Can LLMs find protocol vulnerabilities through symbolic reasoning?

Cristian Curaba Denis D'Ambrosi Alessandro Minisini

July 26, 2024

Abstract

Cryptographic protocols have become ubiquitous in today's interconnected world, but are rarely verified within a formal threat model before being deployed. This may cause the unwanted introduction of hidden, but easily exploitable attack vectors to a distributed infrastructure, leading to a potentially catastrophic aftermath. On the other hand, formally verifying the security properties of new protocols is a complicated and tedious task that can be automated only partially through ad-hoc symbolic reasoning tools. In this work, we propose a novel and automated benchmark to evaluate Large Language Model (LLM) based agents on identifying protocols' vulnerabilities. In particular, we introduce a dataset of new protocols to analyze through a symbolic model checker and an innovative method to validate the results automatically. Furthermore, we investigate the performances of current frontier models in tackling the benchmark by testing a new AI agent design, powered by state-of-the-art prompting and scaffolding techniques. The integration of AI with formal verification systems has never been investigated in the context of cybersecurity, thus the results of this research will offer valuable insights for the development of future cyberdefense applications powered by symbolic-reasoning augmented LLMs.

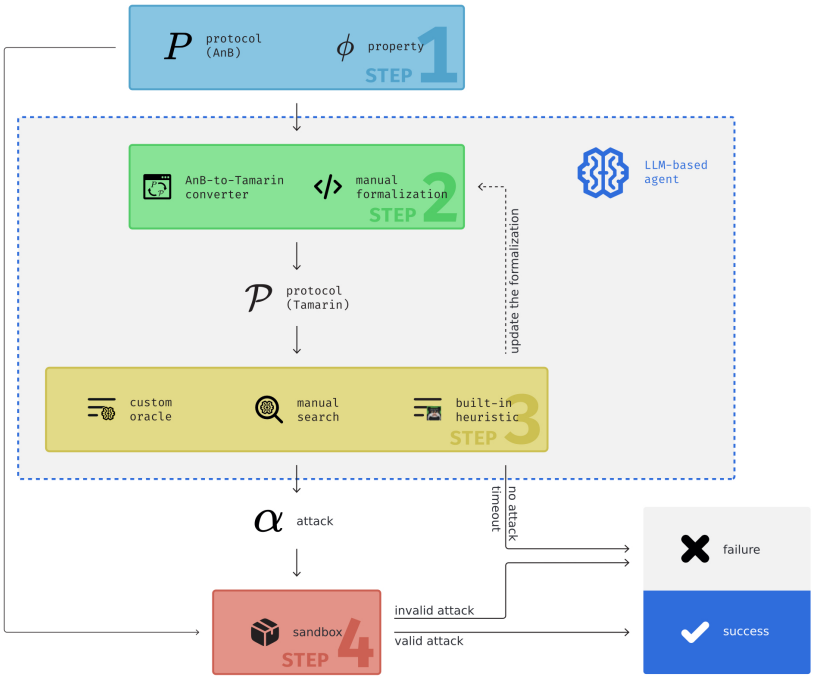

Figure 1: Overview of the benchmark's structure. The AI agent must identify a vulnerability in an unseen protocol by interacting with a symbolic model checker and iteratively adapting to its feedback until an attack is found, or a timeout occurs.

1 Introduction

Verifying the security properties of communication protocols is a well-studied problem in the domain of formal methods. This task is characterized by numerous theoretical complications, while also holding *significant industrial relevance for the secure deployment of distributed algorithms.* A security protocol consists of a sequence of messages, exchanged within a fixed set of parties, that aims at completing a task (generally exchanging a determined piece of information), while also guaranteeing that certain properties are met. Examples of famous security protocols are the 5G-AKA [1](current authentication and key exchange protocol for 5G-based mobile networks), SSH authentication [2] (authentication protocol for Secure Shell connections), X3DH and Double-Ratchet Algorithms [3] (implemented in Signal's and WhatsApp's encryption) and OAuth 2.0 [4] (used for authentication without password sharing on most websites). Although the complexity of such algorithms makes them look impenetrable, there have been several cases where *researchers have found hidden vulnerabilities in protocols well after their initial introduction in the literature.* The most infamous example is perhaps a pair of authentication protocols, proposed by Needham and Schroeder in 1978 [5], which were shown to be flawed respectively in 1981 [6] and 1995 [7]. Formally verifying protocols is crucial to avoid these situations, but is a very complex task that requires time and expertise of the limited group of researchers within this field. On the other hand, the recent proliferation of services reliant on communication technologies, exemplified by the diffusion of blockchain-based systems, has led to a surge in the *demand for new protocols designed to meet previously unforeseen requirements.* Most of the latest proposals cannot be properly checked for security before deployment, at least without the assistance of some automated tool to streamline the process. In this work, *we investigate the possibility of implementing such automatic systems by integrating symbolic reasoning tools with LLM-based agents.*

1.1 Contributions

This work's main contribution consists of a novel benchmark to determine if (current and future) *LLMs can identify vulnerabilities in cryptographic protocols.* In particular, the most distinctive innovation of this project lies in allowing AI agents to exploit external formal verification tools to complete the task. To the best of our knowledge, this marks the first instance where the complementary strengths of symbolic reasoning tools and LLMs are united to *tackle a realistic cybersecurity challenge.* The pipeline we propose for the benchmark consists of multiple steps, during which the AI agent should strategize its interactions with the software in order to identify a flaw. Additionally, since there are potentially infinite attacks that exploit the same vulnerability, we introduce a new *automated method to evaluate the final output.* Finally, we provide some numeric results concerning the performance of current frontier models on this benchmark.

Specifically, our research questions/objectives are the following:

RQ 1: Propose a *novel benchmark* that tests the capabilities of LLM-based agents in the field of security verification by formalizing protocols within a symbolic model checker and interacting with the verification tool to prove given properties.

- **RQ 1.1**: Create a dataset of new, realistic protocols, along with their relative security properties.
- **RQ 1.2**: Create an evaluator to automatically verify the correctness of the final output.

RQ 2: *Assess how well current state-of-the-art LLMs perform* on such a benchmark.

- **RQ 2.1**: Propose an AI agent based on current frontier models capable of tackling the benchmark.
- **RQ 2.2**: Disclose the main obstacles that AI models generally fail to overcome.
- **RQ 2.3**: Identify particular bottlenecks during the process.
- **RQ 2.4**: Analyze performances over input complexity.

To ensure *maximal public benefit*, we will publish our infrastructure and agent as a GitHub repository, while providing the dataset upon request to prevent exploitation through memorization. We will evaluate various LLMs and continually update these evaluations with future models to track advancements in reasoning capabilities. Additionally, we plan to submit a scientific article to a major peer-reviewed AI conference to publish the results of our experiment.

By sharing our observations, we plan to provide significant contributions to the fields of computer-aided cryptography and AGI safety. Since our benchmark is meant to test a real-world skill that could be exploited either to implement advanced AI-supported security engineering systems or automated-attacking tools, it is crucial to quantify and keep track of AI reasoning capabilities within this context: a failure in doing so may lead to a potentially catastrophic underestimation of current threats.

1.2 Related Works

LLMs have recently been tested extensively on cybersecurity benchmarks. They have been shown impressive capabilities even in complex and realistic scenarios, such as Capture-The-Flag challenges [8], social engineering tasks [9] and Common Vulnerabilities and Exposure (CVE) exploiting [10]. Unsurprisingly, the overall results have shown that AI agents perform better on tasks that have already been solved in the past and whose walkthrough probably ended up in a training dataset.

Additionally, Neural Networks have already been used to implement a data-driven search heuristic for the Coq Proof Assistant [11], leading to an impressive 37.2% top-10 accuracy on the IsarStep benchmark for mathematical reasoning. Furthermore, LLMs have shown unexpectedly good capabilities in guiding Coq and Lean proofs through in-context learning [12], completing the demonstration of more than 70 theorems. Such results show that Machine Learning models can be used to implement powerful heuristics for mathematical reasoning.

In a recent study [13], researchers have also demonstrated that LLM-based agents can improve their success rate on logic tasks when coupled with symbolic reasoning tools. The main difference between our work and theirs lies in the complexity of the reasoning procedure, as our benchmark involves a problem that has been proven to be unsolvable through automatic means. As a consequence, the test we propose will evaluate both the formalization (as in [13]) and proof-guidance (as in [12]) capabilities of the LLMs.

Finally, researchers have tried using Machine Learning methods for forecasting security properties of communication protocols, but without including any formal verification algorithm in the pipeline [14]. This approach led to an unforeseen accuracy of 85% on randomly generated protocols and 80% on real-world protocols. Although this article shares various sides with our project, the methodology outlined does not provide any description of the attack found, as it consists of a "simple" binary classifier. As a matter of fact, in the paper, the authors label each generated protocol as either secure or not, without differentiating between the various types of security properties. In a real-world scenario, this would be problematic, as some protocols are not meant to enforce all of the most common security properties by design. Furthermore, in some critical scenarios, forecasting that a protocol is free of flaws would not be considered enough, since a formal proof would be required to ensure that, under a given threat model, no attack will ever be discovered.

2 Background

In this section, we briefly introduce the theoretical background needed to better understand the pipeline of our benchmark. We provide more precise definitions of security protocols and their verification, along with an explanation of some software choices we made for our methodology.

2.1 Security protocols

Security protocols are often described in Alice and Bob (AnB) notation: introduced by Dolev and Yao in 1983 [15], this model abstracts the complexity of real-world cryptographic operations to a purely algebraic view of the messages exchanged. Such an approach allows us to ignore the implementation details of realistic primitives, thereby simplifying the verification proofs and extending its applicability to larger and more complex protocols. Within this model, the messages exchanged are made up of logical terms, which can be either atomic (such as constants, public information or actual messages) or terms derived from the application of function symbols (such as an encryption procedure or a hash algorithm). An example protocol, expressed in AnB notation, is provided in Figure 2. In this model, the adversary is able to overhear, intercept, and synthesize any message, with their actions solely constrained by the limitations of the cryptographic methods employed (e.g.: an attacker cannot decrypt an encrypted message without the right key). The main alternative to this approach is the computational model, where messages are abstracted to bitstrings,

cryptographic primitives are seen as bitstring endomorphisms, and the attacker must find a strategy to invalidate a given property within a certain amount of steps. Computational proofs consider a more complete threat model, however are harder to derive, and thus are often avoided for the verification of large-scale protocols.

$$
\begin{aligned}
&A \to B : \texttt{'g'}^{a} \\
&B \to A : \texttt{'g'}^{b} \\
&A \to B : \texttt{senc}\left(m, \texttt{'g'}^{b*a}\right)
\end{aligned}
$$

Figure 2: The Diffie-Hellman protocol [16], written in AnB notation. In this case, we have three messages exchanged between two parties, A and B. The atomic terms are the constant `'g'`, the ephemeral keys a and b, and the final message m. These terms are combined by three different function operations: exponentiation $(.^{\cdot})$, multiplication $(. * .)$ and symmetric encryption $\texttt{senc}(.,.)$.

2.2 Formal Verification of Security Protocols

Symbolic model checking is one of the prototypical problems in the realm of formal methods. It consists in verifying whether an abstract representation of a system satisfies a given specification, both expressed in the same logical formalism. When the algorithm terminates with a positive result, we have the absolute certainty that the system will never fail in production, under any circumstance. On the other hand, negative results consist of actual interpretable counterexamples which show how a system may incur in a faulty behaviour. Symbolic model-checking algorithms are guaranteed to terminate when considering finite representations, however, this is not always the case.

In the context of network security, formally verifying a communication protocol consists of checking that all of its possible executions (traces) satisfy a given property, expressed as a logical formula. Specifically, we want to ensure that the set of traces that satisfy the specification of the protocol is included within the set of traces that satisfy the formula. Any execution that belongs to the former, but not the latter, is a valid, easily implementable attack. The main complication of this problem is that both sets are generally infinite, thus the symbolic model checking algorithm is not guaranteed to terminate (an issue known as *undecidability*) [17].

In our case, we opted to leverage the Tamarin Prover [18], as its algorithm is both sound and complete with regards to Dolev Yao's threat model[1]. To handle undecidability, Tamarin Prover implements various features that require human intervention, such as manually guiding proofs, user-defined trace restrictions and custom oracles to substitute the built-in search heuristic. Other provers, such as ProVerif [19] and OFMC [20], try to circumvent this problem either by making some internal approximations that can lead to false positive results, or by bounding the number of possible protocol executions in the search space, inevitably invalidating their soundness.

3 Methodology

In this section, we outline our approach towards achieving our contributions, starting with the introduction of a new dataset of protocols. We then present an overview of the benchmark we are proposing, highlighting the key aspects of each component. Finally, we explain the techniques we intend to employ towards developing a proficient AI agent to tackle the test.

[1]In practice, this means that any attack describable in Dolev Yao's notation can also be found by the algorithm if the protocol is correctly formalized within the prover, and vice versa.

3.1 Protocol Dataset (RQ 1.1)

In order to faithfully test the formalization capabilities of LLMs in contrast to their memorization skills, we aimed to create a dataset of new, unseen protocols. We tried generating such new protocols through a random walk on a context-free grammar (similarly to the algorithm proposed in [14]), but it led to very unrealistic and uninteresting results. Since our benchmark was conceived as a red-team evaluation, it does not need a huge number of examples to provide some quality insights about maximal capabilities, and thus some form of manual assistance during the process is acceptable.

Given that [21] provides an open-source dataset of real-world communication protocols and known attacks, and that LLMs have proven to be good Few-Shot learners [22], we plan on opting for an In-context learning approach for generating the examples through GPT-4. The synthetic protocols will then be manually filtered based on a set of criteria, such as executability (are all messages actually constructible for the parties?), freshness (is the protocol actually new?) and security (are there any identifiable flaws?). The latter can be easily checked "by hand", as we can provide some known insecure protocols as a reference in the prompt and then search for the known attacks in the produced example. Note that we may unintentionally discard valid protocols when are not able to identify any attack, but this is not a problem since we are interested in maximizing precision, rather than recall. This approach hugely simplifies the labelling procedure, as it allows us to avoid manually formalizing all the protocols to check for errors (recall that Tamarin is sound and complete, thus any attack found in Dolev Yao notation is also identifiable in the prover's formalism).

The examples will be saved in the dataset as protocol-property pairs and will be partitioned into three levels of perceived difficulty: easy, medium, and hard. Differentiating between complexities can be useful to obtain more detailed insights on the performances of the models, as it allows us to understand better how the quality of outputs degrades with respect to an increase in complication of the inputs. Alternatively to our subjective partition, we also identified three meaningful numeric criteria for measuring the complexity of protocols: the total number of messages in a protocol, the maximum level of term nesting among the messages of a protocol, and the length of the shortest attack (in terms of the number of messages) on a protocol.

3.2 Benchmark Pipeline (RQ 1.2)

The benchmark is structured according to the following pipeline (which is also summarized in Figure 1):

1. The protocol is provided to the AI agent in Alice and Bob notation, along with an incorrect property to verify.

2. The AI agent formalizes the protocol in Tamarin syntax. To make this task more faithful to a real-world scenario, we ease the reasoning task of the AI agent providing an additional tool that automatically translates AnB protocols into Tamarin [23]. This converter is the only one currently available for Tamarin syntax and is not capable of translating most of the security properties specified in our dataset, thus the agent will have to adapt the formalization accordingly[2].

3. The AI agent checks the correctness of the property. The proof, or the counter-example, can either be obtained automatically, through the built-in heuristic, through a custom tactic, implemented as a bespoke oracle, or by manually guiding the proof steps. Neither of the approaches is guaranteed to terminate on its own, so we expect the agent to iterate through steps 2. and 3. repeatedly in order to complete the task. An example of a reasonable strategy could consist of observing that the proof "loops" on a particular term, devising an inductive invariant that helps to avoid the loop (i.e. a so-called *support lemma*), and then executing the standard heuristic.

4. After finding a counter-example that contradicts the property, the AI agent must translate it back to Dolev Yao's model and feed it into a symbolic sandbox. The latter is a software that checks the correctness of the attack. This software, which acts as a model checker, takes as input the original protocol, the property and the attack and verifies that the produced output is correct.

[2]Note that our benchmark does not require the use of the converter at all: some agents may even perform better on self-produced code, considering that the output of the translator is not very "human friendly". We hope to get some interesting insights as a byproduct of this freedom of choice, as seeing the results of our (and future) evaluations may provide some information on whether AI agents are better at implementing from scratch or "understanding" and adapting existing formalizations.

The sandbox will be implemented using again the AnB-to-Tamarin converter as an external library since the tool computes an internal representation of the protocol that is suitable for easily executing all the required checks. Note that, at the moment, no software is publicly available to validate attacks within the Dolev Yao model and that this sandbox will probably constitute a significant contribution on its own (considering the proofs of the correctness of the algorithms involved). In the future, this could be also used to label the outputs of GPT-based vulnerability detectors.

3.3 AI Agent (RQ 2)

The agent architecture will exploit state-of-the-art methods (retrieval-augmented generation, LLM reasoning, In-context/memory recall and existing tools) to harness the full potential of modern LLMs. By exploring and evaluating various scaffolding techniques, we aim to identify an effective and innovative way to address the tasks. Here we highlight a selection of best practices and designs:

- Structuring LLM reasoning with methods like chain of thought ([24]), tree of thought ([25]), self-consistency ([26]) or uncertainty-routed chain-of-thought ([27]) is crucial to exploit the full potential of the next token prediction capability.
- Retrieval-augmented methods enhance the selection of best examples for the few-shot (in-context) learning technique, which is applied in the formalization task (to understand the Tamarin syntax) and to guide (by finding the best premises or goals) the symbolic software proof ([28], [29]).
- Self-reflection, as evaluated in [30], is a statistically proven method to improve performance through self-critiquing.
- Sketching the proof before interacting with the symbolic software is an effective method to enhance the capabilities to guide the proof as shown in ([29] and [31]).
- General practice to improve prompts ([32]) like profiling or using delimiters to divide the input.

A key method to robustly improve the agent design is to test changes systematically: we'll consider it after a minimum viable product is obtained. The AI agent pipeline will be implemented through the Langchain framework.

4 Early Results

Based on some early results we obtained on a minimum-viable product of the benchmark (developed during a hackathon with Apart's lab earlier this year [33]), we are confident that the custom-engineered AI agent will be able to successfully verify the properties of some mildly complicated protocols in our dataset. Such a result would confirm that the addition of reasoning systems to LLMs is a research avenue that is worth devoting effort to. We expect that a well-designed LLM-based agent for Tamarin could, in the future, be used to simplify the procedure of formally proving securities of new communication protocols before their actual deployment. On the other hand, we also forecast that the agent, powered with modern LLMs, will not be able to address complex examples: any proof of the opposite would show that the current AI models have enough offensive capabilities to cause harmful behaviour in the real world.

4.1 Limitations

The reasoning capabilities of the AI agent may not be strong enough to solve even the easy examples in the dataset. In this scenario, we will try to evaluate the capabilities of the agent through checkpointing: as the benchmark is developed as a sequence of tasks (formalizing, proving, and translating the attack), we can provide intermediate results and resume the evaluation from a middle point to separately test each skill required by the benchmark.

5 Conclusions and Future Directions

This proposal introduces a novel benchmark for evaluating the abilities of LLMs to identify vulnerabilities in cryptographic protocols using symbolic reasoning tools. Our approach combines the strengths of AI agents and formal verification techniques, through a well-defined pipeline that includes leveraging a symbolic model checker and utilizing an automated evaluation system. By doing so, we hope to contribute valuable insights to the development of advanced cybersecurity applications powered by symbolic-reasoning augmented AI systems, ultimately promoting safe communication protocols.

Future works may focus on expanding the benchmark to include positive security properties and further refining the AI agent's design to enhance its reasoning and formalization skills.

In conclusion, this paper sets the stage for a new era of automated security protocol verification, where the complementary strengths of symbolic reasoning and AI can be harnessed to address the growing complexity and demands of modern cybersecurity challenges.

References

[1] G. Arfaoui, P. Bisson, R. Blom, R. Borgaonkar, H. Englund, E. Félix, F. Klaedtke, P. K. Nakarmi, M. Näslund, P. O'Hanlon, J. Papay, J. Suomalainen, M. Surridge, J.-P. Wary, and A. Zahariev, "A security architecture for 5g networks," *IEEE Access*, vol. 6, pp. 22466–22479, 2018.

[2] C. M. Lonvick and T. Ylonen, "The Secure Shell (SSH) Protocol Architecture." RFC 4251, Jan. 2006.

[3] M. Marlinspike and T. Perrin, "The sesame algorithm: session management for asynchronous message encryption," *Revision*, vol. 2, pp. 2017–04, 2017.

[4] D. Hardt, "The OAuth 2.0 Authorization Framework." RFC 6749, Oct. 2012.

[5] R. M. Needham and M. D. Schroeder, "Using encryption for authentication in large networks of computers," *Commun. ACM*, vol. 21, pp. 993–999, Dec 1978.

[6] D. E. Denning and G. M. Sacco, "Timestamps in key distribution protocols," *Commun. ACM*, vol. 24, p. 533–536, aug 1981.

[7] G. Lowe, "An attack on the needham- schroeder public- key authentication protocol," *Information processing letters*, vol. 56, no. 3, 1995.

[8] W. Tann, Y. Liu, J. H. Sim, C. M. Seah, and E.-C. Chang, "Using large language models for cybersecurity capture-the-flag challenges and certification questions," *arXiv preprint arXiv:2308.10443*, 2023.

[9] N. Begou, J. Vinoy, A. Duda, and M. Korczyński, "Exploring the dark side of ai: Advanced phishing attack design and deployment using chatgpt," in *2023 IEEE Conference on Communications and Network Security (CNS)*, pp. 1–6, IEEE, 2023.

[10] R. Fang, R. Bindu, A. Gupta, and D. Kang, "Llm agents can autonomously exploit one-day vulnerabilities," *arXiv preprint arXiv:2404.08144*, 2024.

[11] W. Li, L. Yu, Y. Wu, and L. C. Paulson, "Isarstep: a benchmark for high-level mathematical reasoning," *arXiv preprint arXiv:2006.09265*, 2020.

[12] A. Thakur, G. Tsoukalas, Y. Wen, J. Xin, and S. Chaudhuri, "An in-context learning agent for formal theorem-proving," 2024.

[13] L. Pan, A. Albalak, X. Wang, and W. Y. Wang, "Logic-lm: Empowering large language models with symbolic solvers for faithful logical reasoning," 2023.

[14] K. Ohno and M. Nakabayashi, "A security verification framework of cryptographic protocols using machine learning," 2023.

[15] D. Dolev and A. Yao, "On the security of public key protocols," *IEEE Transactions on information theory*, vol. 29, no. 2, pp. 198–208, 1983.

[16] W. Diffie and M. E. Hellman, *New Directions in Cryptography*, p. 365–390. New York, NY, USA: Association for Computing Machinery, 1 ed., 2022.

[17] S. Even and O. Goldreich, "On the security of multi-party ping-pong protocols," in *24th Annual Symposium on Foundations of Computer Science (sfcs 1983)*, pp. 34–39, IEEE, 1983.

[18] B. Schmidt, S. Meier, C. Cremers, and D. Basin, "Automated analysis of diffie-hellman protocols and advanced security properties," in *2012 IEEE 25th Computer Security Foundations Symposium*, pp. 78–94, IEEE, 2012.

[19] B. Blanchet *et al.*, "Modeling and verifying security protocols with the applied pi calculus and proverif," *Foundations and Trends in Privacy and Security*, vol. 1, no. 1-2, pp. 1–135, 2016.

[20] D. Basin, S. Mödersheim, and L. Vigano, "OFMC: A symbolic model checker for security protocols," *International Journal of Information Security*, vol. 4, pp. 181–208, 2005.

[21] EVA national network project, "Security Protocols Open Repository." `http://www.lsv.fr/Software/spore/index.html`, 2003. [Accessed 28-06-2024].

[22] S. Cahyawijaya, H. Lovenia, and P. Fung, "Llms are few-shot in-context low-resource language learners," 2024.

[23] D. Basin, M. Keller, S. Radomirović, and R. Sasse, "Alice and bob meet equational theories," *Logic, Rewriting, and Concurrency: Essays Dedicated to José Meseguer on the Occasion of His 65th Birthday*, pp. 160–180, 2015.

[24] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. Chi, Q. Le, and D. Zhou, "Chain-of-thought prompting elicits reasoning in large language models," 2023.

[25] S. Yao, D. Yu, J. Zhao, I. Shafran, T. L. Griffiths, Y. Cao, and K. Narasimhan, "Tree of thoughts: Deliberate problem solving with large language models," 2023.

[26] X. Wang, J. Wei, D. Schuurmans, Q. Le, E. Chi, S. Narang, A. Chowdhery, and D. Zhou, "Self-consistency improves chain of thought reasoning in language models," 2023.

[27] G. T. et. Al., "Gemini: A family of highly capable multimodal models," 2024.

[28] K. Yang, A. M. Swope, A. Gu, R. Chalamala, P. Song, S. Yu, S. Godil, R. Prenger, and A. Anandkumar, "Leandojo: Theorem proving with retrieval-augmented language models," 2023.

[29] A. Thakur, G. Tsoukalas, Y. Wen, J. Xin, and S. Chaudhuri, "An in-context learning agent for formal theorem-proving," 2024.

[30] M. Renze and E. Guven, "Self-reflection in llm agents: Effects on problem-solving performance," *arXiv preprint arXiv:2405.06682*, 2024.

[31] A. Q. Jiang, S. Welleck, J. P. Zhou, W. Li, J. Liu, M. Jamnik, T. Lacroix, Y. Wu, and G. Lample, "Draft, sketch, and prove: Guiding formal theorem provers with informal proofs," 2023.

[32] OpenAI, "Prompt engineering." `https://platform.openai.com/docs/guides/prompt-engineering`, 2024. [Accessed 02-06-2024].

[33] Apart Research, "Code red hackathon: Can llms replicate and do r&d by themselves?." `https://www.apartresearch.com/event/codered`, 2024. [Accessed 28-06-2024].