Optimizing Small Language Models for Cryptographic Applications

Anthony Brogni brogn002@umn.edu
Evan Pochtar pocht004@umn.edu
Ajitesh Parthasarathy parth057@umn.edu
Tanmay Patwardhan patwa028@umn.edu

Abstract

We present a targeted investigation into the ability of fine-tuned small language models (SLMs) to break mono-alphabetic substitution ciphers. Using Qwen2.5-0.5B and Phi-4-mini-instruct as our base models, we generated a bespoke training corpus by programmatically encrypting 5,000 "Human vs AI" dataset text samples with random mono-alphabetic mappings. We then applied Group-Relative Policy Optimization (GRPO) fine-tuning and evaluated a novel test-time scaling mechanism designed to scale performance during inference. Our experiments measured accuracy using RapidFuzz similarity (a fuzzy similarity metric), comparing GRPO alone as well as GRPO combined with test-time scaling against initial baselines. We found that GRPO fine-tuning yielded improvements in the decryption accuracy of SLMs on this task. The test-time scaling mechanism was evaluated for its ability to reduce ambiguous substitutions, and its effect on overall performance is discussed. Our results quantify the effectiveness and highlight the limitations of these methods for lightweight cryptanalysis using SLMs, illuminating remaining failure modes and informing approaches for future research.

1 Problem Definition and Motivation

Our proposed project explores the novel application of fine-tuned small language models for cryptographic analysis, specifically mono-alphabetic substitution ciphers. By adapting Qwen2.5-0.5B, a lightweight model with just 500 million parameters, we challenge the assumption that only large models can excel at complex pattern recognition problems. This approach is important because it works as a proof of concept on creating cryptographic tools that are accessible on resource-constrained devices like smartphones or similar small gadgets, while also revealing insights about how neural networks process and decode symbolic patterns. We tested this on small reasoning models as well, such as DeepseekR1-1.5B, to see if chain-of-thought reasoning can improve the performance of our system. We also tested a larger SLM, Phi-4-mini-instruct, which has about 3 billion parameters, to assess if increased parameter count can increase the final accuracy.

The motivation for this project comes from both practical and theoretical issues with AI systems. While the performance of large language models is very impressive across many domains, the computational requirements make it hard to run for the general public. By showing that carefully fine-tuned small language models can match or exceed state-of-the-art performance in specific tasks such as cryptographic analysis, we aim to present a more efficient method of completing tasks with specialized AI systems that maintain high performance while reducing computational overhead. Mono-alphabetic substitution ciphers, wherein each letter is consistently replaced by another throughout the text, are a perfect test case, as they are complex enough to challenge pattern recognition capabilities in small models yet structured enough to evaluate performance objectively.

2 Related Work

Many advances in language models have led to the application of neural methods for cryptanalysis (Abood and Guirguis, 2018). Most cryptographic analysis solutions have relied on statistical methods and heuristic patterns such as the Markov Chain Monte Carlo (MCMC) technique (Chen and Rosenthal, 2012), which works by initializing a random substitution alphabet, calculating the probability that the current substitution alphabet produces valid plain text, randomly exchanging certain plain letter-cipher letter correspondences, and repeating steps 2 and 3 as long as the plausibility score of the message obtained increases and the alphabet selected

offers the most probable plain message. The three papers chosen below discuss systems made for natural language understanding or code analysis that can be repurposed for cryptographic tasks. The insights provided by the papers are relevant when considering resource-constrained implementations like fine-tuning small language models.

The first paper focuses on ChatGPT's capability for simple power analysis (SPA) on cryptosystems (Zhou et al., 2024). They note that SPA requires expert intervention and specialized tools. To combat this, the authors created a new prompt template that helps to guide the model. Using this prompt engineering approach to a dataset composed of various implementations of cryptographic algorithms, they showed that ChatGPT's outputs were insufficient and prompting strategies allowed for a successful recovery of private keys. The papers proved how domain-specific adjustments can be used to leverage AI for complex cryptanalysis tasks.

The second paper proposed a framework called FoC that focuses on identifying cryptographic functions within stripped binary executables (Shang et al., 2024). The method first introduced is FoC-BinLLM, a binary LLM model that generates natural-language summaries that capture the semantic essence of cryptographic routines. They then introduced FoCSim, which is a similarity model that refines the representation to receive analogous cryptographic implementations from a reference database. The results show that FoC-BinLLM works well for minor modifications such as vulnerability patches and is far better than ChatGPT and previous state-of-the-art methods. The papers give insight into the potential of leveraging fine-tuned language models for tasks requiring a deep semantic understanding of code.

The third paper offers a theoretical and empirical examination of adapting transformer models for cryptographic analysis (Baek et al., 2024). Even though the metadata provides limited details, its contributions are evident in the architectural optimization of transformers-based models to operate efficiently on resource-constrained devices. By exploring methods for model compression and fine-tuning, the paper bridges the gap between the high computational demands of standard models and the need for lightweight but effective solutions for cryptographic applications. The insights presented are valuable for understanding how transformer architectures can be modified to maintain performance for applications like cryptanalysis of mono-alphabetic substitution ciphers.

Collectively, these works provide compelling evidence that language models designed for natural language processing can be used for complex cryptography applications. They highlight the critical role of prompt engineering and domain-specific fine-tuning as seen in the adaptations of ChatGPT for automated SPA and the semantic extractions by FoC-BinLLM. The third paper reinforces the feasibility of modifying transformer architectures to create efficient and high-performance models suitable for resource-constrained devices. These insights are directly applicable to this project focusing on fine-tuning small language models for the analysis of mono-alphabetic substitution ciphers, as they prove that with careful adaptation, small models can achieve performance levels that rival those of larger ones while being accessible for practical, everyday applications.

3 Proposed Approach

Most works that use LLMs or classical statistical methods to solve these ciphers rely on heuristic-based techniques, which struggle with the tradeoff between model size and inference efficiency. We are trying to fine-tune a smaller model with GRPO and test-time scaling. We believe this will lead to lower computational overhead while maintaining its accuracy. The first novel idea is introducing GRPO, which can give us real-time feedback on its performance, which will allow it to adjust the learning strategy as it goes. We will also include a test-time scaling mechanism. This allows the model to adjust its confidence and output during the decryption. This will be helpful when dealing with various ciphers, which current models struggle with as they use fixed methods. Our plan to achieve this is to start by fine-tuning the Qwen2.5-0.5B and Phi-4-mini-instruct models on our dataset using GRPO and then incorporating a test-time scaling mechanism. We would want to incorporate each one individually and then both together so we can compare how each technique impacts the fine-tuning process. After we have gotten all our results, we will want to compare them to the initial performance baseline of the base models to see if these techniques made any improvements. In short, we believe that incorporating GRPO and test-time scaling can give us dynamic inference, which should be able to overcome current limitations.

Our GRPO system relies on a reward function that measures the similarity between predicted and correct outputs, guiding the model's learning process. We balance efficiency with accuracy using character-wise matching for equal-length strings (with GPU acceleration) and fuzzy matching for different lengths. A small offset adjustment for equal-length matches creates stronger signals for exact character sequences. What makes our approach novel is combining GRPO with smaller language models for cryptography, which is different from the current trend toward larger models or fixed heuristics and potentially enabling powerful tools on resource-constrained devices.

4 Additional Development of our Ideas after the Project Proposal

After submitting the Project Proposal, we got to work fine-tuning a small language model (Qwen2.5-0.5B) using Group Relative Policy Optimization (GRPO) with the Huggingface GRPOTrainer implementation. For training, we are utilizing the popular "Human vs AI dataset" from Kaggle, which contains 500,000 text samples, both human-written and AI-generated. We originally planned to use the "Encrypted Text: Mono-Alphabetic Cipher" dataset from Kaggle, but determined that it would be too challenging as the ciphers used in this dataset map all ascii characters (including things like punctuation) to other ascii characters, as opposed to only mapping lowercase English letters to other lowercase English letters, which is what we wanted. Therefore, when we found out that this dataset from Kaggle was created from the "Human vs AI dataset", we decided to just use that directly and encrypt the text samples ourselves. So we use the "Human vs AI" dataset and programmatically apply random mono-alphabetic substitution ciphers to the text samples in Python and use these transformed (encrypted) texts as training data, where the expected outputs are the original unencrypted texts. We anticipated performance limitations with smaller models and methodological challenges in matching the accuracy of larger models, as our initial testing showed us that even state-of-the-art models frequently failed at this task. Surprisingly, Claude 3.7 Sonnet demonstrated exceptional accuracy, contradicting our expectations. Early tests showed that our own trained small model frequently repeated the ciphertext verbatim or only partially solved sections, and it also often produced long explanations without addressing the cipher itself. We eventually identified one cause of this as the model's inability to perform basic letter substitution, failing even at simple tasks like replacing "a" with "b" in a sentence. This required us to refine our training approach to encourage a complete text solution and to first train fundamental substitution skills before attempting full text decryption.

4.1 Evaluation Metrics

We measured decryption success using accuracy scores comparing the model's predicted decryption to the ground truth (original text). Two metrics were used:

- **Character Correctness Percentage:** For predicted outputs of the same length as the ground truth, we calculated the percentage of characters that matched exactly at each position.
- **Fuzzy String Matching Score:** For predicted outputs of potentially different lengths, we used the 'fuzz.ratio()' function from the RapidFuzz Python library, which calculates a normalized Levenshtein distance-based similarity score.

Both metrics range from 0% (no similarity) to 100% (perfect match). Our final results presented are averaged over a test set of multiple cipher prompts.

5 Results and Analysis

We evaluated our project in three main avenues: one was the investigation of directed prompts and their performance, the second was evaluation of reasoning models and tokens generated versus accuracy, and finally, a general evaluation of large, trained, and small LLMs on monoalphabetic substitution ciphers. To get an absolute accuracy score, we use character correctness percentage for equal-length outputs, and RapidFuzz for variable-length comparisons, which is a type of edit distance accuracy calculation. All metrics range from 0-100%, with 100% representing perfect accuracy. Our first round of tests investigated the performance of different types of prompts on the same task, specifically directed and undirected versions of the same prompt. In this context, directed prompts mean that detailed step-by-step instructions were provided in the prompt, whereas undirected prompts simply asked the model to decipher the text. Surprisingly,

through testing many different LLMs on both types of prompts, we found that the models performed better when given a more undirected prompt. We believe that this is because the extra directions tend to confuse the models and make them less likely to solve the task. This could be due to the fact that the models are not able to understand the extra instructions, or it could be that they are trying to follow the instructions too closely and not focusing on the task at hand. On all LLMs tested, only Qwen's accuracy improved with the more directed prompts, while all other models performed better with the less directed prompts. This led us to choose a non-directed prompt for our final model.

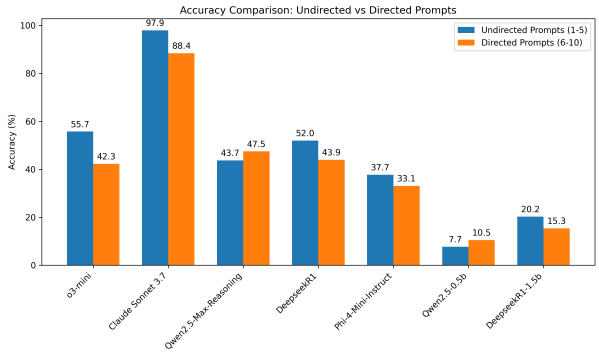

Figure 1: Results of 10 total directed versus non-directed prompts in major LLMs. Orange represents directed prompts, and blue represents undirected prompts. All values are percentages out of 100.

We also evaluated the models' reasoning capabilities by analyzing the number of tokens generated and their accuracy, which we believe is a good indicator of the model's ability to solve the task. We hypothesized that models with more tokens would have higher accuracy, but our results showed that this was not the case. In fact, the models with the highest accuracy generated fewer tokens, indicating that they were more focused on the task at hand. This suggests that reasoning models may not be the best choice for this type of task, as they tend to generate more tokens and less accurate results, and tend to overthink. When it cannot solve the task, it will often generate a lot of tokens in an attempt to explain its reasoning, reach a token max limit, and then return a random English sentence that is not related to the task at hand. This evidence lead us to believe reasoning models aren't necessarily better for this task, and focusing on a non-reasoning model could help us both speed up training as well as create a more accurate final model.

Using these criteria, we've tested numerous

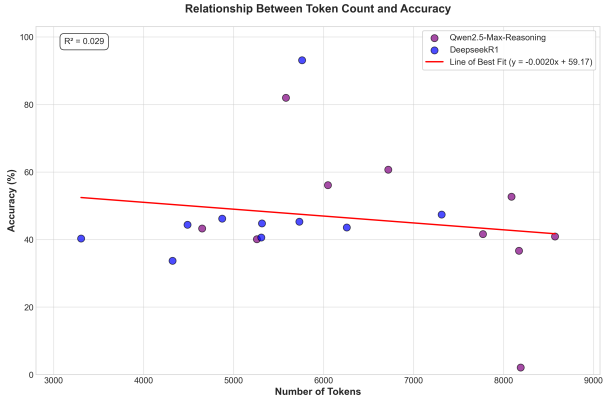

Figure 2: Tokens generated versus final plaintext output accuracy. A token in this case is retrieved using Deepseek's token calculation, which can be found here. 20 Prompts were tested on both Qwen2.5-Max-Reasoning as well as DeepseekR1.

mainstream large models to validate our hypothesis regarding their inability to solve this task. We've also evaluated our small model with and without our novel training methods, demonstrating that these approaches enhance model performance by about 6% for both Qwen and Phi. This rivals large models like Qwen2.5-Max, but cannot reach the heights of a model like Claude 3.7 Sonnet. This limitation stems primarily from small LLMs' difficulty with letter substitution and performing complex reasoning within constrained timeframes. For these evaluations, we used ten directed and undirected prompts with identical tasks but different texts and measured their accuracy.

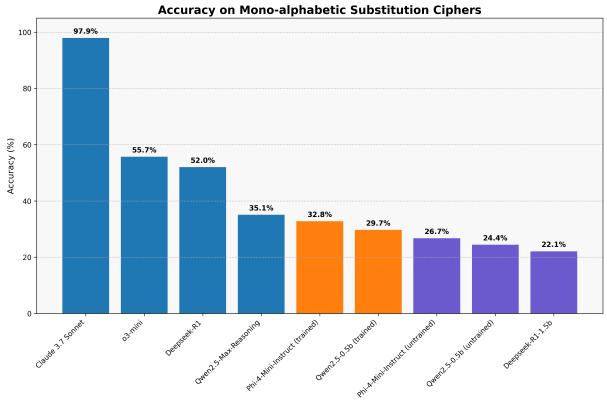

Figure 3: Final results of Monoalphabetic substitution on LLMs. Light blue bars refer to LLMs tested on 10 directed prompts, while orange and purple bars are tested on 500 prompts from our original dataset. Orange defines our final trained models.

5.1 Error Analysis

Our results indicate that while GRPO fine-tuning improves the ability of SLMs like Qwen2.5-0.5B and Phi-4-mini-instruct to handle mono-alphabetic ciphers compared to standard fine-tuning, significant challenges remain. The primary reasons for failure appear to be:

- **Fundamental Substitution Difficulty:** As identified during development, SLMs seem to lack robust mechanisms for consistent, arbitrary symbol mapping across long contexts. This is a core requirement for this task. GRPO helps, but may not fundamentally alter this limitation without architectural changes or different pre-training.
- **Reasoning Limitations:** Solving these ciphers often involves implicit frequency analysis and pattern matching across the entire text. SLMs have limited capacity for such complex, multi-step reasoning compared to larger models. This might explain why they resort to partial decryptions or repetitions.
- **Training Signal Sparsity:** Even with GRPO, the reward signal (decryption accuracy) might be sparse, especially early in training when outputs are random. Getting meaningful feedback to guide learning towards the correct substitutions is difficult.
- **Comparison to Claude 3.7 Sonnet:** The high performance of Claude 3.7 suggests that sufficient scale and potentially specific architectural or training data aspects in larger models *can* overcome these challenges. Our SLM fails where Claude succeeds likely due to these scale and reasoning capacity differences. Conversely, our fine-tuned SLM likely outperforms *other* LLMs (like GPT-3.5 in Figure 1) because its training was highly specialized for this specific task, whereas the general LLMs rely on emergent abilities without task-specific fine-tuning.

5.2 Analysis of Failure Cases

Persistent failure cases often involve longer texts or texts with less common letter frequency distributions (making statistical clues weaker). Our current approach, relying on fine-tuning a general-purpose SLM architecture with GRPO, likely cannot fully address these because:

- **Limited Context Window/Memory:** SLMs struggle to maintain consistency and track substitutions over very long passages.
- **Inherent Architectural Limits:** The standard transformer architecture in SLMs may not be optimally suited for the kind of precise, rule-based symbol manipulation required, compared to tasks relying on semantic understanding.

Potential solutions to explore in future work include:

- **Curriculum Learning:** Start by training the model on extremely simple substitution tasks or very short texts, gradually increasing complexity.
- **Hybrid Approaches:** Combine the SLM with a classical statistical module (e.g., frequency analysis) to provide stronger priors or constraints.
- **Architectural Modifications:** Explore SLM architectures specifically designed for symbolic reasoning or incorporating external memory.
- **Different RL Techniques:** Investigate other RL algorithms or reward shaping strategies that might provide denser or more informative feedback.

6 Discussion Points

6.1 Replicability

Our results should be reasonably replicable. We used publicly available base models (Qwen2.5-0.5B and Phi-4-mini-instruct, both accessible via Hugging Face) and standard libraries for fine-tuning and evaluation (Hugging Face Transformers, ‘trl’ library including ‘GRPOTrainer’, RapidFuzz). The base dataset we built ours from ("Human vs AI") is available here on Kaggle. The core components required for replication are the specific base model architecture (either Qwen2.5-0.5B or Phi-4-mini-instruct), the script used to generate the training dataset by applying random mono-alphabetic substitutions to the "Human vs AI" text samples, the Python code utilizing the ‘trl’ library’s ‘GRPOTrainer’ and our custom reward function (which is based on ‘rapidfuzz.fuzz.ratio’), and the configuration parameters used for the

'GRPOConfig'. Key hyperparameters from our experiments include: 'num_train_epochs=2', 'per_device_train_batch_size=4', 'num_generations=4', 'logging_steps=50', and the use of 'bf16=True' for mixed precision. Distributed training was configured using DeepSpeed with 'num_processes=4' and 'zero_stage=3'. All of our code for this project, including the dataset generation code and the fine-tuning code, is publicly available here on our GitHub repository, ensuring that other researchers can reproduce our experimental setup and results.

6.2 Datasets

Our approach of programmatically generating the training dataset by applying random mono-alphabetic ciphers to a large, existing text corpus ("Human vs AI") addresses the need for a large volume of paired encrypted and unencrypted text examples. This method is valuable because creating a sufficiently diverse and extensive dataset of this specific type is crucial for training models for cryptographic tasks. By using the "Human vs AI" dataset, known for its variety of text styles and topics, we aimed to train a model that could handle different linguistic patterns. Our decision to generate the data ourselves, rather than using the more complex "Encrypted Text: Mono-Alphabetic Cipher" dataset from Kaggle that included encryption of punctuation and symbols as well, was motivated by the need to focus the model specifically on letter substitutions, which is the core challenge of mono-alphabetic ciphers on standard English text. This programmatic dataset generation method provides a clear blueprint that other researchers working on similar symbolic manipulation or classical cryptanalysis tasks with machine learning could adopt, highlighting the feasibility of creating tailored datasets without requiring large, pre-existing encrypted corpora. It also underscores the importance of carefully defining the scope and characteristics of the target cipher and text when preparing training data.

6.3 Ethics

While mono-alphabetic substitution ciphers are not used for serious security purposes today, the development of AI tools capable of cryptanalysis, even for simple ciphers, warrants ethical consideration. The primary potential harm or risk lies in the misuse of such tools. Although unlikely to break modern encryption, a sophisticated and widely accessible tool for solving simple ciphers could potentially be used maliciously in specific, limited contexts, such as rapidly solving puzzles, challenges, or analyzing historical or niche communications that might still employ weak substitution methods. Furthermore, any research in automated code or system breaking, even on seemingly harmless targets, contributes to the broader landscape of AI-driven security analysis, which necessitates a commitment to responsible disclosure and preventing harmful applications. The "Human vs AI" dataset, while convenient, may also contain inherent biases present in its source material, although the impact of semantic bias on this specific task (focused on symbolic manipulation) is likely minimal compared to tasks relying on language understanding. To address these ethical considerations, it is crucial to clearly state the limitations of the developed tool, emphasizing that it is designed for simple, historical ciphers and is not capable of breaking modern cryptographic systems; focus the research on academic exploration of AI capabilities in symbolic reasoning and pattern recognition, rather than promoting it as a practical security-breaking tool; and engage in open discussion about the potential dual-use nature of cryptanalysis research and advocate for responsible AI development practices. Given the simplicity of the target cipher, the immediate societal risk posed by this specific project is low, but the ethical principles discussed are important for the broader field of AI applied to security and cryptanalysis.

6.4 Limitations and Future Work

Despite improvements from GRPO, our current fine-tuned SLM still exhibits significant limitations that constrain its effectiveness on mono-alphabetic substitution ciphers. The accuracy ceiling means the performance achieved, while improved over baseline SLMs, remains well below perfect decryption and significantly lags behind the best-performing large model (Claude 3.7 Sonnet) on this specific task, suggesting inherent limitations in the SLM's capacity for the complex pattern matching and consistent symbolic mapping required. Generalization is limited as the model was trained exclusively on mono-alphabetic substitution ciphers derived from the "Human vs AI" text corpus; its ability to generalize to texts with significantly different linguistic styles, shorter or much longer passages, or variations of the substitution cipher (including non-alphabetic characters or capitalization)

is limited without further specific training. Generalization to more complex classical ciphers, such as polyalphabetic ciphers, is highly unlikely with the current approach. Scalability to complexity is also an issue, as the approach appears to struggle with longer texts, where maintaining consistent substitutions across thousands of characters becomes more challenging.

Building upon this work, several promising future research directions emerge. One avenue is exploring alternative SLM architectures, investigating whether models specifically designed for symbolic reasoning or incorporating external memory mechanisms might be better suited for precise symbol mapping tasks than general-purpose causal language models. Another important direction is to systematically investigate the utility of supervised learning techniques, perhaps using the generated encrypted/decrypted pairs for standard fine-tuning or as a pre-training step before applying reinforcement learning, which could help the model learn basic substitution patterns more efficiently compared to purely relying on sparse rewards from RL. Supervised learning could also serve as a valuable baseline for comparison against RL methods. Hybrid approaches that combine the strengths of neural models (potentially trained with supervised learning or RL) with classical statistical methods are also worth exploring. For instance, an SLM could propose candidate substitutions guided by frequency analysis from a statistical module, and the reward function could incorporate statistical likelihood alongside text similarity. Beyond GRPO, investigating other reinforcement learning algorithms or more sophisticated reward shaping strategies could provide more dense and informative feedback to the model during training, particularly in the early stages when outputs are highly random. Analyzing SOTA LLM success is important to conduct a detailed analysis of why models like Claude 3.7 Sonnet perform exceptionally well on this task. Understanding if it's purely scale, specific architectural features, novel training data, or a combination could inform the development of more capable SLMs for symbolic tasks. Applying the methodology to other classical ciphers could adapt the dataset generation and fine-tuning methodology to other simple classical ciphers (such as Caesar, Simple Transposition, etc.) to test the adaptability and limitations of the approach across different types of cryptographic transformations. Finally, refining the reward function could involve experimenting with variations of the RapidFuzz-based reward function, potentially incorporating penalties for inconsistent substitutions detected within the output, to provide a stronger signal for correct character mapping.

7 Conclusion

This work explored the application of fine-tuned Small Language Models (SLMs) such as Qwen2.5-0.5B and Phi-4-mini-instruct trained with Group Relative Policy Optimization (GRPO), to the task of decrypting mono-alphabetic substitution ciphers. We demonstrated that targeted fine-tuning using GRPO can improve the performance of SLMs on this task compared to standard supervised methods, offering a pathway towards creating computationally efficient cryptographic tools. However, our results also highlight the significant challenges SLMs face with precise, rule-based symbolic manipulation and complex reasoning, as evidenced by the performance gap compared to state-of-the-art large models like Claude 3.7 Sonnet and persistent failure modes. Our investigation contributes to understanding the trade-offs between model size, training methodology, and task complexity, suggesting that while SLMs hold promise for specialized applications, overcoming their inherent limitations for certain reasoning tasks remains an open research question. Future work involving curriculum learning, hybrid approaches, and architectural exploration may further unlock the potential of SLMs in cryptographic and symbolic domains.

References

Omar G. Abood and Shawkat K. Guirguis. 2018. A survey on cryptography algorithms. *International Journal of Scientific and Research Publications (IJSRP)*, 8(7).

Heewon Baek, Minwook Lee, and Hyoungshick Kim. 2024. Cryptollm: Harnessing the power of llms to detect cryptographic api misuse. In *Computer Security – ESORICS 2024*, pages 353–373, Cham. Springer Nature Switzerland.

Jian Chen and Jeffrey S. Rosenthal. 2012. Decrypting classical cipher text using markov chain monte carlo. *Statistics and Computing*, 22(2):397–413.

Serguei A. Mokhov. 2010. Cryptolysis: A framework for verification of optimization heuristics for the automated cryptanalysis of classical ciphers and natural language word segmentation. In *2010 Eighth ACIS International Conference on Software Engineering*

Research, Management and Applications, pages 295–302.

Xiuwei Shang, Guoqiang Chen, Shaoyin Cheng, Yanming Zhang, Weiming Zhang, and Nenghai Yu. 2024. Foc: Figure out the cryptographic functions in stripped binaries with llms.

Wenquan Zhou, An Wang, Yaoling Ding, Congming Wei, Jingqi Zhang, and Liehuang Zhu. 2024. One solves all: Exploring ChatGPT's capabilities for fully automated simple power analysis on cryptosystems. Cryptology ePrint Archive, Paper 2024/2069.