Do Agent Architectures Matter for Crypto CTFs?

Querijn Voet

CSE3000 Research Project, TU Delft | Supervisor: Dr. Zeki Erkin

Background: LLMs solving *hard* problems

Recent advancements in Large Language Models (LLMs), such as Google's Gemini and OpenAI's o1, have demonstrated incredible achievements in pure mathematics. In July 2025, advanced versions of these models reached the International Mathematical Olympiad (IMO) gold-medal threshold.

However, these achievements test the capability of a core fundamental of a LLM (math) and has not transferred to cybersecurity competitions. When entered into real Capture The Flag (CTF) contests, state-of-the-art models solved zero challenges at professional events like PlaidCTF or DEF CON Qualifiers.

Why Cryptography is Harder than Pure Math

- **Obscure Primitives:** CTF challenges often rely on obscure variants of ciphers (e.g., custom elliptic curves) that appear rarely in training data.
- **Interactive Tooling:** Solutions cannot be solved "on-paper". They require writing scripts using libraries like `SageMath` and interactions with a remote server.

This raises a question: How capable are LLMs when they must **combine** these distinct fundamentals? Success requires more than just static reasoning. It demands the combination of mathematical logic and programming, together with the ability to dynamically acquire and apply obscure domain knowledge. So the challenge shifts from simply *knowing* an answer to iteratively *engineering* a solution using external tools.

Research Question

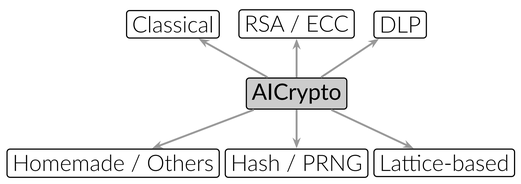
Current research evaluates LLMs on generic benchmarks (like HotPotQA). We lack empirical evidence on how the **control flow architecture** (the specific way an agent plans and loops) affects performance on domain-specific, high-computation tasks.

How does the architecture of an LLM-based agent affect success rate, speed, and compute cost on the AICrypto CTF benchmark?

To answer this, we fix the **Base Model** (Gemini) and **Tool Stack** (SageMath + Retrieval), and vary only the agent logic.

Benchmark: AICrypto

AICrypto provides a categorized dataset of challenges. This allows us to map architecture strengths to cryptographic subdomains.

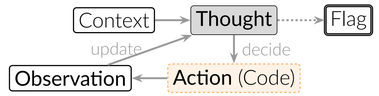


1. Baseline: Chain-of-Thought (CoT)

The model generates a linear path: reasoning followed by code. It executes once. If the tool fails (e.g., SyntaxError), the run fails.

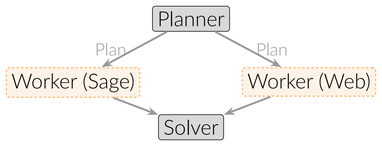
2. ReAct: Reasoning + Acting

A dynamic loop. The model observes the output of its actions and can self-correct. Critical for debugging scripts.



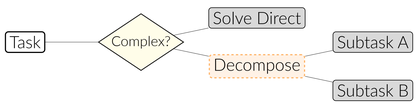
3. ReWOO: Plan-then-Execute

Decouples reasoning from execution. A "Planner" creates a blueprint with variable placeholders. "Workers" fill them in parallel.



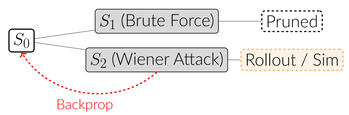
4. ADaPT: As-Needed Decomposition

Recursively breaks complex problems into sub-tasks, but only when necessary.



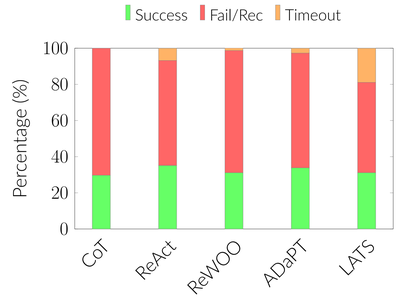
5. LATS: Language Agent Tree Search

Combines LLM reasoning with Monte Carlo Tree Search. Explores multiple attack vectors simultaneously and backtracks if a path fails.

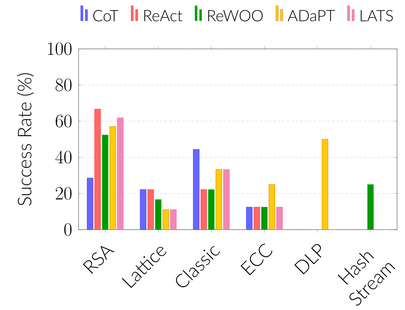


Results

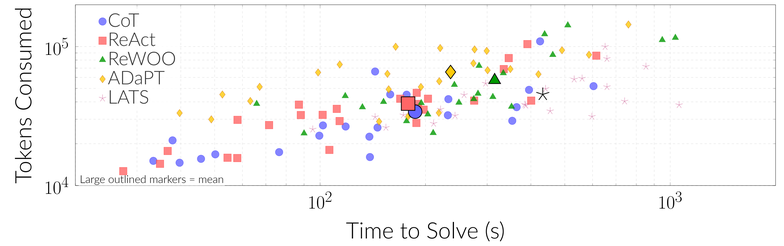
RQ1: Overall Performance



RQ2: Success Rate by Category



RQ3: Efficiency Trade-off (Log-Log Scale)



Conclusion

1. **Small Gains:** Agent architectures improve success rates, but only by 5.4% over the baseline (ReAct 35.1% vs CoT 29.7%).
2. **Engineering vs. Math:** The gain comes from the agent's ability to iteratively debug code rather than discovering new cryptographic insights.
3. **No single best architecture:** Complex agents do not universally outperform baselines; simpler approaches like CoT dominate in categories like 'Classic'.
4. **Cost of Intelligence** Complex planning (LATS/ADaPT) yields diminishing returns, consuming significantly more tokens and time (ADaPT uses 93% more tokens than CoT on successful runs).

Future Research Directions

- **Dynamic CTF Environments:** Moving beyond static files to interactive challenges (network interactions, remote servers).
- **Reliability Metrics (Pass@k):** Addressing LLM non-determinism by running challenges multiple times to get better confidence intervals.
- **Documentation:** Giving agents access to SageMath documentation to reduce engineering errors.

q.voet@student.tudelft.nl

https://github.com/QuerijnSharp/crypto-architectures