that H'_t strictly preserves the system prompt and the maximal contiguous suffix of the recent history that fits within B, discarding the intermediate subsequence $\langle m_1, \ldots, m_{t-k} \rangle$.

Implementation Details: To operationalize the token counting function $\mathcal{T}(\cdot)$, we utilized the *tiktoken* library [9] for precise accounting aligned with the model's specific tokenizer. Unlike standard API implementations that rely on opaque server-side truncation, our middleware explicitly calculates the maximal depth k at every turn. This mechanism effectively applies an "Oldest-First" truncation policy to the intermediate history $\langle m_1, \ldots, m_{t-k} \rangle$, while strictly preserving the System Prompt (m_0) to maintain the agent's identity and core task instructions.

Justification for Truncation Policy: We prioritized this deterministic truncation strategy over semantic compression techniques (e.g., recursive summarization or vector retrieval) to isolate context capacity as the sole independent variable. Incorporating a summarization module would introduce a significant confounding factor—the accuracy and quality of the summary itself—making it impossible to distinguish between failures caused by limited window size versus those caused by information loss during compression. By adhering to a rigid First-In-First-Out (FIFO) truncation protocol, we ensure that experimental outcomes are purely a function of available token capacity.

3.2 Dataset Selection

To manage computational costs while ensuring statistical significance, we utilized a **stratified subsample of 50 challenges** from the full AICrypto dataset. Crucially, this stratification was informed by the category-specific baselines established in the original AICrypto study [18] across 18 models. Given our research focus on *constraint optimization* and *performance degradation*, we prioritized categories—specifically *RSA*, *Discrete Logarithm (DLP)*, *PRNG/Hash*, *Block Ciphers*, and *Classic Cryptography*—that demonstrated quantifiable, non-zero success rates in unconstrained settings. This filtering was essential to avoid "floor effects"; to accurately measure the degradation caused by context truncation, the baseline agents must first possess the fundamental capability to solve the challenge. Consequently, extremely high-difficulty categories with near-zero historical solve rates were excluded to ensure that failure modes could be attributed to context constraints rather than a lack of domain knowledge.

3.3 Experimental Design

We structured the experiment to isolate the effect of context length on model performance. By holding the task set and agent tooling constant, we varied the input context window across different thresholds to observe changes in solvability and behavior.

Independent Variable: Context Budget
We strictly controlled the available context window at four distinct levels:

- **Strict Constraint:** 8,000 tokens.
- **Moderate Constraint:** 16,000 tokens.
- **Relaxed Constraint:** 32,000 tokens.
- **Baseline (Control):** 64,000 tokens.

Each task was evaluated with a single execution pass per model-context configuration. While this design choice limits statistical power, it reflects realistic deployment scenarios where repeated attempts are cost-prohibitive. The implications of this single-pass evaluation for result interpretation are addressed in Section 7.2.

We designated 64,000 tokens as the baseline (unconstrained) condition based on both preliminary empirical testing and recent literature regarding long-context reasoning limits. While modern models technically support windows exceeding 1M tokens, we treat 64k as a saturation point where reasoning stability is expected to be the limiting factor rather than token availability. This aligns with the "Effective Context" saturation thresholds identified by Modarressi et al. [8] and Hsieh et al. [6], effectively establishing an upper bound for useful history in reasoning-intensive tasks.

Dependent Variables and Metrics
We assess agent performance using four primary metrics:

1. **Solve Rate:** A binary success metric determined by the agent's ability to generate the correct flag string (verified against ground truth).
2. **Time to Termination:** The duration from task start to termination (seconds), capped at 1800s (30 min) timeout serving as an indicator of efficiency and latency.
3. **Failure Mode Taxonomy:** Failed runs are classified into primary categories:
    - **Timeout:** The task exceeded the 30-minute execution budget before producing a solution.
    - **Looping:** The agent repeats identical or near-identical action sequences across consecutive turns, indicating reasoning stagnation.

- **Wrong Answer:** The agent terminates with an incorrect flag submission or fails to produce a parseable solution.

Additionally, we perform analysis on agent run logs to identify:

- **Hallucinations:** False claims of success, fabricated file contents, or invocation of nonexistent tools.
- **Fragile Execution:** Syntactically plausible but semantically incorrect tool calls (e.g., referencing stale variables, off-by-one errors).
- **Command Repetition / Error Loops:** Repeated failed commands without adaptive correction.

3.4 Model Selection and Configuration

We selected three state-of-the-art LLMs for this study: **Gemini 3 Pro Preview**, **GPT-5.1**, and **DeepSeek-V3.2**. The selection was guided by two criteria: (1) support for the required context ceilings in our experimental design (up to 64k tokens), and (2) continuity with the lineage of models evaluated in the original AICrypto benchmark, enabling meaningful comparisons to existing literature.

Table 1: Model Specifications.

Model	Reasoning	Claimed Context (Tokens)
DeepSeek-V3.2	No	163k
Gemini 3 Pro Preview	Yes	1M+
GPT-5.1	Yes	400k

DeepSeek-V3.2 as an Architectural Control To isolate *context-induced* failure from effects attributable to hidden test-time compute, we deliberately evaluated **DeepSeek-V3.2** in standard inference mode. This is a methodological control rather than a limitation: optional reasoning modes can introduce additional latent computation whose intermediate artifacts may not be consistently exposed through provider APIs, reducing transparency and complicating attribution when comparing error modes across context budgets. In contrast, standard inference ensures that the observable agent trace (prompts, actions, tool outputs, and model replies) remains the primary substrate for analyzing Cognitive Thrashing, loss of partial progress, and other context-driven instabilities emphasized in Section 1.1.

This choice does *not* imply that other models are "double-reasoning" or experimentally compromised. **GPT-5.1** and **Gemini 3 Pro Preview** were run under their *default API behavior* to reflect standard deployment. If these systems perform internal reasoning as part of their default inference, we treat that behavior as an *intrinsic property* of the model class (and thus part of the baseline established by previous studies), not as an extra experimental manipulation. The only experimental factor systematically varied across all models is the enforced context budget (8k–64k) and its associated context-management policy within the agent framework.

Finally, DeepSeek-V3.2 functions as an *economic efficiency baseline*. By pairing a highly efficient open-weight model with strict, tunable context budgets, we test whether careful context management and agent architecture can narrow performance gaps to large proprietary systems—thereby separating gains due to "more tokens" from gains due to agent design under constraint. All models were executed using default provider settings except for the explicit context-budget constraints imposed by our framework.

4 Results and Analysis

We evaluate Deepseek-V3.2, Gemini-3-Pro-Preview, and GPT-5.1 across four context budgets (8k, 16k, 32k, 64k) to address our research questions on context-constrained agentic performance.

4.1 Context Saturation Threshold

Figure 1 reveals a non-linear relationship between context availability and solve rates. The critical "knee" appears at 16k tokens, where all models achieve 85–95% of their maximum performance. This finding directly addresses **RQ1**, indicating that agents achieve functionally-equivalent performance to larger contexts at approximately 32k tokens. Beyond this threshold, returns diminish sharply: DeepseekV3.2 and GPT-5.1 show **0.0% improvement** from 32k→64k, maintaining static solve rates of 44.0% and 60.0% respectively. Task duration also increases with context size (Appendix Figure 9), reflecting longer reasoning chains and more API calls at larger budgets.

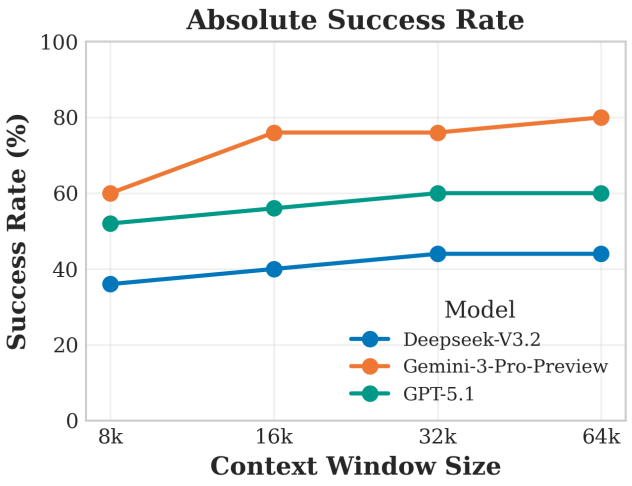

Figure 1: Absolute success rate across varying context window sizes.

The marginal gain analysis (Figure 2) quantifies **RQ2**: the 8k→16k transition yields the highest return-on-investment (ROI) (+16.0% for Gemini). In contrast, utility drops starkly after 16k, with the 32k→64k transition providing near-zero gains and statistically negligible benefits across all models. This confirms a clear saturation threshold—additional context beyond 32k does not translate to proportional performance improvements. This plateau effectively aligns with the empirical observations of Modarressi et al. [8], whose

evaluation of the "Effective vs. Available Context" gap identified 32,000 tokens as the critical threshold where reasoning stability transitions into a state of diminishing returns.

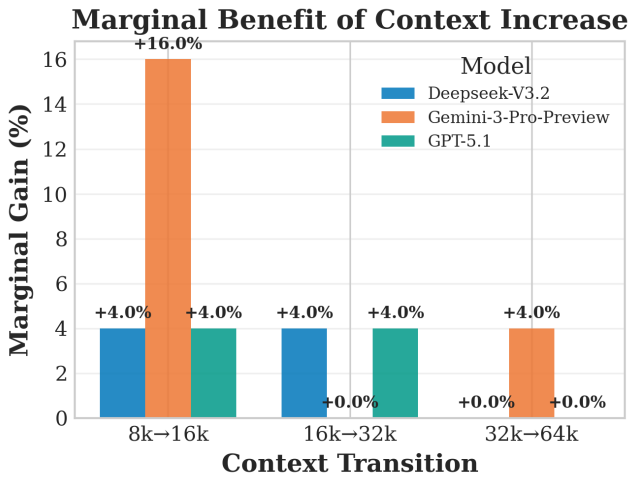

Figure 2: Marginal gain per context transition. Utility drops precipitously after 16k, with near-zero gains at 64k.

4.2 Failure Mode Dynamics

Figure 3 shows the aggregate failure distribution shifts as context expands. Looping behavior persists across all configurations, while timeout failures increase marginally with larger windows—suggesting that extended context enables longer (but not necessarily more effective) reasoning chains.

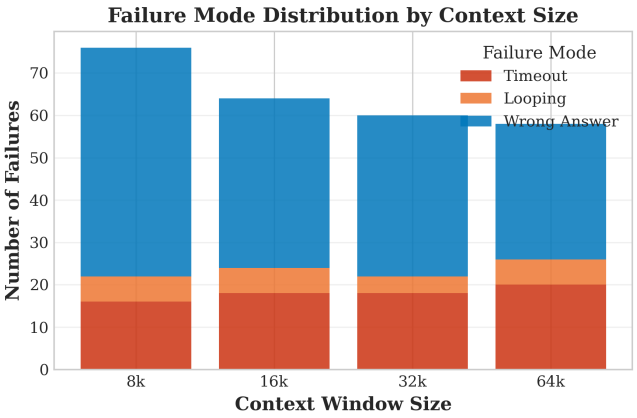

Figure 3: Failure mode distribution by context size. Wrong answers decrease with context while timeouts increase slightly.

The hallucination rate (Figure 4) reveals a nuanced pattern addressing **RQ3**:

- **Context Starvation (8k):** Hallucination rates peak at 3.2% (Gemini) and 1.6% (Deepseek), indicating models fabricate information when history is truncated.
- **Context Overload (64k):** Gemini and GPT show *increased* hallucinations at 64k vs. 32k (2.1% vs. 0.5% for Gemini). We attribute this to task composition bias. 64k spends considerably more time on the hardest unsolved challenges leading to potential noise accumulation from extended conversation histories.

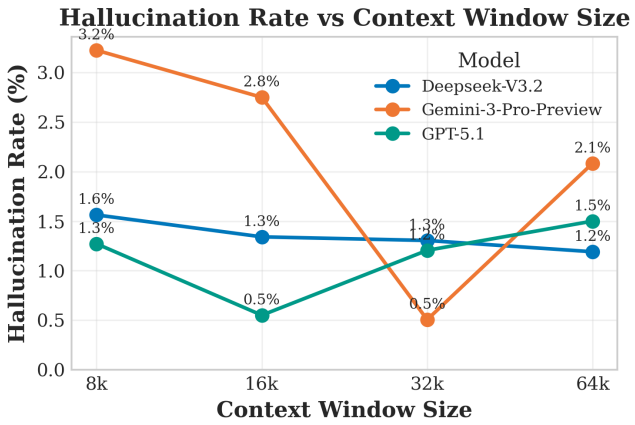

Figure 4: Hallucination rate normalized by agent turns. Deepseek shows monotonic decrease; Gemini and GPT exhibit non-monotonic patterns with upticks at 64k.

- **Fragile Execution:** Subtle tool-use errors peak at 64k (23 instances vs. 8 at 8k), suggesting that larger context introduces state-tracking complexity that degrades execution precision.
- **Timeout (Compute-Blind Strategies):** A significant portion of timeouts stemmed from models proposing computationally infeasible approaches—brute-force key enumeration or exhaustive search—that exceeded the 30-minute budget despite being theoretically valid.

A per-model failure breakdown is provided in Appendix Figure 7.

4.3 Economic Trade-offs

Figure 5 quantifies the cost efficiency across context configurations. Cost per successful solve increases non-linearly: Deepseek maintains sub-dollar efficiency ($0.06–$0.15/solve), while Gemini exhibits steep scaling ($0.46 at 8k to $1.10 at 64k).

The efficiency frontier (Figure 6) reveals the Pareto trade-off: the 16k→64k transitions move substantially rightward (higher cost) with minimal upward movement (performance). For Gemini-3-Pro, 64k incurs a **2× cost increase** over 16k ($21.47 vs $14.37) for only 4% absolute gain—operationally unfeasible for most deployments.

Synthesizing these economic metrics with the saturation findings of **RQ2**, we identify the optimal operating point at **16k–32k tokens**. It is critical to recognize that the costs reported here represent a conservative *lower bound*, effectively capped by our experimental constraints (maximum 30-minute runtime, 20 iterations). In unconstrained deployment scenarios, the "Cognitive Thrashing" observed at high context depths would likely lead to financial penalties significantly, inflating costs without yielding proportional performance gains.

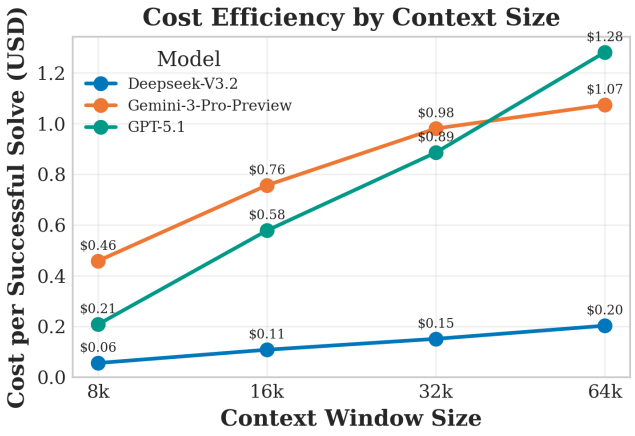

Figure 5: Cost per successful solve by context size. Diminishing returns are evident beyond 16k for all models.

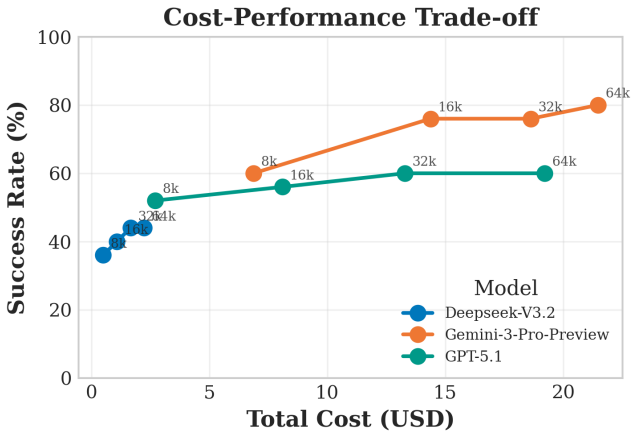

Figure 6: Cost-performance trade-off. Context labels show diminishing returns beyond the 16k–32k range.

5 Discussion

5.1 Effective vs. Available Context

Modern LLMs advertise context windows of 128k–1M tokens, yet our results reveal a stark disconnect between *available* and *effective* context. The saturation plateau at 16k–32k (Figure 1) suggests that for logic-intensive tasks, attention mechanisms fail to prioritize relevant signals over accumulated noise.

This aligns with the "Lost in the Middle" phenomenon [7], where models disproportionately weight tokens at sequence boundaries. Modarressi et al.'s NOLIMA framework [8] similarly demonstrates that effective context degrades non-linearly with window size. Our cryptographic domain—where preciseness and multi-step algebraic reasoning are essential—amplifies this effect: **adding context beyond the model's reasoning capacity acts as interference rather than memory**.

5.2 The ReAct Redundancy Effect

The distinct transition from hallucination to fragile execution observed in Section 4.2 supports a phenomenon we term **ReAct Redundancy**. In an unconstrained agentic loop, the context window inevitably accumulates a history dominated by failed attempts. When the model attends to substantial sequences (e.g., 50k tokens) of its own prior errors—abandoned strategies, repeated error messages, and syntax corrections—it becomes statistically biased toward those failure patterns. This explains the persistence of *action looping* and *execution fragility* at high context depths: the agent effectively confuses stale, failed states with the current objective.

Aggressive truncation thus acts as a critical stabilization mechanism, clearing this recent failure cache, so agents can attempt fresh strategies unbiased by prior mistakes.

5.3 Architectural Leverage over Model Scale

Deepseek-V3.2—a standard (non-reasoning) open-weight model—achieved 36% success at 8k, outperforming unconstrained baselines from previous-generation models [18]. This demonstrates that **complexity can be offloaded from model weights to agent architecture**.

The combination of ReAct scaffolding, structured tool interfaces, and principled context management delivered competitive performance at a fraction of the cost ($0.50 total for 50 tasks at 8k vs. $21+ for Gemini at 64k). For practitioners, this implies a high-ROI strategy: rather than defaulting to expensive reasoning-mode APIs (GPT-5.1, Gemini-3-Pro-Preview), invest in robust orchestration layers that manage context intelligently.

Combined with the efficiency frontier analysis (Figure 6), we recommend **16k–32k tokens** as the operational sweet spot for autonomous security agents—balancing solvability, cost, and cognitive stability. Deploying at 64k+ is not merely expensive; it is computationally wasteful with negligible benefit, contradicting Green AI principles [14].

6 Responsible Research

This research adheres to the TU Delft Code of Conduct [12]. We are committed to the principles outlined in the Integrity Statement throughout both the research and reporting phases of this paper.

6.1 Academic Integrity

In alignment with the core value of academic integrity, we have ensured honesty and due diligence in our reporting. We declare that all experimental results presented are genuine and have not been manipulated to fit a desired narrative. Furthermore, we have prioritized transparency by clearly distinguishing our own contributions from existing work, citing all external sources and tools appropriately to avoid plagiarism and respect intellectual property.

6.2 Computational Responsibility and Sustainability

Beyond standard academic integrity, this thesis addresses the ethical implications of resource allocation in Artificial Intelligence. As highlighted in our motivation, the prevailing indus-

try trend relies on the aggressive expansion of context windows—often exceeding one million tokens. This approach introduces an "Unjustified Cost" (see Section 1.1), where the computational expense of attending to hundreds of thousands of tokens yields diminishing or negative returns in reasoning performance.

From an ethical standpoint, we argue that this "more is better" paradigm contributes to unnecessary energy consumption and increased carbon emissions associated with inference. By challenging the necessity of infinite context and demonstrating that leaner, targeted context windows can outperform massive ones, this research advocates for the principles of *Sustainable Computing* and *Green AI*. We believe that minimizing computational waste is not merely an engineering optimization, but an ethical obligation to reduce the environmental footprint of autonomous agents.

6.3 Reproducibility

The reproducibility of this paper's findings is supported by a transparent methodology and the exclusive use of publicly available tools, libraries, and datasets. All elements regarding the experimental setup are detailed in Section 3. Furthermore, the specific modifications applied to the adopted AICrypto agent [17] have been published and are accessible via GitLab at https://gitlab.com/yusufbarisk/rp-agentic/-/tree/main.

6.4 Use of LLMs

The use of LLMs within the scope of this paper was limited to practical assistance with LaTeX formatting and minor syntactical refinements. No LLMs were utilized to generate the core scientific analysis or experimental outcomes of this work.

7 Conclusion and Future Work

7.1 Conclusion

This thesis challenged the prevailing assumption that "more context is always better" for LLM agents. Through systematic evaluation of three frontier models across four context budgets on a selection of 50 cryptographic CTF challenges across the 5 selected categories, we addressed our research questions:

- **RQ1 (Threshold):** Functional equivalence to larger contexts is achieved at **16k–32k tokens**. Beyond this range, agents reach 85–95% of their maximum performance.
- **RQ2 (Saturation):** Context scaling exhibits strong diminishing returns. The 32k→64k transition yielded **0% improvement** for DeepseekV3.2 and GPT-5.1, confirming a hard saturation threshold.
- **RQ3 (Failure Dynamics):** Relaxing context constraints shifts failure modes from *hallucination* (context starvation) to *fragile execution* (context overload), with the ReAct Redundancy effect degrading reasoning quality at maximum context.

Our central finding is that **constraint optimization outperforms capability maximization**. The bottleneck for agentic performance is not context *size* but context the *signal-to-noise ratio of information* retained in the working window.

7.2 Limitations and Future Work

Non-Deterministic Evaluation. LLM outputs are inherently non-deterministic—identical prompts yield varying responses across runs. Our single-pass evaluation captures one sample from this distribution, potentially missing solutions the model could discover under different sampling conditions. Future work should adopt *Pass@k* evaluation [2], executing multiple attempts per task to estimate true capability bounds and reduce variance from unlucky sampling.

Compute-Blind Reasoning. A recurring failure pattern involved models proposing theoretically correct but computationally infeasible strategies—exhaustive key enumeration, brute-force factorization, or exponential search spaces—that exceeded the 30-minute execution budget. Models appear to lack calibrated intuition for runtime complexity, defaulting to "infinite compute" assumptions. Future work should explore *complexity-aware prompting* that explicitly signals computational constraints, or fine-tuning on execution traces annotated with runtime feedback.

Context Management Strategy. We employed FIFO truncation, discarding oldest turns without semantic consideration. Future work should investigate *semantic compression*—summarizing completed subtasks rather than deleting them—or Retrieval-Augmented-Generation (RAG) based retrieval over agent history to preserve critical context while reducing noise.

Model Diversity. Our evaluation covered three models in standard inference mode. Native reasoning models (DeepSeek-R1, Claude Opus) with internal chain-of-thought may exhibit different context dynamics. Explicit comparison would clarify whether extended internal reasoning substitutes for external context.

Domain Generalization. Cryptographic CTFs demand exact syntax and precise algebraic manipulation—an adversarial setting for context noise. The "context penalty" may attenuate in softer domains (creative writing, legal analysis) where approximate recall suffices. Cross-domain evaluation would establish boundary conditions for our findings.

References

[1] Talor Abramovich, Meet Udeshi, Minghao Shao, Kilian Lieret, Haoran Xi, Kimberly Milner, Sofija Jancheska, John Yang, Carlos E. Jimenez, Farshad Khorrami, Prashanth Krishnamurthy, Brendan Dolan-Gavitt, Muhammad Shafique, Karthik Narasimhan, Ramesh Karri, and Ofir Press. Enigma: Interactive tools substantially assist lm agents in finding security vulnerabilities. 2025.

[2] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code, 2021.

[3] Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. In *Advances in Neural Information Processing Systems*, volume 35, pages 16344–16359, 2022.

[4] Peter Denning. working set model of program behavior. *Communications of the ACM*, 1968.

[5] Peyman Hosseini, Ignacio Castro, Iacopo Ghinassi, and Matthew Purver. Efficient solutions for an intriguing failure of LLMs: Long context window does not mean LLMs can analyze long sequences flawlessly. In *Proceedings of the 31st International Conference on Computational Linguistics (COLING 2025)*. Association for Computational Linguistics, 2025.

[6] Cheng-Ping Hsieh, Simeng Olia, Yang Cheng, Dawn Song, et al. RULER: What's the real context size of your long-context language models? In *The First Conference on Language Modeling (COLM)*, 2024.

[7] Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173, 2024.

[8] Ali Modarressi, Hanieh Deilamsalehy, Franck Dernoncourt, Trung Bui, Ryan Rossi, Seunghyun Yoon, and Hinrich Schütze. Nolima: Long-context evaluation beyond literal matching, 2025.

[9] OpenAI. tiktoken: fast bpe tokeniser for use with openai's models.

[10] Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Anselm Levskaya, Jonathan Heek, Kefan Xiao, Shivani Agrawal, and Jeff Dean. Efficiently scaling transformer inference, 2022.

[11] Machel Reid, Nikolay Savinov, Denis Teplyashin, et al. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*, 2024.

[12] Sabine Roeser and Samantha Copeland. *TU Delft Code of Conduct*. Delft University of Technology (TU Delft), Delft, Netherlands, 2020. Accessed: January 14, 2026.

[13] JV Roig. How do llms fail in agentic scenarios? a qualitative analysis of success and failure scenarios of various llms in agentic simulations, 2025.

[14] Roy Schwartz, Jesse Dodge, Noah A. Smith, and Oren Etzioni. Green ai, 2019.

[15] Minghao Shao, Sofija Jancheska, Meet Udeshi, Brendan Dolan-Gavitt, Haoran Xi, Kimberly Milner, Boyuan Chen, Max Yin, Siddharth Garg, Prashanth Krishnamurthy, Farshad Khorrami, Ramesh Karri, and Muhammad Shafique. Nyu ctf bench: A scalable open-source benchmark dataset for evaluating llms in offensive security, 2025.

[16] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, pages 5998–6008, 2017.

[17] Yu Wang, Yijian Liu, Liheng Ji, Han Luo, et al. Aicrypto-agent: Official implementation for the aicrypto benchmark, 2025. GitHub repository.

[18] Yu Wang, Yijian Liu, Liheng Ji, Han Luo, Wenjie Li, Xiaofei Zhou, Chiyun Feng, Puji Wang, Yuhan Cao, Geyuan Zhang, Xiaojian Li, Rongwu Xu, Yilei Chen, and Tianxing He. Aicrypto: A comprehensive benchmark for evaluating cryptography capabilities of large language models, 2025.

[19] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023.

A Supplementary Figures

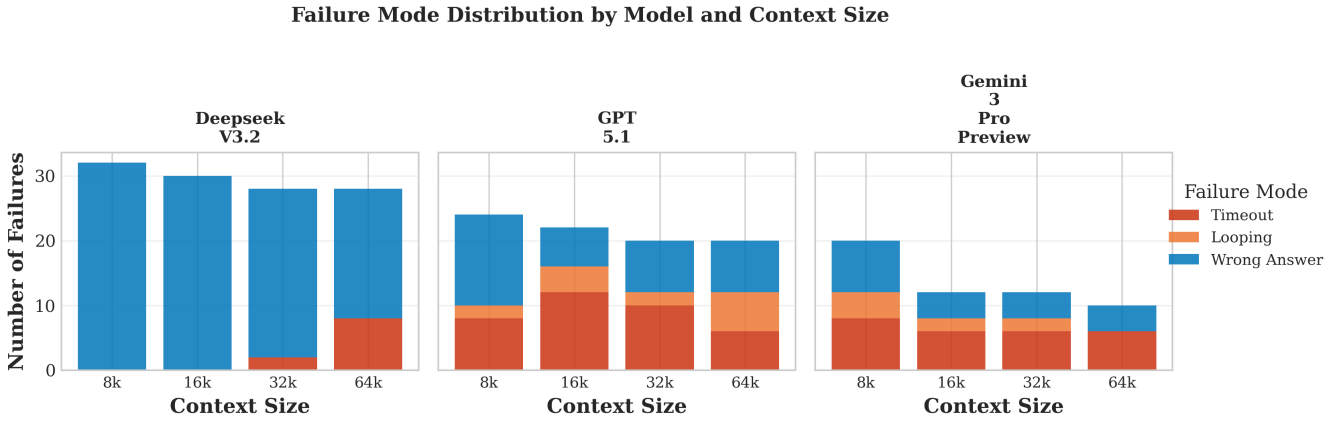

Figure 7: Failure Mode Distribution by Model and Context Size. Deepseek shows persistent looping behavior, GPT-5.1 exhibits balanced failure modes, while Gemini-3-Pro-Preview demonstrates the highest timeout rates at lower context budgets.

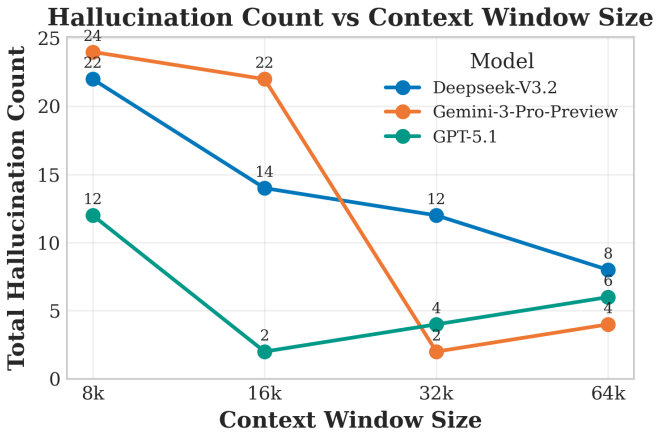

Figure 8: Raw Hallucination Counts by Model and Context. Absolute counts complement the normalized rates in Figure 4.

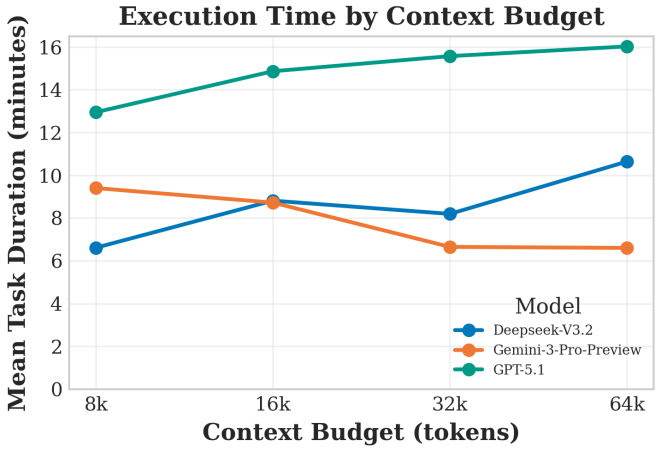

Figure 9: Mean task execution time by context budget. Shaded regions show ± 1 standard deviation.