CipherBank: Exploring the Boundary of LLM Reasoning Capabilities through Cryptography Challenges

Yu Li[1], Qizhi Pei[1,2], Mengyuan Sun[1], Honglin Lin[1],
Chenlin Ming[1,3], Xin Gao[1], Jiang Wu[1], Conghui He[1], Lijun Wu[1]*
[1]Shanghai Artificial Intelligence Laboratory
[2]Renmin University of China [3]Shanghai Jiao Tong University
{liyu1,heconghui,wulijun}@pjlab.org.cn
https://cipherbankeva.github.io

Abstract

Large language models (LLMs) have demonstrated remarkable capabilities, especially the recent advancements in reasoning, such as o1 and o3, pushing the boundaries of AI. Despite these impressive achievements in mathematics and coding, the reasoning abilities of LLMs in domains requiring cryptographic expertise remain underexplored. In this paper, we introduce **CipherBank**, a comprehensive benchmark designed to evaluate the reasoning capabilities of LLMs in cryptographic decryption tasks. CipherBank comprises **2,358** meticulously crafted problems, covering **262 unique plaintexts** across **5 domains** and **14 subdomains**, with a focus on privacy-sensitive and real-world scenarios that necessitate encryption. From a cryptographic perspective, CipherBank incorporates **3 major categories** of encryption methods, spanning **9 distinct algorithms**, ranging from classical ciphers to custom cryptographic techniques. We evaluate state-of-the-art LLMs on CipherBank, e.g., GPT-4o, DeepSeek-V3, and cutting-edge reasoning-focused models such as o1 and DeepSeek-R1. Our results reveal significant gaps in reasoning abilities not only between general-purpose chat LLMs and reasoning-focused LLMs but also in the performance of current reasoning-focused models when applied to classical cryptographic decryption tasks, highlighting the challenges these models face in understanding and manipulating encrypted data. Through detailed analysis and error investigations, we provide several key observations that shed light on the limitations and potential improvement areas for LLMs in cryptographic reasoning. These findings underscore the need for continuous advancements in LLM reasoning capabilities.

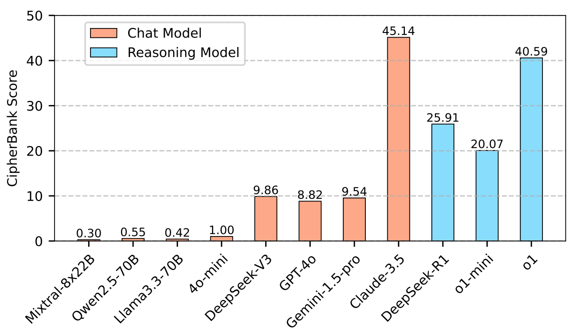


Figure 1: Comprehensive Performance of SOTA Chat and Reasoning Models on CipherBank.

1 Introduction

Large Language Models (LLMs) have revolutionized artificial intelligence by achieving state-of-the-art (SOTA) performance across diverse domains, from Natural Language Understanding (NLP) (Dong et al., 2019; Karanikolas et al., 2023; Sasaki et al., 2024) to complex problem-solving (Yao et al., 2024; Ge et al., 2023). Recent models, such as GPT-4o (Hurst et al., 2024) and Claude 3.5 (Anthropic, 2024), have demonstrated unprecedented versatility, excelling in tasks ranging from creative writing to technical analysis. A particularly notable advancement lies in the reasoning-enhanced LLMs, which have emerged as a critical benchmark for evaluating LLMs' intelligence and now can solve mathematical problems (Wu et al., 2024; Ahn et al., 2024; Liu et al., 2024c), debug intricate code (Lee et al., 2024; Zhong et al., 2024), and even engage in multi-step logical deduction (Sun et al., 2024; Wang et al., 2023) with human-like proficiency. For instance, specialized architectures like o1 (Jaech et al., 2024) and DeepSeek-R1 (Guo et al., 2025) have pushed the boundaries of AI reasoning, achieving breakthroughs in domains such as theorem proving (Yang et al., 2024b) and algorithmic optimization (Liu et al., 2024b). These achievements underscore the transformative potential of LLMs as general-purpose reasoning engines, capable of adapting to both broad and specialized challenges.

* Corresponding author

To quantify progress, the community has proposed numerous benchmarks targeting mathematical reasoning (e.g., MATH (Hendrycks et al., 2021a), AIME[1], coding proficiency (e.g., HumanEval (Chen et al., 2021a), MBPP (Austin et al., 2021)), and general logical deduction (e.g., FOLIO (Han et al., 2024), MMBench (Yuan Liu, 2023), CaLM (Chen et al., 2024). These testbeds have become indispensable tools for assessing model capabilities.

Despite extensive evaluations in mathematics and coding, one critical domain remains underexplored: cryptographic decryption. Cryptographic reasoning (Shree et al., 2017) demands unique capabilities, including pattern recognition, algorithmic Reverse-engineering, and contextual understanding of security constraints (Schneier, 2002)—skills distinct from those tested in conventional benchmarks. This gap is particularly consequential, as cryptography lies at the heart of modern digital security (Konheim, 2007), with applications spanning privacy-preserving communication (Soomro et al., 2019), secure authentication (Rani et al., 2022), and data integrity (Sarkar et al., 2021). The absence of a rigorous benchmark for cryptographic reasoning not only limits the true understanding of LLM's reasoning ability but also hinders progress toward AI systems capable of contributing to security-critical contexts (e.g., jailbreaking (Wei et al., 2024)). OpenAI has scratched the surface of this challenge and put a demo[2] when releasing their strong reasoning model o1, but no serious efforts have been made to reveal this challenge in the committee.

To address this gap, we introduce CipherBank, the first comprehensive benchmark specially designed to evaluate LLMs' reasoning capabilities in cryptographic decryption tasks. CipherBank is meticulously constructed to reflect real-world scenarios requiring encryption, instead of general texts that may serve as a toy testbed, with 2,358 problems derived from 262 unique plaintexts across 5 domains (e.g., Personal Privacy, Financial Information) and 14 subdomains (e.g., Identity Information, Personal Income). As for cipher algorithms, it spans 3 major cryptographic categories—Substitution Ciphers (e.g., Rot13, Vigenère), Transposition Ciphers (e.g., Reverse, SwapPairs), and custom hybrid algorithms—encompassing 9 distinct encryption methods, covering **5** difficulty levels (from Basic to Expert) to ensure a diverse range of challenges. By integrating privacy-sensitive contexts and multi-layered cryptographic challenges, CipherBank provides a nuanced evaluation framework that captures both the complexity and practicality of real-world decryption tasks.

We evaluate CipherBank on SOTA LLMs, including general-purpose models (GPT-4o (Hurst et al., 2024), DeepSeek-V3 (Liu et al., 2024a)) and reasoning-optimized models (o1 (Jaech et al., 2024), DeepSeek-R1 (Guo et al., 2025)). Results reveal striking limitations: even advanced models struggle with classical ciphers, achieving only 45.14 score on tasks solvable by human cryptanalysts. Notably, we observe a significant performance gap between general chat LLMs and specialized reasoning models, suggesting that current reasoning optimizations inadequately address cryptographic challenges. Besides, we also provide studies on different aspects for deep understandings, such as evaluate on noised plaintexts and different length of plaintexts. Observations show the limitations of current models in decryption reasoning, with chat and reasoning models each exhibiting distinct strengths and weaknesses in cryptographic tasks. These findings highlight the need for targeted improvements in LLMs' cryptographic reasoning, with implications for both AI safety (e.g., adversarial robustness) and applications in cybersecurity.

2 CipherBank Construction

CipherBank is a purpose-built benchmark designed to rigorously evaluate the reasoning capabilities of LLMs in cryptographic decryption tasks. It integrates three core components to ensure comprehensive coverage of real-world scenarios and cryptographic complexity: (1) **diverse plaintexts** meticulously constructed from multiple dimensions of real-world privacy-sensitive data, ensuring the decryption process aligns with practical requirements; (2) **a comprehensive suite of encryption algorithms**, including both traditional cryptographic methods and custom-designed algorithms, to thoroughly assess the model's reasoning, inductive, and computational capabilities from multiple perspectives; and (3) **a structured problem set with rich metadata**, enabling granular performance analysis and detailed error analysis based on the diverse properties of the plaintexts.

[1] `https://huggingface.co/datasets/AI-MO/aimo-validation-aime`

[2] `https://openai.com/index/learning-to-reason-with-llms/`

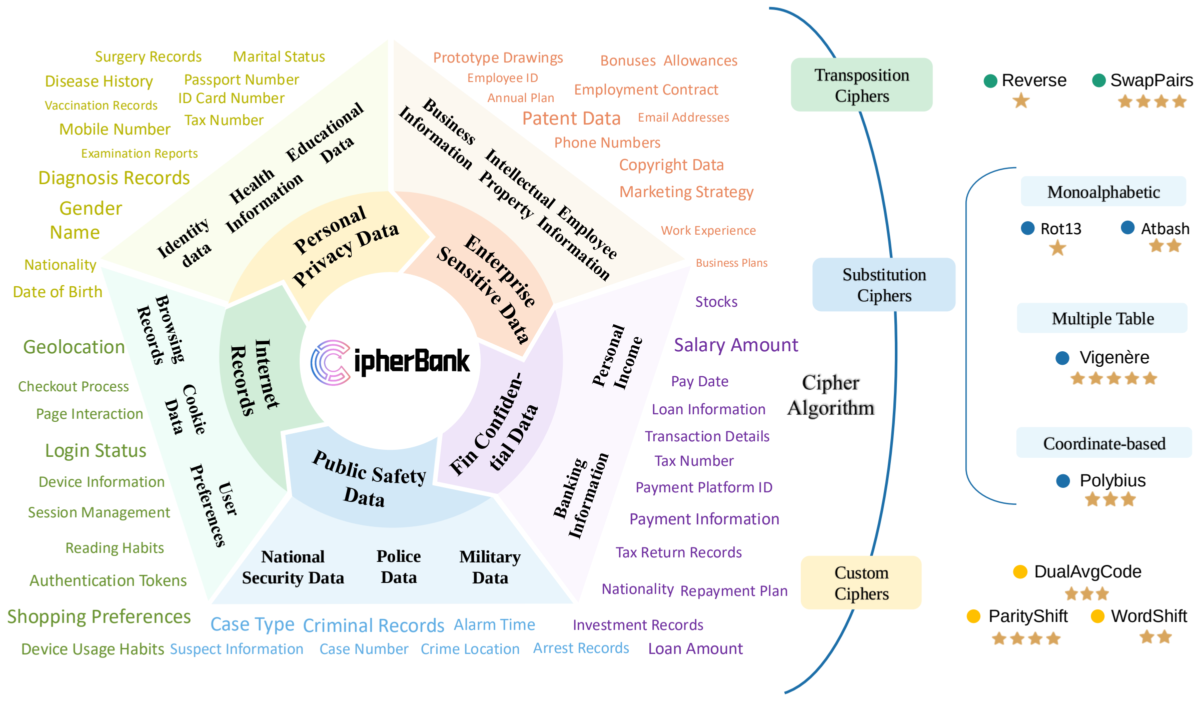


Figure 2: Overview of CipherBank. CipherBank consists of simulated privacy data encrypted using various algorithms. The left side of the figure shows five domains, 14 subdomains, and selected tags. The right side displays three encryption categories, nine specific algorithms, and their corresponding difficulty levels.

2.1 Plaintext Data: Design, Sources, and Real-World Alignment

To construct CipherBank, we meticulously analyze real-world encryption scenarios and categorize the corresponding data types into five primary domains: **Personal Privacy Data**, **Enterprise Sensitive Data**, **Public Safety Data**, **Financial Asset Data** and **Internet Records**. These domains are further refined into **14** subdomains (e.g., Health Information, Policy Data) to ensure comprehensive coverage of encryption needs. Inspired by UltraChat (Ding et al., 2023), we adopt a **tag-based approach** to systematically structure encryption-relevant data, ensuring semantic consistency and domain relevance. Below, we detail the **3-step** process for generating high-quality plaintext data.

Step 1: Tag Definition and Curation. We leverage GPT-4o to generate candidate tags for each subdomain, capturing diverse real-world encryption scenarios. Human experts then curate these tags, eliminating redundancies, irrelevancies, and ambiguous entries, resulting in **89** distinct tags (see Appendix A.1). This structured approach ensures that the generated plaintext data remains realistic, contextually meaningful, and representative of actual encryption use cases. The tags are designed to align with the **Variable Length property**, enabling the generation of inputs of varying sizes to assess model robustness.

Step 2: Controlled Text Generation. Our plaintext generation process employs tag combinations to control text granularity: entries with more tags contain richer contextual details and greater length, while those with fewer tags remain concise and specific. To ensure semantic validity, all generated data are filtered to eliminate generic or redundant descriptions, creating a dataset that reflects diverse encryption scenarios with varying complexity. Additionally, we introduce the **Noise Perturbation** property through controlled noise injection, which serves two key objectives: (1) testing the model's anti-interference capabilities and (2) reducing its reliance on contextual semantics to enhance robustness. Furthermore, we incorporate **Sensitive Numerical Data** by designing scenarios with complex alphanumeric combinations, including critical identifiers such as ID card and passport number. This multifaceted approach enables a comprehensive evaluation of the model's ability to address sophisticated decryption challenges.

Step 3: Expert Validation and Refinement. After generation, we conduct expert validation to ensure data quality, correctness, and relevance. Non-informative content, excessively long or short samples, and entries lacking clear privacy attributes are filtered out. Through this rigorous refinement process, we retain **262** high-quality plaintext samples. This approach enables a practical and application-driven benchmark for evaluating LLMs' decryption capabilities in cryptographic reasoning tasks.

2.2 Encryption Algorithms

CipherBank incorporates **3** major categories of encryption methods: Substitution Ciphers, Transposition Ciphers, and Custom Ciphers. **(1) Substitution-based techniques**, including Rot13, Atbash, Polybius and Vigenère, test a model's ability to decode character-level transformations. These ciphers involve monoalphabetic or polyalphabetic substitutions, where each character is replaced by another based on a fixed rule or key. These methods evaluate the model's capacity to decode symbolic mappings and generalize across substitution rules. **(2) Transposition-based techniques**, such as Reverse and SwapPair, focus on positional rearrangements rather than symbol substitutions. These ciphers challenge the model to recognize structural patterns, such as reversed sequences or pairwise swaps. Unlike substitution ciphers, which alter character identities but preserve their order, transposition ciphers preserve characters but disrupt their sequence. This tests the model's ability to analyze sequential dependencies and reconstruct the original symbol order.

To further assess LLMs' ability to decrypt uncommon encryption methods, we introduce **(3) Custom-designed ciphers** that deviate from standard cryptographic schemes. **(a) DualAvgCode** is inspired by OpenAI's o1 model showcase[3], where iterative transformations require models to infer multi-step encryption patterns. **(b) ParityShift** draws from LSB steganography (Mielikainen, 2006), a common technique in information hiding, incorporating bitwise manipulations based on character parity. **(c) WordShift** Cipher is designed to evaluate LLMs' ability to decrypt ciphers that combine substitution and transposition encryption, performing Caesar-style letter shifts within each word individually, blending character-level substitution with structural reordering.

Meanwhile, We categorize the nine algorithms into five difficulty tiers based on key necessity and computational complexity. T1 (Basic) includes simple ciphers like ROT13 and Reverse. T2 (Intermediate) introduces Atbash and WordShift with slightly more complex rules. T3 (Moderate) covers DualAvgCode and Polybius, requiring structured encoding. T4 (Advanced) involves ParityShift and SwapPairs with intricate data manipulation. T5 (Expert) features the Vigenère cipher, a polyalphabetic substitution cipher known for its keyword-based complexity. This framework organizes encryption techniques from basic to expert.

2.3 CipherBank Statistics

As shown in Figure 2, we provides an overview of CipherBank structure. The encryption algorithm in Section 2.2 applies to the expert-curated dataset from Section 2.1, yielding **2,358** test data points.

Table 1: Statistics of CipherBank.

Domains	**#Tag**	**#Plaintext**	**#Test**	**Avg(len)**
Personal Privacy Data	23	50	450	107.88
Enterprise Sensitive Data	16	52	468	103.10
Public Safety Data	17	63	567	110.89
Financial Asset Data	13	44	396	163.68
Internet Records	20	53	477	191.92
Summary	89	262	2358	134.03

Table 1 summarizes the distribution of plaintexts across 5 domains, each with varying numbers of tags, samples, and test cases. Notably, Internet Records has the longest plaintexts (191.92), while Enterprise Sensitive Data has shorter samples (103.10). This diversity ensures a comprehensive evaluation of model performance across different encryption contexts.

3 Evaluations

3.1 Evaluation Setup

Evaluation Protocols. In terms of testing methodology, CipherBank's evaluation follows the Known-Plaintext Attack framework (Zulkifli and Mohd, 2008), employing a **3-shot** testing approach. We prompt the model with three plaintext-ciphertext pairs as demonstrations to infer encryption rules, identify potential keys, and apply the learned patterns to decrypt a new ciphertext. The detailed prompt can be found in Appendix B.1.

For evaluation metrics, we primarily employ *accuracy* to measure overall decryption success, which is the ratio of correctly decrypted cases to total test cases, where correctness requires an exact character match with the plaintext. Additionally, to capture finer-grained differences between the decrypted output and the original plaintext, we incorporate *Levenshtein similarity* (Yujian and Bo,

[3]https://openai.com/index/learning-to-reason-with-llms/

Example 2.1: Plain-Ciphertext Pair

# Domain: Personal Privacy Data

## Subdomain: Identity Information

### Tag Combination: ["Name", "Date of Birth", "Passport Number"]

Plaintext:

Peter was born on April 23, 1985, and carries a passport with the number X123456789.

Encryption results:

(1) **Rot13**: *Crgre jnf obea ba Ncevy 23, 1985, naq pneevrf n cnffcbeg jvgu gur ahzore K123456789.*

(2) **SwapPairs**: *ePet raw sobnro npAir l32 ,9158 ,na dacrrei s aapssoptrw ti hht eunbmreX 21436587.9*

(3) **WordShift** : *erPet was nbor no ilApr 23, 5,198 and riescar a sportpas hwit the bernum 3456789.X12*

(4) ...

More results can be found in the appendix.

2007). We compute the Levenshtein distance for each sentence individually and report the average Levenshtein similarity across all test cases, providing a more nuanced assessment of model performance beyond binary correctness.

LLM Candidates. For a comprehensive evaluation, we carefully selected **18** SOTA LLMs for evaluation, ensuring a diverse representation of open-source, closed-source, and reasoning-specialized models. Below, we outline the tested models:

⋆ ***Open-Source Chat Models:*** We evaluate leading open-source LLMs, including Mistral AI's Mixtral-8x22B (Jiang et al., 2024a), Alibaba's Qwen2.5-72B-Instruct (Yang et al., 2024a), Meta's Llama-3.1-70B-Instruct and Llama-3.3-70B-Instruct (Dubey et al., 2024), as well as the rising star - DeepSeek-V3 (Liu et al., 2024a).

⋆ ***Closed-Source Models:*** For proprietary models, evaluation is conducted via API access. The tested models include OpenAI's 4o-mini and GPT-4o series (0806, 1120) (Hurst et al., 2024), DeepMind's Gemini-1.5-Pro (Team, 2024a) and Gemini-2.0-Flash-Exp[4], along with Anthropic's Claude-Sonnet-3.5 (1022)[5].

⋆ ***Reasoning Models:*** We further investigate models optimized for reasoning tasks, including QwQ-32B-Preview (Team, 2024b), DeepSeek-R1 (Guo et al., 2025), Gemini-2.0-Flash-Thinking (1219)[6], o1-mini (0912) and o1 (1217) (Jaech et al., 2024).

[4]https://deepmind.google/technologies/gemini/flash/

[5]https://www.anthropic.com/news/claude-3-5-sonnet

[6]https://deepmind.google/technologies/gemini/flash-thinking/

3.2 Benchmark Results

Table 2 presents the evaluation results of all candidate LLMs (Levenshtein similarity results are in Appendix C.1). Below, we distill the experimental findings into several observations:

Limitations of Current Models in Cryptographic Reasoning. Despite advancements in LLMs, Table 2 highlights their limitations in structured cryptographic reasoning. The overall performance remains low, with most SOTA models struggling to achieve meaningful accuracy. In Cipher Score, common models like Qwen and LLaMA perform particularly poorly, with some scoring in the single digits or near zero. Even the best-performing models, Claude-3.5 and o1, achieve less than 50 in accuracy, underscoring the significant difficulty of CipherBank and the challenges LLMs face in systematic decryption.

Reasoning Models Generally Outperform Chat Models. When comparing reasoning models to chat models, generally we can find that the reasoning models do outperform chat models on all cipher algorithms and achieve better overall performance. The only expectation is the superior performance of Claude-3.5 (45.14) even better than o1, and also the bad performance of QwQ-32B-Preview (only 0.76 accuracy). This clearly demonstrate the advantages of the reasoning-specialized models.

Closed-Source Models Retain an Edge Over Open-Source Models. Overall, closed-source models outperform open-source models in cryptographic decryption. Claude-3.5 (45.14) and o1 (40.59) achieve the highest performance across all cipher categories. However, DeepSeek-V3 (9.86) and DeepSeek-R1 (25.91) surpass most models in the GPT and Gemini families, indicating that advanced open-source models are closing the gap.

Table 2: 3-shot scores (%) of LLMs across three major encryption paradigms and nine specific encryption algorithms on CipherBank. The highest scores in each category are highlighted with a **blue background**, while the second-best results are <u>underlined</u> for emphasis.

Model	Substitution Ciphers				Transposition Ciphers		Custom Ciphers			Cipher Score
	Rot	Atbash	Polybius	Vigenère	Reverse	SwapPairs	DualAvgCode	ParityShift	WordShift	Accuracy$_{avg}$
Open-source Chat Models										
Mixtral-8x22B-v0.1	0.38	0	0	0	0.76	0	0.38	0	1.15	0.30
Qwen2.5-72B-Instruct	1.15	0	0	0	0	0.38	1.15	0	2.29	0.55
Llama-3.1-70B-Instruct	1.15	0.38	0	0.38	0	0	0.38	0.38	0.76	0.38
Llama-3.3-70B-Instruct	2.67	0.38	0	0	0	0	0	0.76	0	0.42
DeepSeek-V3	32.44	14.88	2.29	0.76	28.47	0.38	0.38	1.14	8.02	9.86
Closed-source Models										
GPT-4o-mini-2024-07-18	3.69	2.03	0	0.51	2.16	0	0.38	0	0.25	1.00
GPT-4o-2024-08-06	38.17	3.05	0.38	0.76	25.19	2.29	0	1.14	8.40	8.82
GPT-4o-2024-11-20	26.46	6.99	0.13	0.76	15.27	0.76	0.25	0.89	6.11	6.40
gemini-1.5-pro	55.34	0.76	0.38	0.76	10.31	0.76	0.38	0.76	16.41	9.54
gemini-2.0-flash-exp	35.88	3.05	1.53	0.38	<u>29.39</u>	1.53	0	0.76	5.34	8.65
Claude-Sonnet-3.5-1022	**83.21**	<u>75.19</u>	<u>72.90</u>	<u>1.91</u>	**63.93**	<u>6.87</u>	4.96	**58.21**	**39.12**	**45.14**
Reasoning Models										
QwQ-32B-Preview	1.53	0.38	1.91	0	0	0	0.38	0.38	2.29	0.76
DeepSeek-R1	<u>73.28</u>	58.78	44.27	0.38	10.69	0.38	<u>24.05</u>	<u>12.98</u>	8.40	25.91
gemini-2.0-flash-thinking	40.46	17.18	21.76	1.15	22.90	1.15	0	7.63	9.16	13.49
o1-mini-2024-09-12	46.18	68.32	46.95	1.53	5.15	0.38	2.93	7.63	1.53	20.07
o1-2024-12-17	59.92	**79.01**	**79.39**	**7.25**	14.89	**32.14**	**50.38**	12.39	<u>29.90</u>	<u>40.59</u>

Nevertheless, both still lag behind Claude-3.5 and o1, suggesting that while open-source models are improving, there is significant potential for open-source models to achieve even better performance in the future.

The performance variance among models of the same category is remarkably significant. Within the Open-source Chat Models category, the top-performing model, deepseek-v3 (9.86), outperforms the weakest model, Mixtral-8x22B (0.30), by a factor of 33. Similarly, in the Closed-source Models category, Claude-Sonnet-3.5 (45.14) demonstrates a performance 45 times greater than that of GPT-4o-mini (1.00). The disparity is even more pronounced in the Reasoning Models category, where o1 (40.59) surpasses QwQ-32B-Preview (0.76) by a factor of 53. Such substantial performance variations are rarely observed in other benchmarks, highlighting the challenging nature of CipherBank. This benchmark effectively distinguishes the reasoning capabilities of different models through its decryption dimension, providing a robust framework for evaluating model performance.

4 Detailed Analysis

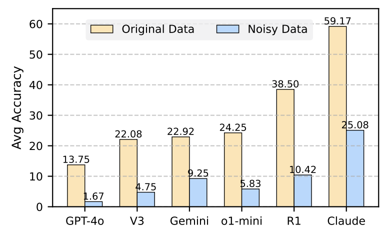
In this section, we conduct a detailed analysis from the perspectives of plaintext characteristics, noise levels, testing methodologies, finer-grained evaluation metrics, and error analysis to gain deeper insights into the strengths and limitations of different LLMs in cryptographic decryption.

Table 3: Model Performance on Short and Long Plaintext Setting (Lower Difference and Decrease Ratio Are Better). We highlight the most stable and sensitive results in blue and green respectively.

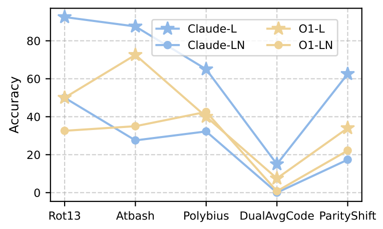
Model	Short	Long	Diff	Decrease Ratio(%)
GPT-4o-2024-11-20	9.47	4.46	5.01	52.60
gemini-2.0-flash-exp	11.50	6.42	5.08	44.35
DeepSeek-V3	13.24	5.22	8.02	60.60
gemini-2.0-flash-thinking	19.90	8.47	11.43	42.61
DeepSeek-R1	32.27	20.94	11.33	33.16
o1-mini-2024-09-12	33.77	17.35	16.42	48.57
o1-2024-12-17	47.61	34.38	13.23	27.78
Claude-Sonnet-3.5	48.70	47.85	0.85	1.74

4.1 Impact of Plaintext Length

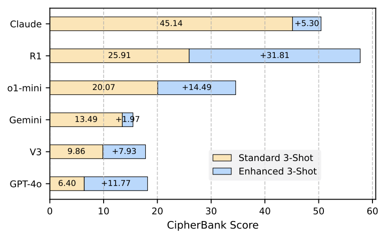
To test models' sensitivity to text length, we categorize plaintexts into short (fewer than three tags) and long groups, averaging 70.29 and 181.61 characters, respectively. As shown in Table 3 (full results and plaintext examples can be found in Appendix C.2), longer plaintexts lead to a significant performance decline in most models. Most models exhibit a significant decline in decryption performance as text length increases. Among them, Claude-3.5 (-0.85) shows the most stable performance, while o1-mini (-16.42) is the most sensitive. This contrasts with human performance, highlighting LLMs' **length bias** in decryption reasoning.



(a) Model Robustness to Noisy Inputs: Performance Comparison.



(b) Effect of Encryption Scope: Letters Only vs. Letters & Numbers.



(c) Evaluating the Benefit of Explicit Algorithm Hints in 3-Shot Prompting.

Figure 3: Evaluation of LLM Performance Under Different Encryption and Prompting Conditions.

4.2 Effect of Noise on Model Robustness

We observe that models frequently substituted synonyms instead of strictly applying decryption rules to each character (examples in Appendix C.2), indicating the presence of **shortcut reasoning**, where models partially decrypt the text and infer the remainder based on semantic context rather than adhering to the encryption pattern.

To evaluate robustness and mitigate reliance on semantic inference, we select the 40 plaintexts with the lowest **perplexity** (PPL) scores, computed using Llama-3.1-8B-Instruct, for noise injection. Figure 3a shows a substantial performance drop across all models, including Claude-3.5 (from 59.17 to 25.08) and o1-mini (from 24.25 to 5.83), highlighting their vulnerability to structural perturbations and further exposing the limitations of current models in systematic reasoning and precise decryption.

4.3 Effect of Encryption Scope

In previous evaluations, only letters are encrypted. To better reflect real-world scenarios, here we select plaintexts with sensitive numerical data and apply encryption to both letters and numbers, focusing on algorithms that directly affect numbers (test prompt in Appendix C.2). As shown in Table 3b, model performance drops significantly in this more complex setting. This suggests difficulty in adapting decryption strategies to numerical transformations. Even under the same encryption principles, encrypting both letters and numbers greatly increases task complexity, posing a significant challenge for current reasoning models. This highlights a critical limitation in LLMs' ability to generalize across diverse data types, particularly when numerical transformations are involved. Future work should focus on enhancing models' capacity to handle mixed data encryption.

4.4 Effect of Explicit Algorithm Hints on Decryption Performance

Previous evaluations highlight the significant challenges posed by CipherBank. To evaluate the models' decryption capabilities when provided with algorithm details, we enhance the 3-shot setting by explicitly informing the models of the specific algorithm during testing. Under the revised setting, models are no longer required to independently deduce encryption logic but instead focus on identifying the necessary key and applying the specified decryption rules. The enhanced prompt is provided in Appendix C.2. Table 3c reveals distinct performance patterns. Most chat models show minimal improvement even with algorithm details, struggling with key inference and decryption—highlighting persistent limitations, especially in models like Claude (+5.30) and Gemini (+1.97).

In contrast, reasoning models show marked performance gains, with R1 (+31.81) and o1-mini (+14.49) achieving significant improvements. The observed contrast underscores a fundamental distinction: chat models primarily rely on surface-level pattern recognition, while reasoning models excel in structured inference when provided with appropriate guidance.

4.5 Error Analysis

We conduct a comprehensive error analysis based on the test results in Table 2, identifying six distinct error types. To gain deeper insights, we examine the three best-performing chat models and three best-performing reasoning models, summarizing their error distributions. Detailed error definitions and examples are provided in Appendix D.1 and D.2.

As shown in Figure 4, the distribution of error types reveals key differences between reasoning and chat models. Surprisingly, (1) reasoning models exhibit a higher rate of reasoning failures than

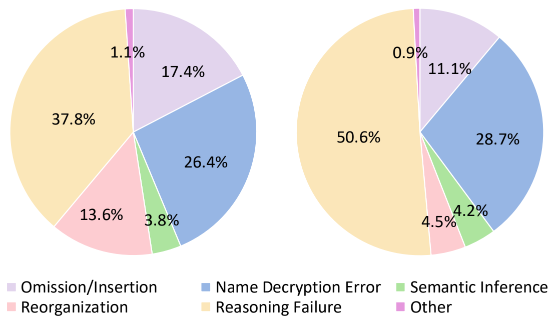


Figure 4: Decryption Error Distribution. The left represents chat models, while the right corresponds to reasoning models.

chat models. A deeper examination of Appendix D.3 reveals that many of these failures occur on simpler tasks, suggesting that reasoning models may overanalyze problems, leading to incorrect conclusions. This indicates that their complex inference processes can sometimes hinder performance on straightforward decryption cases. Conversely, (2) chat models show a higher frequency of omission/insertion and reorganization errors, indicating that while they are stronger in semantic understanding, this often results in excessive auto-completion and sentence restructuring rather than strict rule adherence. This tendency suggests that chat models prioritize fluency over exact decryption, leading to unintended modifications. Additionally, (3) both model types frequently make errors in name decryption, highlighting a broader challenge in handling structured entity transformations. This suggests that current LLMs struggle to consistently apply encryption rules to proper nouns, potentially due to memorization biases or difficulties in preserving entity-level consistency during decryption.

5 Related Work

Benchmarks for Reasoning Evaluating reasoning abilities in LLMs has been a key focus in AI research, with various benchmarks assessing models across mathematical, logical, and inferential tasks. MATH (Hendrycks et al., 2021b), MathBench (Liu et al., 2024c), and LiveMathBench (Liu et al., 2024d) test arithmetic and algebraic reasoning, while HumanEval (Chen et al., 2021b), DebugBench (Tian et al., 2024) and BigCodeBench (Zhuo et al., 2024) evaluates code generation that require programming logic. Additionally, BIG-Bench (Srivastava et al., 2022), BBH (Suzgun et al., 2022), and LiveBench (White et al., 2024) measure broader cognitive abilities, such as abstract reasoning and analogical problem-solving. KOR-Bench (Ma et al., 2024) is new benchmark that examines strong reasoning by introducing Knowledge-Orthogonal Reasoning (KOR) tasks, assessing models' ability to apply newly introduced rules independent of pretrained knowledge. Specially, it also contains a cipher reasoning task, which provides explicit encryption rules and keys, guiding models through step-by-step decryption rather than requiring pattern inference. In contrast, CipherBank presents a more realistic challenge, requiring models to identify encryption patterns from examples without prior knowledge, better reflecting real-world scenarios where encryption schemes are unknown.

Jailbreaking via Cipher Characters Recent work demonstrates that encoding adversarial prompts via encryption (Yuan et al., 2023; Wei et al., 2024) or obfuscation (Yong et al., 2023; Jiang et al., 2024b; Kang et al., 2024) can bypass LLM safety filters by exploiting models' ability to process encoded inputs. While CipherBench (Handa et al., 2024) evaluates cipher-based jailbreaking, its reliance on 40 curated plaintexts and explicit algorithm hints limits practical relevance. Our CipherBank removes prior guidance, requiring autonomous pattern inference from plaintext-ciphertext pairs to simulate privacy-sensitive decryption scenarios, establishing a robust benchmark for LLM security evaluation.

6 Conclusion

In this work, we introduce CipherBank, a comprehensive benchmark for evaluating reasoning capabilities through cryptographic decryption. CipherBank includes 5 domains, 14 subdomains of plaintext data, 9 encryption algorithms, and 2,358 decryption tasks. By testing SOTA LLMs on CipherBank, we uncover significant limitations in their decryption abilities, revealing distinct strengths and weaknesses between reasoning and chat models. Our analysis identifies key deficiencies in current reasoning approaches and suggests directions for improvement, positioning CipherBank as a novel benchmark for advancing structured inference and cryptographic reasoning in developing future LLMs.

Limitations

Our evaluation is constrained by the reliance on closed-source models, which are accessible only via API calls. This introduces potential variability due to API updates and version changes, though we mitigate this by documenting the specific versions and dates used. Additionally, access restrictions prevent us from evaluating more advanced models such as o1 Pro and o3 series, limiting the scope of our benchmark. From a design perspective, CipherBank primarily focuses on classical encryption algorithms, as modern cryptographic schemes introduce complexities beyond current model capabilities. While this choice ensures feasibility in evaluation, it also restricts the benchmark's applicability to real-world cryptographic challenges. As models improve, expanding CipherBank to modern encryption techniques will provide a more comprehensive assessment of reasoning in cryptographic tasks.

Acknowledgements

This work is supported by National Key R&D Program of China (2022ZD0160201).

References

Janice Ahn, Rishu Verma, Renze Lou, Di Liu, Rui Zhang, and Wenpeng Yin. 2024. Large language models for mathematical reasoning: Progresses and challenges. arXiv preprint arXiv:2402.00157.

Anthropic. 2024. Claude 3.5 sonnet. `https://www.anthropic.com/news/claude-3-5-sonnet`. Accessed: 2025-02-09.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. 2021. Program synthesis with large language models. Preprint, arXiv:2108.07732.

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. 2021a. Evaluating large language models trained on code.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. 2021b. Evaluating large language models trained on code. arXiv preprint arXiv:2107.03374.

Sirui Chen, Bo Peng, Meiqi Chen, Ruiqi Wang, Mengying Xu, Xingyu Zeng, Rui Zhao, Shengjie Zhao, Yu Qiao, and Chaochao Lu. 2024. Causal evaluation of language models. Preprint, arXiv:2405.00622.

Ning Ding, Yulin Chen, Bokai Xu, Yujia Qin, Zhi Zheng, Shengding Hu, Zhiyuan Liu, Maosong Sun, and Bowen Zhou. 2023. Enhancing chat language models by scaling high-quality instructional conversations. Preprint, arXiv:2305.14233.

Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2019. Unified language model pre-training for natural language understanding and generation. Advances in neural information processing systems, 32.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. arXiv preprint arXiv:2407.21783.

Yingqiang Ge, Wenyue Hua, Kai Mei, Juntao Tan, Shuyuan Xu, Zelong Li, Yongfeng Zhang, et al. 2023. Openagi: When llm meets domain experts. Advances in Neural Information Processing Systems, 36:5539–5568.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. 2025. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. arXiv preprint arXiv:2501.12948.

Simeng Han, Hailey Schoelkopf, Yilun Zhao, Zhenting Qi, Martin Riddell, Wenfei Zhou, James Coady, David Peng, Yujie Qiao, Luke Benson, Lucy Sun, Alex Wardle-Solano, Hannah Szabo, Ekaterina Zubova, Matthew Burtell, Jonathan Fan, Yixin Liu, Brian Wong, Malcolm Sailor, Ansong Ni, Linyong Nan, Jungo Kasai, Tao Yu, Rui Zhang, Alexander R. Fabbri, Wojciech Kryscinski, Semih Yavuz, Ye Liu, Xi Victoria Lin, Shafiq Joty, Yingbo Zhou, Caiming Xiong, Rex Ying, Arman Cohan, and Dragomir Radev. 2024. Folio: Natural language reasoning with first-order logic. Preprint, arXiv:2209.00840.

Divij Handa, Zehua Zhang, Amir Saeidi, and Chitta Baral. 2024. When "competency" in reasoning opens the door to vulnerability: Jailbreaking llms via novel complex ciphers. Preprint, arXiv:2402.10601.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021a. Measuring mathematical problem solving with the math dataset. NeurIPS.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021b. Measuring mathematical problem solving with the math dataset. arXiv preprint arXiv:2103.03874.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. 2024. Gpt-4o system card. arXiv preprint arXiv:2410.21276.

Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. 2024. Openai o1 system card. arXiv preprint arXiv:2412.16720.

Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. 2024a. Mixtral of experts. arXiv preprint arXiv:2401.04088.

Fengqing Jiang, Zhangchen Xu, Luyao Niu, Zhen Xiang, Bhaskar Ramasubramanian, Bo Li, and Radha Poovendran. 2024b. Artprompt: Ascii art-based jailbreak attacks against aligned llms. arXiv preprint arXiv:2402.11753.

Daniel Kang, Xuechen Li, Ion Stoica, Carlos Guestrin, Matei Zaharia, and Tatsunori Hashimoto. 2024. Exploiting programmatic behavior of llms: Dual-use through standard security attacks. In 2024 IEEE Security and Privacy Workshops (SPW), pages 132–143. IEEE.

Nikitas Karanikolas, Eirini Manga, Nikoletta Samaridi, Eleni Tousidou, and Michael Vassilakopoulos. 2023. Large language models versus natural language understanding and generation. In Proceedings of the 27th Pan-Hellenic Conference on Progress in Computing and Informatics, pages 278–290.

Alan G. Konheim. 2007. Computer Security and Cryptography. John Wiley & Sons.

Cheryl Lee, Chunqiu Steven Xia, Longji Yang, Jentse Huang, Zhouruixin Zhu, Lingming Zhang, and Michael R Lyu. 2024. A unified debugging approach via llm-based multi-agent synergy. arXiv preprint arXiv:2404.17153.

Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. 2024a. Deepseek-v3 technical report. arXiv preprint arXiv:2412.19437.

Fei Liu, Yiming Yao, Ping Guo, Zhiyuan Yang, Zhe Zhao, Xi Lin, Xialiang Tong, Mingxuan Yuan, Zhichao Lu, Zhenkun Wang, et al. 2024b. A systematic survey on large language models for algorithm design. arXiv preprint arXiv:2410.14716.

Hongwei Liu, Zilong Zheng, Yuxuan Qiao, Haodong Duan, Zhiwei Fei, Fengzhe Zhou, Wenwei Zhang, Songyang Zhang, Dahua Lin, and Kai Chen. 2024c. Mathbench: Evaluating the theory and application proficiency of llms with a hierarchical mathematics benchmark. arXiv preprint arXiv:2405.12209.

Junnan Liu, Hongwei Liu, Linchen Xiao, Ziyi Wang, Kuikun Liu, Songyang Gao, Wenwei Zhang, Songyang Zhang, and Kai Chen. 2024d. Are your llms capable of stable reasoning? arXiv preprint arXiv:2412.13147.

Kaijing Ma, Xinrun Du, Yunran Wang, Haoran Zhang, Zhoufutu Wen, Xingwei Qu, Jian Yang, Jiaheng Liu, Minghao Liu, Xiang Yue, et al. 2024. Kor-bench: Benchmarking language models on knowledge-orthogonal reasoning tasks. arXiv preprint arXiv:2410.06526.

Jarno Mielikainen. 2006. Lsb matching revisited. IEEE signal processing letters, 13(5):285–287.

S. Rani, A. Kataria, and M. Chauhan. 2022. Cyber security techniques, architectures, and design. In Holistic Approach to Quantum Cryptography in Cyber Security, pages 41–66. CRC Press.

A. Sarkar, S. R. Chatterjee, and M. Chakraborty. 2021. Role of cryptography in network security. The "Essence" of Network Security: An End-to-End Panorama, pages 103–143.

Miyu Sasaki, Natsumi Watanabe, and Tsukihito Komanaka. 2024. Enhancing contextual understanding of mistral llm with external knowledge bases.

Bruce Schneier. 2002. Cryptographic design vulnerabilities. Computer, 31(9):29–33.

Divya Shree, Seema Ahlawat, et al. 2017. A review on cryptography, attacks and cyber security. International Journal of Advanced Research in Computer Science, 8(5).

S. Soomro, M. R. Belgaum, Z. Alansari, et al. 2019. Review and open issues of cryptographic algorithms in cyber security. In 2019 International Conference on Computing, Electronics & Communications Engineering (iCCECE), pages 158–162. IEEE.

Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, et al. 2022. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. arXiv preprint arXiv:2206.04615.

Hongda Sun, Weikai Xu, Wei Liu, Jian Luan, Bin Wang, Shuo Shang, Ji-Rong Wen, and Rui Yan. 2024. Determlr: Augmenting llm-based logical reasoning from indeterminacy to determinacy. In Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 9828–9862.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V Le, Ed H Chi, Denny Zhou, et al. 2022. Challenging big-bench tasks and whether chain-of-thought can solve them. arXiv preprint arXiv:2210.09261.

Gemini Team. 2024a. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. Preprint, arXiv:2403.05530.

Qwen Team. 2024b. Qwq: Reflect deeply on the boundaries of the unknown.

Runchu Tian, Yining Ye, Yujia Qin, Xin Cong, Yankai Lin, Zhiyuan Liu, and Maosong Sun. 2024. Debugbench: Evaluating debugging capability of large language models. Preprint, arXiv:2401.04621.

Boshi Wang, Xiang Yue, and Huan Sun. 2023. Can chatgpt defend its belief in truth? evaluating llm reasoning via debate. In Findings of the Association for Computational Linguistics: EMNLP 2023, pages 11865–11881.

Alexander Wei, Nika Haghtalab, and Jacob Steinhardt. 2024. Jailbroken: How does llm safety training fail? Advances in Neural Information Processing Systems, 36.

Colin White, Samuel Dooley, Manley Roberts, Arka Pal, Ben Feuer, Siddhartha Jain, Ravid Shwartz-Ziv, Neel Jain, Khalid Saifullah, Siddartha Naidu, et al. 2024. Livebench: A challenging, contamination-free llm benchmark. arXiv preprint arXiv:2406.19314.

Yiran Wu, Feiran Jia, Shaokun Zhang, Hangyu Li, Erkang Zhu, Yue Wang, Yin Tat Lee, Richard Peng, Qingyun Wu, and Chi Wang. 2024. Mathchat: Converse to tackle challenging math problems with llm agents. In ICLR 2024 Workshop on Large Language Model (LLM) Agents.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. 2024a. Qwen2. 5 technical report. arXiv preprint arXiv:2412.15115.

Kaiyu Yang, Aidan Swope, Alex Gu, Rahul Chalamala, Peiyang Song, Shixing Yu, Saad Godil, Ryan J Prenger, and Animashree Anandkumar. 2024b. Leandojo: Theorem proving with retrieval-augmented language models. Advances in Neural Information Processing Systems, 36.

Wenlin Yao, Haitao Mi, and Dong Yu. 2024. Hdflow: Enhancing llm complex problem-solving with hybrid thinking and dynamic workflows. arXiv preprint arXiv:2409.17433.

Zheng-Xin Yong, Cristina Menghini, and Stephen H Bach. 2023. Low-resource languages jailbreak gpt-4. arXiv preprint arXiv:2310.02446.

Youliang Yuan, Wenxiang Jiao, Wenxuan Wang, Jen-tse Huang, Pinjia He, Shuming Shi, and Zhaopeng Tu. 2023. Gpt-4 is too smart to be safe: Stealthy chat with llms via cipher. arXiv preprint arXiv:2308.06463.

Haodong Duan Yuan Liu. 2023. Mmbench: Is your multi-modal model an all-around player? arXiv:2307.06281.

Li Yujian and Liu Bo. 2007. A normalized levenshtein distance metric. IEEE transactions on pattern analysis and machine intelligence, 29(6):1091–1095.

Li Zhong, Zilong Wang, and Jingbo Shang. 2024. Ldb: A large language model debugger via verifying runtime execution step-by-step. arXiv preprint arXiv:2402.16906.

Terry Yue Zhuo, Minh Chien Vu, Jenny Chim, Han Hu, Wenhao Yu, Ratnadira Widyasari, Imam Nur Bani Yusuf, Haolan Zhan, Junda He, Indraneil Paul, et al. 2024. Bigcodebench: Benchmarking code generation with diverse function calls and complex instructions. arXiv preprint arXiv:2406.15877.

MZWM Zulkifli and Zaid W Mohd. 2008. Attack on cryptography. Comput. Secur, 12(5):33–45.

A Detailed Benchmark Description

In this chapter, we provide additional details on CipherBank that were not extensively covered in the main text. This includes a detailed breakdown of plaintext tags and their distribution across subdomains, as well as a more comprehensive description of the encryption algorithms used. These details offer deeper insights into the dataset construction and the encryption schemes evaluated in this benchmark.

A.1 Tags and Plaintext Distribution Across Subdomains

Table 4 provides an overview of the specific tags associated with each subdomain within CipherBank. The dataset spans **five primary domains** and **14 subdomains**, ensuring diverse and realistic plaintext scenarios for cryptographic evaluation.

A.2 Detailed Descriptions of Encryption Algorithms

This section provides detailed descriptions of the nine encryption algorithms used in CipherBank. These algorithms span substitution, transposition, and custom-designed ciphers, covering a range of complexity levels. Notably, Rot13, Atbash, Polybius, DualAvgCode, and ParityShift also support numeric encryption, further enhancing the diversity of decryption challenges. **Table 5** outlines each algorithm and its transformation rules.Some detailed encryption examples are provided below, illustrating how different ciphers transform plaintext into ciphertext.

For each encryption algorithm, we have implemented a corresponding decryption algorithm to ensure that ciphertext can be fully restored to its original plaintext. This guarantees the reversibility and integrity of the encryption schemes used in CipherBank, allowing for a rigorous evaluation of model decryption capabilities. The decryption process follows the exact inverse of the encryption transformations, ensuring consistency across all test cases.

B Experimental Setup Details

In our evaluation, we adopt a 3-shot approach. A more natural Ciphertext-Only Attack (zero-shot) setting was not adopted, as it would reduce the task to brute-force decryption, where the model blindly applies all known encryption algorithms in search of a coherent output. This contradicts the goal of reasoning-based inference, where the model is expected to deduce encryption rules from provided examples rather than rely on exhaustive trial and error.

To ensure a balanced evaluation of decryption difficulty, substitution ciphers exclude numbers to prevent inconsistencies arising from differing cyclic structures. In contrast, ciphers that do not involve direct substitution, such as Reverse, WordShift, and similar methods, process numbers normally, preserving structural integrity within the encrypted text.

For all open-source models, we conduct evaluations using the OpenCompass[7] framework with default temperature to ensure consistent outputs. For models evaluated via API, **we perform** 5 **independent test runs** per model and report the average result to enhance stability and reliability.

B.1 Prompts Used for Querying

This section outlines the prompts used to query models during evaluation. To ensure consistency, all models were tested under a 3-shot setting, where they were provided with three plaintext-ciphertext pairs before attempting to decrypt a new ciphertext. The prompts were designed to encourage logical inference rather than relying on prior knowledge, guiding models to extract encryption patterns and apply the learned rules systematically. Below, **Figure 5** provides the system prompt (some reasoning models may not support system prompts), while **Figure 6** present the detailed user prompts.

B.2 Post-processing Methods

During querying, we instruct the model to think step by step and enclose the final decrypted output within **<result>...</result>** tags. To extract the decoded plaintext, we apply the regular expression **'<result>(.*?)</result>'**, capturing the content between these tags. The matching process is **case-insensitive**, aligning with algorithms like Polybius, which inherently do not differentiate between uppercase and lowercase letters when restoring plaintext. This ensures consistency across different decryption schemes.

[7]https://github.com/open-compass/opencompass

Table 4: Tag Distribution Across Subdomains in CipherBank

Domain	**Subdomain**	**Tags**
Personal Privacy Data	Identity Information	Name, ID Card Number, Passport Number, Date of Birth, Gender, Nationality, Marital Status, Mobile Number, Family Member Information (e.g., immediate family names, contact information), Residential Address
	Health Information	Medical Record Number (Patient ID), Diagnosis Records, Surgery Records, Examination Reports (e.g., X-ray, CT scan results, heart rate, blood pressure, blood sugar level, blood type), Disease History, Allergy History, Vaccination Records, Family Medical History
	Educational Data	Student ID (Student Number), School Records (Enrollment Date, Graduation Date), Academic Records (Subjects, Grades, GPA, Ranking), Degree Information (Bachelor, Master, Doctorate), Awards and Penalties Records (Disciplinary Records)
Enterprise Sensitive Data	Business Information	Business Plans (e.g., Annual Plan, Five-Year Plan), Marketing Strategy (e.g., Marketing Promotion Plan, Advertising Budget), Customer Lists (e.g., Customer Contacts, Preferences), Supplier Information (Supplier List, Cooperation Agreements), Internal Financial Budgets (Cost Structure, Profit Forecasts)
	Intellectual Property	Product Design Plans (e.g., Prototype Drawings, Design Documents), Internal Technical Documents (e.g., Technical Manuals, Specifications), Test Data (e.g., Product Performance Test Results, Quality Control Records), Copyright Data, Patent Data
	Employee Information	Contact Information (e.g., Phone Numbers, Email Addresses), Work Experience, Position and Department Information, Salary and Benefits Information (e.g., Salary Amount, Bonuses, Allowances), Performance Evaluation (e.g., Performance Scores, Promotion Records), Contract Information (e.g., Employment Contract, Non-Disclosure Agreement)
Public Safety Data	Police Data	Case Information (Case Number, Case Type, Filing Date), Criminal Records (Suspect Information, Crime Time, Crime Location), Alarm Records (Informer Information, Alarm Time, Alarm Content), Investigation Reports (Investigation Results, Investigation Progress), Arrest Records (Arrest Time, Location, Action Description), Traffic Enforcement Data (Violation Records, Penalty Information), Police Officer Information (Officer Number, Name, Position, Department), Police Resource Allocation (Vehicle, Equipment, Weapon Usage Records)
	National Security Data	Border Crossing Records (Entry and Exit Personnel Information, Vehicle Registration), Customs Inspection Data (Cargo List, Contraband Records), Territorial Patrol Data (Patrol Reports, Anomalies Records), Cyber Security Monitoring Data (Cyber Attack Records, Threat Intelligence)
	Military Data	Operation Plans, Target Location, Troop Deployment, Military Base Distribution, Defense Works Location
Financial Confidential Data	Banking Information	Account Number, Bank Card Number, Payment Method, Payment Platform ID, Transaction Details, Loan Amount, Interest Rate, Repayment Plan, Investment Records (Stocks, Funds, Bonds)
	Personal Income	Salary Amount, Pay Date, Tax Number, Tax Return Records
Internet Records	Browsing Records	Page Interaction, Search Behavior, Click Activity, Device Information, Geolocation, Checkout Process, Multimedia Interaction, Download Records
	Cookie Data	Session Management, User Identification, Ad Targeting, Behavior Tracking, Authentication Tokens, Login Status
	User Preferences	Preferred Genres, Device Usage Habits, Notification Preferences, Shopping Preferences, Video Preferences, Reading Habits

Example A.1: Plain-Ciphertext Pair (Identity Information) - Only Letter

# Domain: Personal Privacy Data

## Subdomain: Identity Information

### Tag Combination: ["Name", "Date of Birth", "Passport Number"]

Plaintext:

Peter was born on April 23, 1985, and carries a passport with the number X123456789.

Encryption results:

(1) **Rot13**: *Crgre jnf obea ba Ncevy 23, 1985, naq pneevrf n cnffcbeg jvgu gur ahzore K123456789.*

(2) **Atbash**: *Kvgvi dzh ylim lm Zkiro 23, 1985, zmw xziirvh z kzhhklig drgs gsv mfnyvi C123456789.*

(3) **Polybius**: *34 15 42 15 36 45 11 41 12 33 36 32 33 32 11 34 36 23 26 2 3 , 1 9 8 5 , 11 32 14 13 11 36 36 23 15 41 11 34 11 41 41 34 33 36 42 45 23 42 22 42 22 15 32 43 31 12 15 36 46 1 2 3 4 5 6 7 8 9.*

(4) **Vigenère**: *Pgeet wcd dzrp op Arcin 23, 1985, cyd natcigd c pcdsrzrv wkeh ehg nwxbgc Z123456789.*

(5) **Reverse**: *.987654321X rebmun eht htiw tropssap a seirrac dna ,5891 ,32 lirpA no nrob saw reteP*

(6) **SwapPairs**: *ePet raw sobnro npAir l32 ,9158 ,na dacrrei s aapssoptrw ti hht eunbmreX 21436587.9*

(7) **DualAvgCode**: *OQdfsudfqs vxaart acnpqsmo npmo AAoqqshjkm 23, 1985, aamoce bdaaqsqshjdfrt aa oqaartrtoqnpqssu vxhjsugi sugidf motvlnacdfqs WY123456789.*

(8) **ParityShift**: *Qduds vzr cnso no Zqshm 23, 1985, zoe bzsshdr z qzrrqnsu vhui uid otlcds Y123456789.*

(9) **WordShift**: *erPet was nbor no ilApr 23, 5,198 and riescar a sportpas hwit the bernum 3456789.X12*

Example A.2: Plain-Ciphertext Pair (Police Data) - Only Letter

# Domain: Public Safety Data

## Subdomain: Police Data

### Tag Combination: ["Suspect Information", "Crime Time", "Crime Location", "Police Officer Information"]

Plaintext:

Suspect: Jonathan, Crime: Burglary, Time: 2022-03-12 14:30, Location: 123 Elm Street, Officer Smith observed suspicious activity near 5th Ave on 2022-03-13.

Encryption results:

(1) **Rot13**: *Fhfcrpg: Wbanguna, Pevzr: Ohetynel, Gvzr: 2022-03-12 14:30, Ybpngvba: 123 Ryz Fgerrg, Bssvpre Fzvgu bofreirq fhfcvpvbhf npgvivgl arne 5gu Nir ba 2022-03-13.*

(2) **Atbash**: *Hfhkvxg: Qlmzgszm, Xirnv: Yfitozib, Grnv: 2022-03-12 14:30, Olxzgrlm: 123 Von Hgivvg, Luurxvi Hnrgs lyhvievw hfhkrxrlfh zxgrergb mvzi 5gs Zev lm 2022-03-13.*

(3) **Polybius**: *41 43 41 34 15 13 42 : 24 33 32 11 42 22 11 32 , 13 36 23 31 15 : 12 43 36 21 26 11 36 51 , 42 23 31 15 : 2 0 2 2 - 0 3 - 1 2 1 4 : 3 0 , 26 33 13 11 42 23 33 32 : 1 2 3 15 26 31 41 42 36 15 15 42 , 33 16 16 23 13 15 36 41 31 23 42 22 33 12 41 15 36 44 15 14 41 43 41 34 23 13 23 33 43 41 11 13 42 23 44 23 42 51 32 15 11 36 5 42 22 11 44 15 33 32 2 0 2 2 - 0 3 - 1 3.*

(4) **Vigenère**: *Swdpgnt: Jqyavsap, Eciop: Mutrlccy, Tkxe: 2022-03-12 14:30, Lqnavtop: 123 Plo Svcege, Zfhtcgc Uxivs qmsgcvgo ufsrtckzuu aeeixtta nglr 5tj Axp qy 2022-03-13.*

(5) **Reverse**: *31-30-2202 no evA ht5 raen ytivitca suoicipsus devresbo htimS reciffO ,teertS mlE 321 :noitacoL ,03:41 21-30-2202 :emiT ,yralgruB :emirC ,nahtanoJ :tcepsuS.*

(6) **SwapPairs**: *uSpsce:tJ notaah,nC irem :uBgralyr ,iTem :02220–3211 :403 ,oLacitno :21 3lE mtSerte ,fOifec rmSti hboesvrdes suipicuo scaitivytn ae rt5 hvA eno2 20-2301-3.*

(7) **DualAvgCode**: *RTtvrtoqdfbdsu: IKnpmoaasugiaamo, BDqshjlndf: ACtvqsfhkmaaqsxz, SUhjlndf: 2022-03-12 14:30, KMnpbdaasuhjnpmo: 123 DFkmln RTsuqsdfdfsu, NPegeghjbddfqs RTlnhjsugi npacrtdfqsuwdfce rttvrtoqhjbdhjnptvrt aabd-suhjuwhjsuxz modfaaqs 5sugi AAuwdf npmo 2022-03-13.*

(8) **ParityShift**: *Rtrqdbu: Knozuizo, Bshld: Ctsfmzsx, Uhld: 2022-03-12 14:30, Mnbzuhno: 123 Dml Rusddu, Ngghbds Rlhui ncrdswde rtrqhbhntr zbuhwhux odzs 5ui Zwd no 2022-03-13.*

(9) **WordShift**: *pect:Sus athan,Jon me:Cri glary,Bur e:Tim 2-03-12202 30,14: ation:Loc 123 Elm eet,Str icerOff thSmi ervedobs picioussus ivityact rnea 5th Ave no 2-03-13202.*

Example A.3: Plain-Ciphertext Pair (Health Information) - Letter&Number

# Domain: Personal Privacy Data

## Subdomain: Health Information

### Tag Combination: ["Patient ID", "Diagnosis Records"]

Plaintext:

Patient ID: R094713; Name: Jamie Lee; Age: 45; Gender: Female; EMR: EHR-234987.

Encryption results:

(1) **Rot13**: *Cngvrag VQ: E327046; Anzr: Wnzvr Yrr; Ntr: 78; Traqre: Srznyr; RZE: RUE-567210.*

(2) **Atbash**: *Kzgrvmg RW: I905286; Mznv: Qznrv Ovv; Ztv: 54; Tvmwvi: Uvnzov; VNI: VSI-765012.*

(3) **Polybius**: *34 11 42 23 15 32 42 23 14 : 36 66 65 56 63 53 55 ; 32 11 31 15 : 24 11 31 23 15 26 15 15 ; 11 21 15 : 56 61 ; 21 15 32 14 15 36 : 16 15 31 11 26 15 ; 15 31 36 : 15 22 36 - 54 55 56 65 64 63.*

(4) **Reverse**: *.789432-R HRE ;elameF :redneG ;54 :egA ;eeL eimaJ :emaN ;317490 R :DI tneitaP*

(5) **SwapPairs**: *aPtetni DI: 0R94713; aNme: aJmei eLe; gAe: 45; eGndre: eFmale; MRE: HRE-239487.*

(6) **WordShift**: *atientP ID: R94713; ameN: Jamie eLe; geA: 45; enderG: emaleF; REM: EHR-234987.*

(7) **DualAvgCode**: *OQaasuhjdfmosu HJCE: QS009935680224; MOaalndf: IKaalnhjdf KMdfdf; AAfhdf: 3546; FHdfmoced-fqs: EGdflnaakmdf; DFLNQS: DFGIQS-132435997968.*

(8) **ParityShift**: *Qzuhdou HE: S185602; Ozld: Kzlhd Mdd; Zfd: 54; Fdoeds: Gdlzmd; DLS: DIS-325896.*

Example A.4: Plain-Ciphertext Pair (Banking Information) - Letter&Number

# Domain: Financial Confidential Data

## Subdomain: Banking Information

### Tag Combination: ["Account Number", "Bank Card Number","Payment Platform ID"]

Plaintext:

Account Number: 123456789, Bank: LA Bank, Card Number: 9876-5432-1098-7654, Payment Method: Virtual Credit Card, Payment Platform ID: ABC123XYZ, Timestamp: 2023-09-15 14:35, Amount: $250.00.

Encryption results:

(1) **Rot13**: *Nppbhag Ahzore: 456789012, Onax: YN Onax, Pneq Ahzore: 2109-8765-4321-0987, Cnlzrag Zrgubq: Iveghny Perqvg Pneq, Cnlzrag Cyngsbez VQ: NOP456KLM, Gvzrfgnzc: 5356-32-48 47:68, Nzbhag: $583.33.*

(2) **Atbash**: *Zxxlfmg Mfnyvi: 876543210, Yzmp: OZ Yzmp, Xziw Mfnyvi: 0123-4567-8901-2345, Kzbnvmg Nvgslw: Erigfzo Xivwrg Xziw, Kzbnvmg Kozgulin RW: ZYX876CBA, Grnvhgznk: 7976-90-84 85:64, Znlfmg: $749.99.*

(3) **Polybius**: *11 13 13 33 43 32 42 32 43 31 12 15 36 : 53 54 55 56 61 62 63 64 65 , 12 11 32 25 : 26 11 12 11 32 25 , 13 11 36 14 32 43 31 12 15 36 : 65 64 63 62 - 61 56 55 54 - 53 66 65 64 - 63 62 61 56 , 34 11 51 31 15 32 42 31 15 42 22 33 14 : 44 23 36 42 43 11 26 13 36 15 14 23 42 13 11 36 14 , 34 11 51 31 15 32 42 34 26 11 42 16 33 36 31 23 14 : 11 12 13 53 54 55 46 51 52 , 42 23 31 15 41 42 11 31 34 : 54 66 54 55 - 66 65 - 53 61 53 56 : 55 61 , 11 31 33 43 32 42 : $ 54 61 66 . 66 66 .*

(4) **Vigenère**: *Swdpgnt: Jqyavsap, Eciop: Mutrlccy, Tkxe: 2022-03-12 14:30, Lqnavtop: 123 Plo Svcege, Zfhtcgc Uxivs qmsgcvgo ufsrtckzuu aeeixtta nglr 5tj Axp qy 2022-03-13.*

(5) **Reverse**: *.00.052$:tnuomA ,53:41 51-90-3202 :pmatsemit ,ZYX321CBA :DI mroftalP tnemyap ,draC tiderC lautriV :dohtem tnemyap ,4567-8901-2345-6789 :rebmuN draC ,knaB AL :knaB ,987654321 :rebmuN tnuoccA*

(6) **SwapPairs**: *cAotcnu mNuber: 214365879, aBnk: A Lank, aCrd Nmu:bre 8967-5423-1980-7564, aPymnet Mtohed: Vritaul Cerdti aCdr, aPymnet Ptaforml DI: BAC321YXZ, iTmsetamp: 3202-90-51 53:41, aAmount: $250.00.*

(7) **DualAvgCode**: *AAbdbdnptvmosu MOtvlnacdfqs: 021324354657687999, ACaamojl: KMAA ACaamojl, BDaaqsce MOtvlnacdfqs: 99796857-46352413-02009979-68574635, OQaaxzlndfmosu LNdfsuginpce: UWhjqssutvaakm BDqsdfcehjsu BDaaqsce, OQaaxzlndfmosu OQkmaasuegnpqsln HJCE: AAACBD021324WYXZZZ, SUhjlndfrtsuaalnoq: 13001324-0099-0246 0235:2446, AAlnnptvmosu: $134600.0000.*

(8) **ParityShift**: *Zbbntou Otlcds: 032547698, Czoj: MZ Czoj, Bzse Otlcds: 8967-4523-0189-6745, Qzxldou Lduine: Whsutzm Bsdehu Bzse, Qzxldou Qmzugnsl HE: ZCB032YXA, Uhldruzlq: 3132-18-04 05:24, Zlntou: $341.11.*

Table 5: Descriptions of Encryption Algorithms in CipherBank

Algorithm	Description
Rot13	A simple *substitution cipher* that shifts each letter 13 places forward in the alphabet. Encryption and decryption are identical, as applying the transformation twice restores the original text. Non-alphabetic characters remain unchanged. Additionally, Rot13 in CipherBank supports number encryption by shifting digits cyclically within the range 0-9.
Atbash	A *monoalphabetic substitution cipher* where each letter is replaced with its counterpart from the reversed alphabet (e.g., A→Z, B→Y). Since the transformation is symmetric, encryption and decryption follow the same process. CipherBank's Atbash implementation extends this to digits, where each number is replaced with its complement relative to 9 (e.g., 0→9, 1→8, ..., 9→0).
Polybius	A *fractionating substitution cipher* that replaces each letter with a two-digit coordinate from a 6×6 grid, mapping characters to numerical positions. Traditional Polybius squares typically use a 5×5 grid, supporting only letter encryption while merging I and J into the same cell, leading to ambiguity during decryption. To address this limitation and enable number encryption, CipherBank extends the Polybius square to a 6×6 grid, allowing both letters and numbers to be uniquely represented as coordinate pairs, increasing the cipher's complexity.
Vigenère	A *polyalphabetic substitution cipher* that employs multiple shifting alphabets determined by a repeating key. Unlike monoalphabetic ciphers that use a single mapping, Vigenère utilizes multiple substitution tables, where each plaintext letter is shifted based on the corresponding key character's position in the alphabet. By default, the key is set to `"ACL"`. This multi-table approach enhances security by distributing letter frequencies across different shifts, making it more resistant to frequency analysis. Decryption reverses this process by applying the inverse shifts dictated by the key. Unlike Rot13, it requires a key for both encryption and decryption.
Reverse	A *transposition cipher* that reverses the order of all characters in the plaintext. Since it does not substitute characters, it preserves all information but alters the sequence, making it effective against naive attacks.
SwapPairs	A *transposition cipher* that swaps adjacent characters in the plaintext. If the text length is odd, the final character remains unchanged. Decryption follows the same swapping process.
DualAvgCode	A *custom transformation* where each letter expands into two adjacent characters, shifting one position forward and one position backward in the ASCII table. Special cases (e.g., 'a', 'z', 'A', 'Z') are duplicated instead. CipherBank extends this method to digits, where each number expands into two adjacent values (e.g., 2 → "13", 5 → "46"), increasing redundancy in the encrypted text.
ParityShift	A *custom encryption method* that shifts each letter one position forward or backward based on its ASCII parity. Even-ASCII characters shift forward, while odd-ASCII characters shift backward. For digits, ParityShift follows a similar rule, shifting numbers based on their parity (e.g., even numbers shift up, odd numbers shift down within 0-9).
WordShift	A transformation applied at the word level rather than the character level. Each word undergoes a left shift by a fixed number of positions, cycling characters within the word while preserving word spacing. Decryption reverses this shift, ensuring character order is restored within each word. By default, the shift is set to 3 positions.

Example B.1

## Role:

Cryptography Analysis Expert.

## Goals:

Utilize the provided ciphertext and plaintext examples to analyze encryption patterns and decrypt new ciphertext.

## Workflow:

1. Analyze the provided ciphertext and plaintext examples to identify possible encryption patterns and rules.
2. Apply the decryption algorithm to the new ciphertext, attempt to decrypt, and verify the results.

Figure 5: System Prompt

Example B.2

## Background:

Decrypt the provided Ciphertext and return the corresponding Plaintext following the given examples.

Think step by step.

Provide the Plaintext result in the format <result>text</result>, where text is the decrypted text.

## Examples:

* Example 1:
  - Ciphertext:
    Anzr: Rzvyl Wbuafba; Qngr bs Ovegu: Whyl 15, 1990; Cnffcbeg Ahzore: L987654321
  - Plaintext:
    Name: Emily Johnson; Date of Birth: July 15, 1990; Passport Number: Y987654321
* Example 2:
  - Ciphertext:
    Pnfr Ahzore: 2024-CF-001234; Pnfr Glcr: Gursg/Oernx-Va; Svyvat Qngr: Bpgbore 19, 2024
  - Plaintext:
    Case Number: 2024-PS-001234; Case Type: Theft/Break-In; Filing Date: October 19, 2024
* Example 3:
  - Ciphertext:
    gnk_vqragvsvre: GKA-2023-NOP456, gnk_erpbeqf: lrne: 2023, fgnghf: Cebprffrq, ershaq_vffhrq: 620.00
  - Plaintext:
    tax_identifier: TXN-2023-ABC456, tax_records: year: 2023, status: Processed, refund_issued: 620.00

## Input:

- Ciphertext:
  Yrqvn, na Nzrevpna, erfvqrf va Ybf Natryrf.
- Plaintext:

Figure 6: User Prompt (Rot13 - 3shot - Only Letter)

C Extended Experimental Results

C.1 Levenshtein Distance Evaluation from Main Results

In the main text, most reported results are based on *accuracy*, which provides a binary assessment of decryption success. However, accuracy does not account for cases where decrypted outputs closely resemble the ground truth but contain minor errors. To provide a more fine-grained evaluation, we also compute *Levenshtein similarity*, which measures the edit distance between the model output and the correct plaintext.

We define the **Levenshtein similarity score** as follows:

$$S_{\text{lev}} = 1 - \frac{d_{\text{lev}}(P_{\text{pred}}, P_{\text{ref}})}{\max(|P_{\text{pred}}|, |P_{\text{ref}}|)} \tag{1}$$

where:

- $d_{\text{lev}}(P_{\text{pred}}, P_{\text{ref}})$ is the *Levenshtein distance* between the predicted and reference plaintexts.
- $|P_{\text{pred}}|$ and $|P_{\text{ref}}|$ denote the lengths of the predicted and reference plaintexts, respectively.

This metric *normalizes the edit distance* by the length of the longer string, ensuring that similarity is measured on a scale from 0 to 1, where 1 represents an exact match and lower values indicate increasing deviations from the ground truth.

The corresponding **Levenshtein-based evaluation results** for **Table 2** are presented in **Table 6** and **Figure 7**, offering deeper insights into models' decryption performance beyond strict accuracy metrics.

One key observation is that most models achieve significantly higher Levenshtein similarity scores than their accuracy scores, indicating that even when decryption is incorrect, outputs often retain structural similarities to the original plaintext. This suggests that models capture some encryption patterns but struggle with full decryption, failing to consistently apply correct transformations. Notably, Claude-Sonnet-3.5 achieves near-perfect scores (>0.99 for most ciphers), demonstrating its ability to minimize decryption errors while maintaining structural accuracy, making it the most reliable model overall.

Interestingly, reasoning models such as DeepSeek-R1 and o1 exhibit a large gap between accuracy and Levenshtein similarity. Despite their moderate accuracy, their similarity scores often exceed 0.80, indicating that they frequently produce outputs that preserve much of the original structure but contain systematic errors. This suggests that reasoning models are better at capturing encryption logic but may struggle with precise execution, sometimes overcomplicating simpler tasks.

Conversely, chat models such as DeepSeek-V3 and Llama-based models exhibit high variability, showing relatively low accuracy but moderate Levenshtein similarity (0.40 - 0.70). This indicates a tendency toward semantic approximation rather than strict decryption, where models generate linguistically plausible outputs that fail to adhere to precise encryption rules.

Another notable trend is that transposition ciphers (e.g., Reverse, SwapPairs) yield lower Levenshtein similarity scores across all models, confirming that character reordering remains a major challenge. Unlike substitution ciphers, where models can rely on token-level mappings, transposition ciphers require strict positional tracking, which even the strongest models struggle to handle effectively.

Overall, Levenshtein similarity results highlight fundamental differences in how chat and reasoning models approach decryption. Chat models rely more on semantic fluency, leading to structurally incorrect but coherent outputs, whereas reasoning models exhibit stronger pattern retention but occasionally fail due to overgeneralization or overthinking. These findings suggest that while LLMs can approximate decryption rules, achieving precise symbolic transformations remains a significant challenge, especially for positional-based ciphers.

C.2 Additional Analysis and Insights

In this section, we present more detailed experimental results that complement the findings in the main text. These additional analyses provide further insights into model performance across different encryption schemes, highlighting trends, challenges, and specific cases where models excel or struggle.

In the analysis of length sensitivity, plaintexts of different lengths can be seen in **Figure 8.** The impact of plaintext length on decryption performance is shown in **Table 7** and **Table 8**, where we compare model accuracy on short vs. long texts. These results illustrate how increasing text length affects model performance, revealing notable differences in decryption robustness across various architectures

The dataset used for the noise interference exper-

Table 6: Results on CipherBank(3-shot) Levenshtein similarity

Model	**Substitution Ciphers**				**Transposition Ciphers**		**Custom Ciphers**			**Cipher Score**
	Rot13	At ba sh	Polybius	Vigenère	Reverse	SwapPairs	DualAvgCode	ParityShift	WordShift	Levenshtein Similarity
Open-source Chat Models										
Mixtral-8x22B-v0.1	0.4542	0.3744	0.2694	0.4032	0.3810	0.4745	0.3330	0.3871	0.6401	0.4130
Qwen2.5-72B-Instruct	0.5556	0.4288	0.3042	0.4016	0.4022	0.5308	0.3718	0.4786	0.8427	0.4796
Llama-3.1-70B-Instruct	0.5776	0.4378	0.3132	0.4431	0.3775	0.5542	0.3990	0.4505	0.7288	0.4758
Llama-3.3-70B-Instruct	0.5754	0.4054	0.1317	0.4397	0.2482	0.5375	0.3833	0.4096	0.4580	0.3988
DeepSeek-V3	0.9195	0.7594	0.4562	0.4844	0.9088	0.6975	0.4205	0.5731	0.8887	0.6787
Closed-source Models										
GPT-4o-mini-2024-07-18	0.6459	0.4935	0.2463	0.4499	0.5664	0.6005	0.3418	0.4188	0.7258	0.4988
GPT-4o-2024-08-06	0.9603	0.5876	0.3445	0.5346	0.8170	0.7968	0.4304	0.5850	0.8940	0.6612
GPT-4o-2024-11-20	0.9340	0.6054	0.3511	0.5338	0.7277	0.6780	0.4235	0.5530	0.8715	0.6309
gemini-1.5-pro	0.9309	0.5043	0.4969	0.5201	0.7536	0.7317	0.4784	0.5720	0.8819	0.6522
gemini-2.0-flash-exp	0.9616	0.6567	0.4813	0.5064	0.8901	0.7569	0.4476	0.5308	0.8605	0.6769
Claude-Sonnet-3.5-1022	**0.9984**	**0.9961**	**0.9955**	**0.7143**	**0.9893**	**0.9262**	0.7874	**0.9883**	**0.9712**	**0.9296**
Reasoning Models										
QwQ-32B-Preview	0.2477	0.1591	0.1231	0.1660	0.1444	0.1666	0.1564	0.1645	0.3057	0.1815
DeepSeek-R1	0.9920	0.9761	0.9344	0.5227	0.7368	0.7213	0.8316	0.6928	0.8491	0.8063
gemini-2.0-flash-thinking	0.9664	0.8571	0.9074	0.5511	0.8508	0.7788	0.4261	0.7353	0.8777	0.7723
o1-mini-2024-09-12	0.9757	0.9860	0.9563	0.5412	0.5959	0.5267	0.3954	0.6935	0.7236	0.7105
o1-2024-12-17	0.8320	0.9928	0.9640	0.5642	0.7725	0.9208	**0.8653**	0.6562	0.9335	0.8335

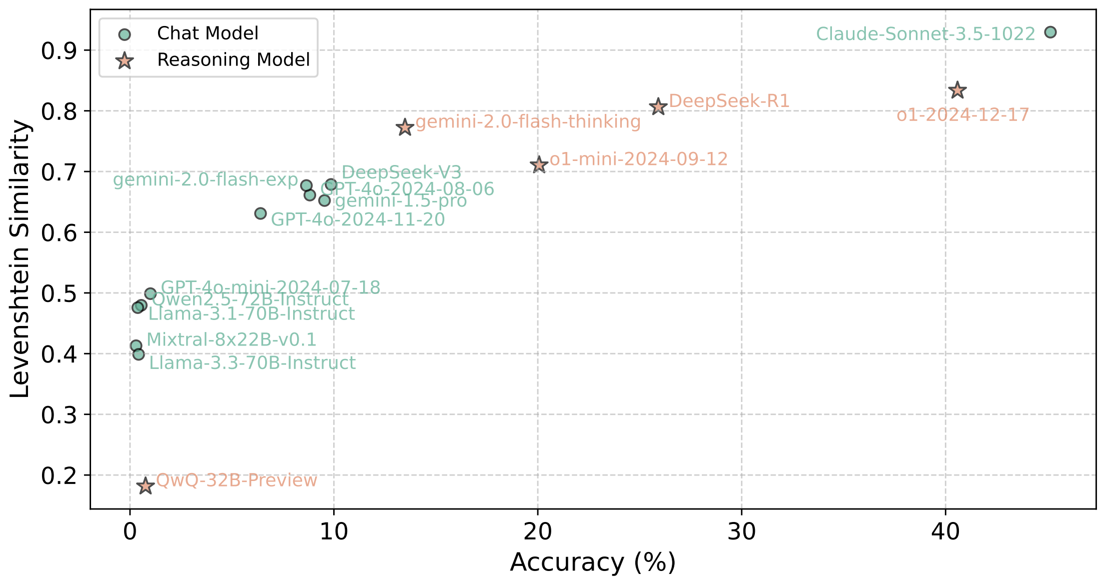


Figure 7: Model Performance - Accuracy vs. Levenshtein Similarity.

Table 7: Decryption Performance on Short Texts

Model	**Substitution Ciphers**				**Transposition Ciphers**		**Custom Ciphers**			**Cipher Score**
	Rot13	Atbash	Polybius	Vigenère	Reverse	SwapPairs	DualAvgCode	ParityShift	WordShift	$\text{Accuracy}_{\text{avg}}$
DeepSeek-V3	40.00	27.83	4.35	1.74	29.57	0.87	0.87	2.61	11.3	13.24
DeepSeek-R1	80.00	71.30	53.04	0.87	18.26	0.87	35.65	18.26	12.17	32.27
GPT-4o-2024-11-20	34.78	13.04	0.87	0	21.74	1.74	0.87	1.74	10.43	9.47
gemini-2.0-flash-exp	42.61	4.35	1.74	0.87	40.87	2.61	0	1.74	8.70	11.50
Claude-Sonnet-3.5-1022	86.09	77.39	69.57	3.48	77.39	8.70	9.57	63.48	42.61	48.70
gemini-2.0-flash-thinking	52.17	26.96	33.91	2.61	33.91	0.87	0	13.91	14.78	19.90
o1-mini-2024-09-12	64.35	82.61	65.22	0	15.65	0	6.67	13.91	2.61	33.77
o1-2024-12-17	61.74	89.57	84.55	0.87	23.48	46.67	61.74	17.17	35.80	47.61

Table 8: Decryption Performance on Long Texts

Model	Substitution Ciphers				Transposition Ciphers		Custom Ciphers			Cipher Score
	Rot13	Atbash	Polybius	Vigenère	Reverse	SwapPairs	DualAvgCode	ParityShift	WordShift	$\text{Accuracy}_{\text{avg}}$
DeepSeek-V3	26.53	4.76	0.68	0	9.52	0	0	0	5.44	5.22
DeepSeek-R1	68.03	48.98	37.41	0	4.76	0	14.97	8.84	5.44	20.94
GPT-4o-2024-11-20	20.41	4.08	0	0	12.24	0	0	0	3.40	4.46
gemini-2.0-flash-exp	30.61	2.04	1.36	0	20.41	0.68	0	0	2.72	6.42
Claude-Sonnet-3.5-1022	92.52	78.91	82.31	1.36	63.95	5.44	2.72	63.27	40.14	47.85
gemini-2.0-flash-thinking	31.29	9.52	12.24	0	14.29	1.36	0	2.72	4.76	8.47
o1-mini-2024-09-12	31.97	57.14	32.65	0	0	0	0	2.72	0	17.35
o1-2024-12-17	58.50	70.75	61.11	0.68	8.16	15.38	41.5	8.66	25.66	34.38

Example C.1: Plaintext Examples

Short: James, American, is married to Susan.

Long: John Smith, born on January 15, 1990, holds American nationality and resides at 123 Elm Street, Springfield, Illinois. His mobile number is +1-312-555-6789, and his ID card number is IDURITY1234567. He is married to Jane Smith, who can be reached at +1-312-555-6789. They have two children: Emily (16, high school) and Michael (12, middle school). Their address and contact information are the same.

Short: Jimmy, GPA: 3.71.

Long: David Wilson, Masters in Data Science, GPA: 3.95, Expected Graduation: 2023, Courses: Big Data Analytics, Machine Learning, Data Visualization.

Short: Medical Record Number: 987-654-321; Patient Name: James.

Long: David Wilson, Masters in Data Science, GPA: 3.95, Expected Graduation: 2023, Courses: Big Data Analytics, Machine Learning, Data Visualization.

Short: Lucas, lucas@ucc.company.com

Long: Hank, Senior Developer, IT Department, Salary: $95,000, Bonuses: $5,000, Allowances: $2,000 (Remote Work), Performance Rating: A, Full-time, Start Date: 2020-03-15, Last Promotion: 2021-08-10, Benefits: Health Insurance, Retirement 5%, Training: $1,500/year, Projects: Nexus, Zeta, Feedback: 4.5/5

Figure 8: Samples used for length sensitivity analysis

Example C.2: Noise Example

Example 1:

Origin: Card Number: 9876 5432 1098 7654

Noise: Card Numbr: 9876 54-32 1O98 765four

Example 2:

Origin: Pay Date: 2023-05-15, Income: $75,000, Currency: USD, Bonus: $5,000

Noise: Pay Date (scheduled): 2023-05-15! Income approx: $75,000. Currency spec: USD, and Bonus = $5,000.

Example 3:

Predictions: Officer ID: P12345, Name: John, Position: Sergeant, Department: Homicide

References: Officer Identification-No.: P12345, Full-Name: John (J.), Job-Title: Sergeant, Dept.: Homicide Squad.

Figure 9: The samples used for the noise comparison experiments.

Table 9: Decryption Performance without Noise

Model	Rot13	Atbash	Reverse	SwapPairs	ParityShift	WordShift	$\text{Accuracy}_{\text{avg}}$
			Open-source Models				
DeepSeek-V3	50.00	31.50	18.50	6.50	9.00	17.00	22.08
DeepSeek-R1	83.50	77.50	42.00	2.50	20.00	5.50	38.50
			Closed-source Models				
GPT-4o-2024-11-20	49.50	10.50	13.50	0	3.50	5.50	13.75
Gemini-2.0-flash-exp	45.00	7.50	42.50	2.50	5.00	15.50	19.67
Claude-Sonnet-3.5-1022	92.50	85.00	62.50	10.00	70.00	35.00	59.17
Gemini-2.0-flash-thinking	62.50	33.50	22.50	0	17.50	1.50	22.92
o1-mini-2024-09-12	55.50	67.50	5.00	0	17.50	0	24.25

Table 10: Decryption Performance with Noise

Model	Rot13	Atbash	Reverse	SwapPairs	ParityShift	WordShift	$\text{Accuracy}_{\text{avg}}$
			Open-source Models				
DeepSeek-V3	8.50	10.50	7.50	0	0.50	1.50	4.75
DeepSeek-R1	33.50	23.00	4.50	0	1.50	0	10.42
			Closed-source Models				
GPT-4o-2024-11-20	5.50	0	4.50	0	0	0	1.67
Gemini-2.0-flash-exp	2.50	0	0	2.50	0	0	0.83
Claude-Sonnet-3.5-1022	50.50	40.00	20.00	2.50	30.00	7.50	25.08
Gemini-2.0-flash-thinking	30.50	19.00	3.50	0	2.50	0	9.25
o1-mini-2024-09-12	15.00	20.0	0	0	0	0	5.83

iments can be found in **Figure 9**. Detailed results on the impact of noise on decryption performance are presented in **Table 9** and **Table 10**, comparing model performance on short and long plaintexts under noisy conditions. These findings highlight the varying degrees of resilience across models, with some maintaining reasonable performance under noise while others degrade significantly.

In the analysis of the impact of encryption scope on decryption performance, the test prompts used are shown in **Figure 10**. Detailed results are presented in **Table 11**. This analysis compares model performance when encrypting only letters versus encrypting both letters and numbers. The results highlight how different models handle the increased complexity introduced by number encryption, showing varying degrees of adaptability. While some models maintain relatively stable per-

Example C.3

## Background:

Decrypt the provided Ciphertext and return the corresponding Plaintext following the given examples.

Think step by step.

Provide the Plaintext result in the format <result>text</result>, where text is the decrypted text.

## Examples:

* Example 1:

- Ciphertext:

  Mznv: Vnrob Qlsmhlm; Wzgv lu Yrigs: Qfob 84, 8009; Kzhhklig Mfnyvi: B012345678

- Plaintext:

  Name: Emily Johnson; Date of Birth: July 15, 1990; Passport Number: Y987654321

* Example 2:

- Ciphertext:

  Xzhv Mfnyvi: 7975-KH-998765; Xzhv Gbkv: Gsvug/Yivzp-Rm; Urormt Wzgv: Lxglyvi 80, 7975

- Plaintext:

  Case Number: 2024-PS-001234; Case Type: Theft/Break-In; Filing Date: October 19, 2024

* Example 3:

- Ciphertext:

  gzc_rwvmgrurvi: GCM-7976-ZYX543, gzc_ivxliwh: bvzi: 7976, hgzgfh: Kilxvhhvw, ivufmw_rhhfvw: 379.99

- Plaintext:

  tax_identifier: TXN-2023-ABC456, tax_records: year: 2023, status: Processed, refund_issued: 620.00

## Input:

- Ciphertext:

  Wvzm slowh gsv kzhhklig mfnyvi Z87654321.

- Plaintext:

Figure 10: User Prompt (Atbash - 3shot - Letter & Number)

Table 11: Impact of Encryption Scope on Decryption Performance

Model	Rot13	Atbash	Polybius	DualAvgCode	ParityShift	$\text{Accuracy}_{\text{avg}}$
			Open-source Models			
DeepSeek-V3	68.94/23.32	24.02/14.64	19.35/6.01	3.51/0	11.31/0	25.23 / 8.79
DeepSeek-R1	59.10/43.05	63.19/23.02	39.21/43.23	37.36/0	13.05/0.76	42.38 / 22.01
			Closed-source Models			
GPT-4o-2024-11-20	27.53/0	10.08/0	0/0	2.54/0	2.67/0	8.56 / 0
gemini-2.0-flash-exp	47.54/0	7.50/2.50	7.50/5.05	0/0	2.67/0	13.04 / 1.51
Claude-Sonnet-3.5-1022	92.50/50.00	87.56/27.53	65.00/32.25	15.00/0	62.54/17.35	64.52 / 25.43
gemini-2.0-flash-thinking	35.00/2.65	0/2.54	0/10.00	0/0	2.50/0	7.50 / 3.04
o1-mini-2024-09-12	50.00/32.59	72.57/35.00	40.00/42.53	0/0	7.50/0.76	34.01 / 22.18

Note: Values before the '/' indicate performance when encrypting **letters only**, while values after the '/' represent performance when encrypting **both letters and numbers**.

formance, others exhibit significant drops when required to decrypt mixed alphanumeric ciphertexts.

For the enhanced prompt template, please refer to **Figures 11-19**, while more detailed experimental results can be found in **Table 12**.

C.3 Impact of Plaintext Source on Decryption Performance

To assess how plaintext characteristics influence decryption performance, we compare results on synthetically generated privacy-sensitive data ver-

Example C.4

## Background:

Decrypt the provided Ciphertext and return the corresponding Plaintext following the given algorithm flow and examples.

Think step by step.

Provide the Plaintext result in the format <result>text</result>, where text is the decrypted text.

## Algorithm Flow:

Uses the Caesar cipher with a fixed shift of 13 positions. For each letter in the Plaintext, shift it forward by 13 positions in the alphabet to produce the Ciphertext.

## Examples:

* Example 1:
  - Ciphertext:
    Anzr: Rzvyl Wbuafba; Qngr bs Ovegu: Whyl 15, 1990; Cnffcbeg Ahzore: L987654321
  - Plaintext:
    Name: Emily Johnson; Date of Birth: July 15, 1990; Passport Number: Y987654321
* Example 2:
  - Ciphertext:
    Pnfr Ahzore: 2024-CF-001234; Pnfr Glcr: Gursg/Oernx-Va; Svyvat Qngr: Bpgbore 19, 2024
  - Plaintext:
    Case Number: 2024-PS-001234; Case Type: Theft/Break-In; Filing Date: October 19, 2024
* Example 3:
  - Ciphertext:
    gnk_vqragvsvre: GKA-2023-NOP456, gnk_erpbeqf: lrne: 2023, fgnghf: Cebprffrq, ershaq_vffhrq: 620.00
  - Plaintext:
    tax_identifier: TXN-2023-ABC456, tax_records: year: 2023, status: Processed, refund_issued: 620.00

## Input:

- Ciphertext:
  Yrqvn, na Nzrevpna, erfvqrf va Ybf Natryrf.
- Plaintext:

Figure 11: Enhenced Prompt (ROT13 - 3shot - Letter)

Example C.5

## Background:

Decrypt the provided Ciphertext and return the corresponding Plaintext following the given algorithm flow and examples.

Think step by step.

Provide the Plaintext result in the format <result>text</result>, where text is the decrypted text.

## Algorithm Flow:

Uses the Atbash cipher. Each letter in the **Plaintext** is replaced with its reverse counterpart in the alphabet.

## Examples:

* Example 1:

- Ciphertext:

  Mznv: Vnrob Qlsmhlm; Wzgv lu Yrigs: Qfob 15, 1990; Kzhhklig Mfnyvi: B987654321

- Plaintext:

  Name: Emily Johnson; Date of Birth: July 15, 1990; Passport Number: Y987654321

* Example 2:

- Ciphertext:

  Xzhv Mfnyvi: 2024-KH-001234; Xzhv Gbkv: Gsvug/Yivzp-Rm; Urormt Wzgv: Lxglyvi 19, 2024

- Plaintext:

  Case Number: 2024-PS-001234; Case Type: Theft/Break-In; Filing Date: October 19, 2024

* Example 3:

- Ciphertext:

  gzc_rwvmgrurvi: GCM-2023-ZYX456, gzc_ivxliwh: bvzi: 2023, hgzgfh: Kilxvhhvw, ivufmw_rhhfvw: 620.00

- Plaintext:

  tax_identifier: TXN-2023-ABC456, tax_records: year: 2023, status: Processed, refund_issued: 620.00

## Input:

- Ciphertext:

  Ovwrz, zm Znvirxzm, ivhrwvh rm Olh Zmtvovh.

- Plaintext:

Figure 12: Enhenced Prompt (Atbash - 3shot - Letter)

Example C.6

## Background:

Decrypt the provided Ciphertext and return the corresponding Plaintext following the given algorithm flow and examples.

Think step by step.

Provide the Plaintext result in the format <result>text</result>, where text is the decrypted text.

## Algorithm Flow:

Uses the Polybius cipher. Each letter in the **Plaintext** is mapped to a pair of coordinates in the Polybius square, forming the **Ciphertext**.

## Examples:

* Example 1:

- Ciphertext:

32 11 31 15 : 15 31 23 26 51 24 33 22 32 41 33 32 ; 14 11 42 15 33 16 12 23 36 42 22 : 24 43 26 51 1 5 , 1 9 9 0 ; 34 11 41 41 34 33 36 42 32 43 31 12 15 36 : 51 9 8 7 6 5 4 3 2 1

- Plaintext:

Name: Emily Johnson; Date of Birth: July 15, 1990; Passport Number: Y987654321

* Example 2:

- Ciphertext:

13 11 41 15 32 43 31 12 15 36 : 2 0 2 4 - 34 41 - 0 0 1 2 3 4 ; 13 11 41 15 42 51 34 15 : 42 22 15 16 42 / 12 36 15 11 25 - 23 32 ; 16 23 26 23 32 21 14 11 42 15 : 33 13 42 33 12 15 36 1 9 , 2 0 2 4

- Plaintext:

Case Number: 2024-PS-001234; Case Type: Theft/Break-In; Filing Date: October 19, 2024

* Example 3:

- Ciphertext:

42 11 46 _ 23 14 15 32 42 23 16 23 15 36 : 42 46 32 _ 2 0 2 3 - 11 12 13 4 5 6 , 42 11 46 _ 36 15 13 33 36 14 41 : 51 15 11 36 : 2 0 2 3 , 41 42 11 42 43 41 : 34 36 33 13 15 41 41 15 14 , 36 15 16 43 32 14 _ 23 41 41 43 15 14 : 6 2 0 . 0 0

- Plaintext:

tax_identifier: TXN-2023-ABC456, tax_records: year: 2023, status: Processed, refund_issued: 620.00

## Input:

- Ciphertext:

26 15 14 23 11 , 11 32 11 31 15 36 23 13 11 32 , 36 15 41 23 14 15 41 23 32 26 33 41 11 32 21 15 26 15 41 .

- Plaintext:

Figure 13: Enhanced Prompt (Polybius - 3shot - Letter)

Example C.7

## Background:

Decrypt the provided Ciphertext and return the corresponding Plaintext following the given algorithm flow and examples.

Think step by step.

Provide the Plaintext result in the format <result>text</result>, where text is the decrypted text.

## Algorithm Flow:

Uses the Vigenère cipher. Each letter in the **Plaintext** is shifted by the corresponding letter in the **Key** to produce the **Ciphertext**.

## Examples:

* Example 1:
  - Ciphertext:
    Ncxe: Eotla Jqsnuzn; Dcee zf Miteh: Jwwy 15, 1990; Pcdsrzrv Nwxbgc: J987654321
  - Plaintext:
    Name: Emily Johnson; Date of Birth: July 15, 1990; Passport Number: Y987654321
* Example 2:
  - Ciphertext:
    Ccde Yuomet: 2024-PU-001234; Naup Vjpg: Vsehe/Dcecv-Ky; Qintni Dcee: Oeeodpr 19, 2024
  - Plaintext:
    Case Number: 2024-PS-001234; Case Type: Theft/Break-In; Filing Date: October 19, 2024
* Example 3:
  - Ciphertext:
    tci_koepeihtet: VIN-2023-CMC456, tci_tpcqcdu: jecc: 2023, dtceuu: Rcoepsupd, rgqupo_kdswpd: 620.00
  - Plaintext:
    tax_identifier: TXN-2023-ABC456, tax_records: year: 2023, status: Processed, refund_issued: 620.00

## Input:

- Ciphertext:
  Lgoic, cy Cxettccy, ceutdgd ky Nzs Lniplgd.
- Plaintext:

Figure 14: Enhanced Prompt (Vigenère - 3shot - Letter)

Example C.8

## Background:

Decrypt the provided Ciphertext and return the corresponding Plaintext following the given algorithm flow and examples.

Think step by step.

Provide the Plaintext result in the format <result>text</result>, where text is the decrypted text.

## Algorithm Flow:

Reverses the **Plaintext** to create the **Ciphertext**.

## Examples:

* Example 1:
    - Ciphertext:
        123456789Y :rebmuN tropssaP ;0991 ,51 yluJ :htriB fo etaD ;nosnhoJ ylimE :emaN
    - Plaintext:
        Name: Emily Johnson; Date of Birth: July 15, 1990; Passport Number: Y987654321
* Example 2:
    - Ciphertext:
        4202 ,91 rebotcO :etaD gniliF ;nI-kaerB/tfehT :epyT esaC ;432100-SP-4202 :rebmuN esaC
    - Plaintext:
        Case Number: 2024-PS-001234; Case Type: Theft/Break-In; Filing Date: October 19, 2024
* Example 3:
    - Ciphertext:
        }00.026 :deussi_dnufer ,dessecorP :sutats ,3202 :raey{ :sdrocer_xat ,654CBA-3202-NXT :reifitnedi_xat
    - Plaintext:
        tax_identifier: TXN-2023-ABC456, tax_records: year: 2023, status: Processed, refund_issued: 620.00

## Input:

- Ciphertext:
    .selegnA soL ni sediser ,naciremA na ,aideL
- Plaintext:

Figure 15: Enhenced Prompt (Reverse - 3shot - Letter)

Example C.9

## Background:

Decrypt the provided Ciphertext and return the corresponding Plaintext following the given algorithm flow and examples.
Think step by step.
Provide the Plaintext result in the format <result>text</result>, where text is the decrypted text.

## Algorithm Flow:

For each pair of letters in the **Plaintext**, their positions are swapped to produce the **Ciphertext**. If the number of letters is odd, the last letter remains in its original position.

## Examples:

* Example 1:
    - Ciphertext:
        aNem :mEli yoJnhos;nD ta efoB riht :uJyl1 ,51 99;0P sapsro tuNbmre :9Y78563412
    - Plaintext:
        Name: Emily Johnson; Date of Birth: July 15, 1990; Passport Number: Y987654321
* Example 2:
    - Ciphertext:
        aCesN mueb:r2 20-4SP0-1032;4C sa eyTep :hTfe/trBae-knI ;iFilgnD ta:eO tcbore1 ,92 204
    - Plaintext:
        Case Number: 2024-PS-001234; Case Type: Theft/Break-In; Filing Date: October 19, 2024
* Example 3:
    - Ciphertext:
        at_xdineitifre :XT-N0232A-CB54,6t xar_cerosd :yae:r2 20,3s atut:sP orecssde ,erufdni_sseu:d6 020.0
    - Plaintext:
        tax_identifier: TXN-2023-ABC456, tax_records: year: 2023, status: Processed, refund_issued: 620.00

## Input:

- Ciphertext:
    eLid,aa nmArecina ,erised sniL soA gnlese.
- Plaintext:

Figure 16: Enhenced Prompt (SwapPairs - 3shot - Letter)

Example C.10

## Background:

Decrypt the provided Ciphertext and return the corresponding Plaintext following the given algorithm flow and examples.

Think step by step.

Provide the Plaintext result in the format <result>text</result>, where text is the decrypted text.

## Algorithm Flow:

This encryption method converts each letter of the **Plaintext** into two letters in the **Ciphertext**, such that the average of their ASCII values equals the ASCII value of the original letter.

## Examples:

* Example 1:

- Ciphertext:

MOaalndf: DFlnhjkmxz IKnpgimortnpmo; CEaasudf npeg AChjqssugi: IKtvkmxz 15, 1990; OQaartrtoqn-pqssu MOtvlnacdfqs: XZ987654321

- Plaintext:

Name: Emily Johnson; Date of Birth: July 15, 1990; Passport Number: Y987654321

* Example 2:

- Ciphertext:

BDaartdf MOtvlnacdfqs: 2024-OQRT-001234; BDaartdf SUxzoqdf: SUgidfegsu/ACqsdfaajl-HJmo; EGhjkmhjmofh CEaasudf: NPbdsunpacdfqs 19, 2024

- Plaintext:

Case Number: 2024-PS-001234; Case Type: Theft/Break-In; Filing Date: October 19, 2024

* Example 3:

- Ciphertext:

suaawy_hjcedfmosuhjeghjdfqs: SUWYMO-2023-AAACBD456, suaawy_qsdfbdnpqscert: xzdfaaqs: 2023, rtsuaasutvrt: OQqsnpbddfrtrtdfce, qsdfegtvmoce_hjrtrttvdfce: 620.00

- Plaintext:

tax_identifier: TXN-2023-ABC456, tax_records: year: 2023, status: Processed, refund_issued: 620.00

## Input:

- Ciphertext:

KMdfcehjaa, aamo AAlndfqshjbdaamo, qsdfrthjcedfrt hjmo KMnprt AAmofhdfkmdfrt.

- Plaintext:

Figure 17: Enhenced Prompt (DualAvgCode - 3shot - Letter)

Example C.11

## Background:

Decrypt the provided Ciphertext and return the corresponding Plaintext following the given algorithm flow and examples.

Think step by step.

Provide the Plaintext result in the format <result>text</result>, where text is the decrypted text.

## Algorithm Flow:

For each letter in the **Plaintext**: - If the ASCII value is even, add 1 to it to get the corresponding character in the **Ciphertext**. - If the ASCII value is odd, subtract 1 to get the new character in the **Ciphertext**.

## Examples:

* Example 1:

- Ciphertext:

  Ozld: Dlhmx Kniorno; Ezud ng Chsui: Ktmx 15, 1990; Qzrrqnsu Otlcds: X987654321

- Plaintext:

  Name: Emily Johnson; Date of Birth: July 15, 1990; Passport Number: Y987654321

* Example 2:

- Ciphertext:

  Bzrd Otlcds: 2024-QR-001234; Bzrd Uxqd: Uidgu/Csdzj-Ho; Ghmhof Ezud: Nbuncds 19, 2024

- Plaintext:

  Case Number: 2024-PS-001234; Case Type: Theft/Break-In; Filing Date: October 19, 2024

* Example 3:

- Ciphertext:

  uzy_hedouhghds: UYO-2023-ZCB456, uzy_sdbnser: xdzs: 2023, ruzutr: Qsnbdrrde, sdgtoe_hrrtde: 620.00

- Plaintext:

  tax_identifier: TXN-2023-ABC456, tax_records: year: 2023, status: Processed, refund_issued: 620.00

## Input:

- Ciphertext:

  Mdehz, zo Zldshbzo, sdrhedr ho Mnr Zofdmdr.

- Plaintext:

Figure 18: Enhenced Prompt (ParityShift - 3shot - Letter)

Example C.12

## Background:

Decrypt the provided Ciphertext and return the corresponding Plaintext following the given algorithm flow and examples.

Think step by step.

Provide the Plaintext result in the format <result>text</result>, where text is the decrypted text.

## Algorithm Flow:

The algorithm splits the **Plaintext** into words based on spaces. Each word is then individually encrypted using the Caesar cipher, resulting in the **Ciphertext**.

## Examples:

* Example 1:
  - Ciphertext:
    e:Nam lyEmi nson;Joh eDat fo th:Bir yJul 15, 0;199 sportPas ber:Num 7654321Y98
  - Plaintext:
    Name: Emily Johnson; Date of Birth: July 15, 1990; Passport Number: Y987654321
* Example 2:
  - Ciphertext:
    eCas ber:Num 4-PS-001234;202 eCas e:Typ ft/Break-In;The ingFil e:Dat oberOct 19, 4202
  - Plaintext:
    Case Number: 2024-PS-001234; Case Type: Theft/Break-In; Filing Date: October 19, 2024
* Example 3:
  - Ciphertext:
    _identifier:tax -2023-ABC456,TXN _records:tax ar:ye 3,202 tus:sta cessed,Pro und_issued:ref .00620
  - Plaintext:
    tax_identifier: TXN-2023-ABC456, tax_records: year: 2023, status: Processed, refund_issued: 620.00

## Input:

- Ciphertext:
  ia,Led na rican,Ame idesres ni Los eles.Ang
- Plaintext:

Figure 19: Enhenced Prompt (WordShift - 3shot - Letter)

Table 12: Results on CipherBank(Enhanced Prompt)

Model	Substitution Ciphers				Transposition Ciphers		Custom Ciphers			Cipher Score
	Rot13	Atbash	Polybius	Vigenère	Reverse	SwapPairs	DualAvgCode	ParityShift	WordShift	$\text{Accuracy}_{\text{avg}}$
Open-source Chat Models										
Mixtral-8x22B-v0.1	0.76	0	0	0	0.38	0	2.67	0.38	0.38	0.51
Qwen2.5-72B-Instruct	12.60	9.16	0	0	0	0	2.29	0.38	1.53	2.88
Llama-3.1-70B-Instruct	2.67	1.15	0	0	1.53	0.38	1.15	0	0	0.76
Llama-3.3-70B-Instruct	4.58	1.53	0	0.38	1.15	0	1.15	0	0	0.98
DeepSeek-V3	41.60	27.86	0.38	0.38	65.95	5.34	12.66	0.76	5.17	17.79
Closed-source Models										
GPT-4o-mini-2024-07-18	21.76	19.08	0	0.38	4.39	0	0	0	0	5.07
GPT-4o-2024-08-06	45.42	24.05	0	0.76	51.53	8.40	1.91	1.15	10.31	15.95
GPT-4o-2024-11-20	45.42	41.98	0	0	53.63	8.02	3.82	1.15	9.54	18.17
gemini-1.5-pro	63.69	5.73	0.76	0.38	14.12	2.67	0.38	1.91	10.69	11.15
gemini-2.0-flash-exp	45.04	22.90	2.29	0.38	46.56	4.58	3.82	0	1.15	14.08
Claude-Sonnet-3.5-1022	**92.75**	82.06	**78.24**	**2.48**	**79.39**	9.73	2.48	62.02	**44.85**	50.44
Reasoning Models										
QwQ-32B-Preview	1.91	3.05	2.67	0	0	0	2.67	0.38	0.38	1.23
DeepSeek-R1	88.37	**86.54**	72.73	0.76	46.96	**75.01**	**73.17**	**74.42**	1.51	**57.72**
gemini-2.0-flash-thinking	37.98	19.09	10.50	0	55.34	4.96	4.77	0.38	6.11	15.46
o1-mini-2024-09-12	54.20	72.14	50.0	0.76	11.07	18.70	47.33	49.62	7.25	34.56

sus externally sourced structured text (e.g., quotes from Shakespeare's works). The structured text exhibits greater linguistic familiarity, while the privacy-sensitive data represents real-world encryption needs, lacking inherent semantic patterns.

As shown in **Table 13** and **Table 14,** models generally perform better on structured text, suggesting that they leverage linguistic priors rather than strictly following decryption rules. When encountering encrypted text with recognizable patterns, models tend to shortcut reasoning, aligning decoded fragments with plausible linguistic structures instead of strictly adhering to learned transformation rules. Conversely, for less structured, domain-specific text, models struggle to infer decryption patterns, reinforcing the advantage of CipherBank's privacy-sensitive dataset, which forces models to engage in independent reasoning rather than rely on pretraining biases.

D Error Analysis

D.1 Error Classification

This section defines the error categories observed in model decryption outputs. These classifications help identify systematic failure patterns and provide insights into how models approach cryptographic reasoning.

- (A) Omission/Insertion: The model output contains missing or extra characters, words, or punctuation compared to the reference plaintext. These errors indicate incomplete decryption or unintended modifications, leading to partial but inaccurate results.
- (B) Name Decryption Error: The decryption result is correct except for the name part, which remains incorrect or partially distorted. This suggests challenges in handling named entities, possibly due to memorization effects or entity-based biases.
- (C) Semantic Inference: The model makes errors based on semantic reasoning rather than strictly following decryption rules. Instead of decoding symbols precisely, the model hallucinates plausible but incorrect outputs that fit the general meaning of the sentence. This indicates a tendency to prioritize linguistic coherence over strict decryption fidelity.
- (D) Reorganization: The output preserves the exact meaning of the reference plaintext but rearranges the sentence structure. This suggests that the model prioritizes fluency over strict character-level fidelity, leading to errors in cryptographic tasks where precision is essential.
- (E) Reasoning Failure: The model output is significantly different from the reference, and decryption is essentially unsuccessful. This suggests a fundamental failure in identifying encryption patterns, leading to outputs that bear little resemblance to the expected plaintext. This category includes cases where the model fails to infer transformation rules or apply correct decryption strategies.

Table 13: Decryption Performance on Privacy-Sensitive Data

Model	Rot13	Atbash	Polybius	Vigenère	Reverse	Swap	DualAvgCode	ParityShift	WordShift	Accuracy$_{\text{avg}}$
					Open-source Models					
DeepSeek-V3	24.34	15.64	15.70	0	33.72	3.51	0	4.35	15.64	12.54
DeepSeek-R1	57.88	71.02	71.55	4.35	33.57	4.35	0	12.71	8.70	29.35
					Closed-source Models					
GPT-4o-2024-11-20	21.74	21.74	0	0	30.43	8.70	0	0	13.04	10.63
Gemini-2.0-Flash-Exp	47.83	4.35	4.35	0	52.17	0	4.35	4.35	13.04	14.49
Claude-Sonnet-3.5-1022	86.96	78.26	65.22	4.35	91.30	13.04	4.35	52.17	47.83	49.28
Gemini-2.0-Flash-Thinking	39.13	4.35	0	0	60.87	0	0	4.35	30.43	15.46
o1-Mini-2024-09-12	60.87	86.96	69.57	0	8.70	0	13.04	17.39	4.35	28.99

Table 14: Decryption Performance on Structured Text

Model	Rot13	Atbash	Polybius	Vigenère	Reverse	SwapPair	DualAvgCode	ParityShift	WordShift	Accuracy$_{\text{avg}}$
					Open-source Models					
DeepSeek-V3	76.12	24.03	15.70	0	52.17	29.40	0	12.71	55.13	29.47
DeepSeek-R1	84.51	85.04	100	7.59	79.10	8.70	8.70	15.64	30.43	46.63
					Closed-source Models					
GPT-4o-2024-11-20	78.26	39.13	4.35	0	86.96	21.74	0	4.35	43.48	30.92
Gemini-2.0-Flash-Exp	86.96	13.04	4.35	0	86.96	8.70	0	17.39	43.48	28.99
Claude-Sonnet-3.5-1022	91.30	95.65	95.65	4.35	100	52.17	8.70	78.26	95.65	69.08
Gemini-2.0-Flash-Thinking	86.96	13.04	8.70	0	69.57	17.39	0	0	52.17	27.54
o1-Mini-2024-09-12	82.61	95.65	78.26	0	60.87	4.35	13.04	17.39	43.48	43.96

- (F) Other: Miscellaneous errors that do not fit into the defined categories.

This classification framework provides a structured approach to analyzing decryption errors, helping to pinpoint systematic weaknesses and guide future improvements in cryptographic reasoning models.

D.2 Examples of Different Error Types

To further illustrate the types of decryption errors encountered in our evaluation, we provide concrete examples corresponding to each error category. These cases demonstrate how models fail in various aspects of decryption, including omission/insertion, name decryption errors, semantic inference, reorganization, reasoning failures, and other anomalies. **Example D.1 - D6** showcase representative examples of each error type.

D.3 Detailed Error Distribution Tables

Tables 15–20 present a detailed breakdown of error distributions across different encryption algorithms for the six selected models. From these results, we identify several common trends and model-specific differences.

Challenges in Name Decryption and Symbolic Reasoning. Across all models, name decryption errors remain prevalent, particularly in Atbash and Polybius, indicating persistent difficulties in handling entity-based transformations. Additionally, models struggle with key-based and transposition ciphers such as Vigenère and SwapPairs, suggesting limitations in tracking multi-step transformations and generalizing decryption strategies.

Semantic Overreliance vs. Overthinking in Decryption. Chat models often exhibit semantic inference errors, where decrypted outputs align with linguistic patterns rather than encryption rules. In contrast, reasoning models tend to overthink simple tasks, leading to unnecessary self-correction loops that degrade performance in straightforward ciphers like Reverse.

Structural Alignment and Insertion Errors. Frequent omission and insertion errors in WordShift and Reverse ciphers highlight difficulties in preserving character order. This suggests that models rely on semantic priors rather than strict symbolic reasoning, leading to misaligned outputs.

Key Takeaways:

- Chat models (Claude, Gemini) perform well in substitution ciphers but struggle with complex rule-based encryption.
- Reasoning models (DeepSeek-R1, o1) maintain better structural accuracy but underperform in transposition-based and key-dependent ciphers.

Example D.1: Error Type: Omission/Insertion

Example 1:
Predictions: Card Number: ID 1245-6789-0123
References: Clark holds the ID Card Number 1245-6789-0123.

Example 2:
Predictions: Card Number: ID 1245-6789-0123
References: Clark holds the ID Card Number 1245-6789-0123.

Example 3:
Predictions: Salary Amount: $67,000; Pay Date: 2023-10-25
References: Salary Amount: $67,000, Pay Date: 2023-10-25.

Example D.2: Error Type: Name Decryption Error

Example 1:
Predictions: Learn, an American, inside on Los Angeles.
References: Ledia, an American, resides in Los Angeles.

Example 2:
Predictions: Individual ID: A1234567; Name: John Doe; Age: 34; Gender Identity: Cisgender 16k11.2 Location, Country State Citizenship.
References: Individual ID: A1234567; Name: Jane Doe; Age: 34; Genetic Testing: Chromosome 16p11.2 Deletion, Celiac Disease Predisposition.

Example 3:
Predictions: Handed lost the passport Number A12345678.
References: Dean holds the passport number A12345678.

Example D.3: Error Type: Semantic Inference

Example 1:
Predictions: Jessica Brown, Bachelor of Biology, GPA: 3.9, Graduated 2023, Skills: Genetics, Microbiology, Ecology, Bioinformatics.
References: Jessica Brown, Bachelors in Biology, GPA: 3.9, Graduated 2023, Courses: Genetics, Microbiology, Ecology, Biochemistry.

Example 2:
Predictions: Patent-pending design specification PR2023_KURITY, Company Z, including batch production requirements.
References: Patent-pending design specification PR2023_KURITY, Company Z, including batch performance analysis.

Example 3:
Predictions: Office ID: P12345, Name: John, Position: Manager, Department: Finance.
References: Officer ID: P12345, Name: John, Position: Sergeant, Department: Homicide.

Example D.4: Error Type: Reorganization

Example 1:
Predictions: Name: Ledia; Nationality: American; Address: Los Angeles
References: Ledia, an American, resides in Los Angeles.

Example 2:
Predictions: Passport Number: A12345678; Holder Name: Dean
References: Dean holds the passport number A12345678.

Example 3:
Predictions: Name: Peter; Date of Birth: April 23, 1985; Passport Number: X123456789
References: Peter was born on April 23, 1985, and carries a passport with the number X123456789.

Example D.5: Error Type: Reasoning Failure

Example 1:
Predictions: Address: 123 Main St, Apt 4B, New York, NY, Zip Code: 10001, Phone: 555-1234, Unit: 101, Floor: 2.
References: Officer: Lisa Grant, Sergeant, Downtown Precinct, Patrol Car: VG2301, Equipment: Radar Gun, Model: RGX501, Weapon: Taser X2.

Example 2:
Predictions: Welcome, once more, securely within.
References: Ledia, an American, resides in Los Angeles.

Example 3:
Predictions: Passport ID: 123456789; Expiry Date: Emily, 38, Issued By Authority, Renewal By Agency. Valid Until 2025, Expiry Passport.
References: Patient ID: MRN456789; Medical History: Emily, 38, diagnosed with asthma, treated with inhalers. Allergic to dust mites, pollen.

Example D.6: Error Type: Other

Example 1:
Predictions: CookieID12345 maintain login status for UserID98765 on www.example.com, facilitating seamless access. Analyzing records UserID98765's engagement, deducting 500 page views and a click-through rate of 4.5% across the session.
References: CookieID12345 maintains login status for UserID98765 on www.example.com, facilitating seamless access. Analytics tracks UserID98765's engagement, documenting 500 page views and a click-through rate of 4.5% across the session.

Example 2:
Predictions: Code: Coordinates: Latitude Longitude: 38.251° N, -85.754° W, Latitude Longitude: 34.091° N, -118.493° W.
References: Base Distribution: North Plains Base: 38.251° N, -85.754° W, East Valley Site: 34.091° N, -118.493° W.

Example 3:
Predictions: Name: Alex Smith; Salary: $87,500; Pay Frequency: Biweekly; Position: Software Developer; Employee ID: EID-257846; Department: IT.
References: Name: Alex Smith, Salary: $87,500, Pay Frequency: Biweekly, Position: Software Developer, Employee ID: EID-257846, Department: IT.

Table 15: Error Type Percentages for Different Algorithms in Claude-Sonnet-3.5-1022 Model

Algorithm	Error Types					
	Omission/Insertion	Name Decryption Error	Semantic Inference	Reorganization	Reasoning Failure	Other
Rot13	33.33	51.85	0.00	11.11	3.70	0.00
Atbash	15.79	78.95	0.00	3.51	0.00	1.75
Polybius	42.62	45.90	0.00	11.48	0.00	0.00
Vigenère	2.73	32.42	5.08	3.52	56.25	0.00
Reverse	39.24	48.10	0.00	5.06	6.33	1.27
SwapPairs	15.98	38.52	2.05	2.87	38.11	2.46
DualAvgCode	6.88	39.68	8.50	2.43	41.30	1.21
ParityShift	19.79	70.83	4.17	3.12	2.08	0.00
WordShift	51.95	22.08	2.60	8.44	12.34	2.60

Table 16: Error Type Percentages for Different Algorithms in DeepSeek-R1 Model

Algorithm	Error Types					
	Omission/Insertion	Name Decryption Error	Semantic Inference	Reorganization	Reasoning Failure	Other
Rot13	40.00	30.00	4.29	21.43	1.43	2.86
Atbash	42.59	24.07	0.93	29.63	0.00	2.78
Polybius	48.63	17.12	0.68	21.92	8.90	2.74
Vigenère	4.60	18.01	2.68	2.30	71.65	0.77
Reverse	25.64	19.66	1.71	45.30	6.41	1.28
SwapPairs	9.20	25.29	3.07	2.30	58.62	1.53
DualAvgCode	25.63	22.61	3.52	28.64	19.10	0.50
ParityShift	7.02	29.39	6.58	3.95	52.19	0.88
WordShift	29.17	22.92	2.08	25.42	20.00	0.42

Table 17: Error Type Percentages for Different Algorithms in DeepSeek-V3 Model

Algorithm	Error Types					
	Omission/Insertion	Name Decryption Error	Semantic Inference	Reorganization	Reasoning Failure	Other
Rot13	10.73	55.93	15.82	5.08	11.86	0.56
Atbash	8.07	38.12	7.17	3.59	41.26	1.79
Polybius	5.47	12.11	2.34	2.73	76.95	0.39
Vigenère	0.38	20.77	2.69	0.77	74.23	1.15
Reverse	21.50	40.19	5.61	13.55	18.22	0.93
SwapPairs	1.92	18.39	2.68	0.38	76.25	0.38
DualAvgCode	3.07	12.64	3.45	2.68	77.78	0.38
ParityShift	1.93	28.57	3.86	0.77	64.48	0.39
WordShift	27.80	29.46	4.56	17.01	20.33	0.83

Table 18: Error Type Percentages for Different Algorithms in gemini-1.5-pro Model

Algorithm	Error Types					
	Omission/Insertion	Name Decryption Error	Semantic Inference	Reorganization	Reasoning Failure	Other
Rot13	12.98	58.02	0.76	5.34	22.14	0.76
Atbash	1.15	15.00	3.08	0.77	78.85	1.15
Polybius	4.21	17.24	3.07	1.92	71.65	1.92
Vigenère	2.29	14.89	3.44	0.76	78.63	0.00
Reverse	20.85	33.19	8.94	10.21	26.38	0.43
SwapPairs	6.49	25.57	1.91	1.53	63.36	1.15
DualAvgCode	2.68	13.03	4.60	1.92	77.39	0.38
ParityShift	3.08	28.46	3.08	0.38	64.23	0.77
WordShift	34.25	24.20	2.74	18.72	19.63	0.46

- All models show high name decryption errors and reasoning failures in Vigenère and SwapPairs, highlighting gaps in symbolic reasoning and long-term dependency tracking.

These observations reveal that no single model excels across all ciphers, emphasizing the need for advancements in structured reasoning and symbolic manipulation for decryption tasks. **Future**

Table 19: Error Type Percentages for Different Algorithms in o1-mini Model

Algorithm	Error Types					
	Omission/Insertion	Name Decryption Error	Semantic Inference	Reorganization	Reasoning Failure	Other
Rot13	26.95	38.30	13.48	17.02	1.42	2.84
Atbash	37.35	31.33	7.23	16.87	6.02	1.20
Polybius	30.94	32.37	1.44	25.18	8.63	1.44
Vigenère	0.00	21.43	10.71	3.57	64.29	0.00
Reverse	12.70	29.10	8.20	32.38	17.21	0.41
SwapPairs	1.91	9.54	1.53	0.00	86.64	0.38
DualAvgCode	0.00	18.52	0.00	3.70	77.78	0.00
ParityShift	4.55	34.30	3.31	4.96	52.48	0.41
WordShift	11.58	28.57	4.63	5.79	49.03	0.39

Table 20: Error Type Percentages for Different Algorithms in o1 Model

Algorithm	Error Types					
	Omission/Insertion	Name Decryption Error	Semantic Inference	Reorganization	Reasoning Failure	Other
Rot13	16.19	28.57	4.76	5.71	43.81	0.95
Atbash	29.09	49.09	5.45	10.91	3.64	1.82
Polybius	40.91	28.79	6.06	10.61	12.12	1.52
Vigenère	4.62	36.15	1.54	1.15	56.15	0.38
Reverse	16.14	25.56	3.59	14.35	38.57	1.79
SwapPairs	5.26	31.58	5.26	5.26	52.63	0.00
DualAvgCode	24.62	33.85	3.08	2.31	35.38	0.77
ParityShift	4.04	26.77	4.55	2.02	62.12	0.51
WordShift	30.88	24.26	2.94	18.38	21.32	2.21

improvements could focus on:

- Minimizing the Impact of Semantic Bias in Logical Inference: Cryptographic reasoning tasks often necessitate abstract rule extraction rather than reliance on semantic interpretation. An excessive dependence on linguistic priors can impede the model's ability to identify underlying structural transformations, resulting in systematic errors. Future advancements should focus on reducing semantic interference to improve the extraction of abstract logical patterns.

- Enhancing Comparative Reasoning for Pattern Recognition: While many decryption tasks in CipherBank are straightforward for humans, models frequently fail to derive correct transformation rules from provided exemplars. Strengthening contrastive reasoning mechanisms can enable models to better differentiate encryption structures, facilitating more effective pattern recognition and decryption.

- Addressing Overthinking in Model Reasoning: Experimental results indicate that reasoning models exhibit superior performance on complex tasks but underperform on simpler problems. Analysis of inference trajectories reveals a tendency toward recursive self-evaluation, where models continuously revise their approach, even when a straightforward solution is available. For example, in the Reverse cipher, models occasionally attempt unnecessarily complex reasoning paths instead of applying direct positional transformations. Mitigating such overthinking behaviors could enhance efficiency and robustness in logical reasoning.

Addressing these limitations will bridge the gap between linguistic fluency and structured cryptographic reasoning, making LLMs more robust in real-world encryption scenarios.