# Evaluating Large Language Models on Aristocrat Cipher Decryption

An Experimental Study of Explanation, Solving Ability, and Instruction Following.

Rabeya Hamood Khawaja

## Introduction

The purpose of this experiment was to explore how large language models (LLMs) respond to classical cipher challenges, focusing specifically on the Aristocrat cipher. Rather than testing advanced cryptographic strength, the study aimed to see whether LLMs could recognize the task, follow instructions, and produce accurate or useful decryptions. To compare performance, the experiment was conducted using two widely available LLMs: **ChatGPT** and **Claude**. By presenting the same ciphertexts and prompts to both models, we were able to observe their problem-solving behavior, consistency, and ability to adapt to structured reasoning tasks.

## Tools and Approach - ChatGPT

For this experiment, **ChatGPT (OpenAI, GPT-5)** was used as one of the test models. Interaction took place through the standard chat interface, with prompts carefully structured to ensure fairness and consistency across test cases. The approach for ChatGPT involved:

1. **Consistent Prompting** - Each ciphertext was paired with a fixed instruction set, such as "Decrypt this Aristocrat ciphertext and provide only the plaintext." Prompts were kept uniform across runs to minimize variation introduced by wording.

2. **Task Variety** - ChatGPT was asked to perform multiple related tasks, including explaining the Aristocrat cipher, solving guided examples step-by-step, attempting blind decryption without a key, and returning a full substitution key. This allowed us to evaluate both reasoning and mechanical accuracy.

3. **Instruction Following** - Special attention was given to whether ChatGPT followed strict formatting rules (e.g., returning only plaintext, or providing a key in a specified format) since this is crucial for evaluating reliability in controlled experiments.

4. **Output Evaluation** - Responses were collected for accuracy (exact match with ground-truth plaintext), partial correctness (word and character accuracy), and reasoning quality (clarity of explanations and method).

This structured setup allowed for a systematic assessment of ChatGPT's ability to handle Aristocrat cipher challenges in both explanation and decryption tasks.

## Tools and Approach – Claude Sonnet 4.5

For this experiment, **Claude Sonnet 4.5 (Anthropic)** was used as the other coding collaborator. Interaction took place through the claude.ai web interface, with an iterative, conversation-driven development style that embodied true "vibe coding." The approach for Claude involved:

1. **Exploratory Dialogue** – Rather than providing rigid specifications upfront, I shared the complete assignment document and engaged in open-ended conversation. Initial prompts like "provide the python script for the problem defined in this" allowed Claude to interpret requirements freely and propose solutions organically. This conversational flow continued through multiple iterations, with each exchange building on previous attempts.
2. **Iterative Refinement Through Feedback** – Claude's development process was highly responsive to critique. When solutions failed (e.g., "I am not getting the answer to the decipher"), Claude didn't just debug - it fundamentally reconsidered the approach. The code evolved through four major iterations: (1) over-engineered multi-algorithm suite, (2) enhanced scoring systems, (3) dictionary-based constraint solver, and (4) simplified simulated annealing. Each iteration represented a complete architectural shift based on conversational feedback.
3. **Algorithm Implementation Focus** - Claude specialized in producing working code artifacts rather than just explanations. The final deliverable was a complete Python script (~200 lines) implementing simulated annealing with: random key initialization, English text scoring (common words + bigrams), temperature-based acceptance criteria, and 50,000 iteration optimization. The code included proper type hints, docstrings, both automatic and interactive modes, and real-time progress reporting.
4. **Artifact-Based Development** - Claude leveraged its artifact system to maintain living documents of code that could be directly downloaded and executed. Each iteration produced a complete, runnable script rather than snippets. This allowed for immediate testing and provided clear snapshots of the evolution process for the reproducibility bundle.
5. **Self-Correction and Simplification** - A notable pattern emerged where Claude initially over-complicated solutions (adding genetic algorithms, quadgram analysis, backtracking search) before recognizing through feedback that simpler approaches were more effective. The final solution deliberately used only Python standard library (no external dependencies) and a single proven algorithm, demonstrating that Claude could pivot from complexity to elegance when guided.
6. **Output Evaluation** - Success was measured by: (1) whether the code executed without errors, (2) whether it correctly decrypted the sample Aristocrat cipher, (3) code readability and maintainability, (4) reproducibility (no special dependencies), and (5) usefulness of intermediate outputs (progress updates, scoring feedback). The final implementation successfully cracked substitution ciphers within 1-2 minutes of runtime.

This unstructured, collaborative setup allowed for authentic assessment of Claude's ability to function as a pair-programming partner-adapting to vague requirements, accepting critical feedback, and iteratively converging on working solutions through natural dialogue rather than formal specifications.

## Comparative Insights

Working with both ChatGPT and Claude highlighted complementary strengths and weaknesses.

- **Understanding Requirements.** ChatGPT took longer to fully grasp the specific goal of "a Python program that can directly decode the hardcoded ciphertext." Early responses tended to offer general cryptanalysis strategies or explanations rather than a concrete solution. Claude, on the other hand, more quickly pointed toward the correct direction of implementing a working script, though sometimes by overcomplicating the design.

- **Specificity vs. Over-Engineering.** ChatGPT's answers were more specific and structured once the requirement was clarified - for example, providing focused explanations of quadgrams, simulated annealing, and scoring functions. Claude often introduced extra complexity (e.g., unnecessary abstractions or broader generalizations) that could obscure the main task.

- **Coding Ability.** Claude was stronger in directly generating runnable Python code snippets with fewer syntax errors. ChatGPT sometimes required iteration and corrections before producing code that executed properly.

- **Conversational Flow.** ChatGPT was better at maintaining a collaborative "conversation," adapting explanations, and giving step-by-step refinements. Claude was less conversational and more task-oriented, which sometimes made it less flexible during the debugging process.

In short, Claude was the stronger "coder" while ChatGPT was the stronger "collaborator." Their differences suggest that pairing them - one for generating raw code, the other for refining and contextualizing - could yield even better outcomes in cryptanalysis projects.

## Reflections

Several aspects of the interaction process were surprising. First, the level of persistence required to get a direct decoding script was higher than expected. Even though both models are advanced, they often defaulted to *explaining* cryptanalysis methods (quadgrams, frequency analysis, simulated annealing) rather than immediately applying them to the hardcoded ciphertext. This revealed a gap between conversational knowledge and applied problem-solving.

Another surprise was how differently the models handled ambiguity. Claude tended to "jump ahead," producing runnable code more quickly, but often introduced unnecessary layers of abstraction. ChatGPT, by contrast, hesitated at first, but once the requirement was pinned down, it became very systematic and precise. This showed that model personalities can diverge: one behaving like a quick but opinionated coder, the other like a patient but sometimes slower tutor.

It was also unexpected how much trial and error was needed when guiding the models toward the *exact* format of output (hardcoded ciphertext + plaintext decoding). Neither model “got it right the first time.” This reinforced the importance of human steering - AI is not yet a drop-in replacement for careful programming but rather a collaborative assistant.

Finally, an ethical and reproducibility consideration emerged: because both models generated different scripts (with varying scoring methods and heuristics), another researcher might get different results unless code, prompts, and settings are fully documented. Ensuring transparency in the workflow is as important as the cryptanalysis itself.