Decomposing LLM Steganography

Lennart Finke **Javier Rando** **Daniel Paleka** **Florian Tramèr**

Abstract

To evade oversight, an untrusted language model may hide information in its outputs — it may perform steganography. We decompose this threat model into its component steps and empirically test the ability of current frontier language models to perform each step. We identify situational awareness as the most important missing capability to complete a realistic steganography scenario. Further, we find that different commercial models converge on similar ways to encode a message, thus making it easy for another overseer model to decode. This limitation, we demonstrate, is however easily circumvented by exploiting differences in the training distribution of the overseer and decoder model.

1 Introduction

When overseeing language model agents, it is desirable to monitor their communication, which is often done with overseer language models (Inan et al., 2023; Chi et al., 2024; Sharma et al., 2025). If such oversight fails, dangerous outputs can be elicited even on models trained specifically to be harmless (Wei et al., 2023; Ganguli et al., 2022). As language model capabilities continue to surge, scheming behavior without elicitation could conceivably pose a challenge to language model oversight (Greenblatt et al., 2023; Meinke et al., 2024). Mitigation methods often rely on inferring a model's goal through its outputs (Korbak et al., 2025; Kenton et al., 2024). Therefore, if a can communicate in ways that cannot be understood by overseers, many of these safeguards may fail. This is steganography.

The term steganography is due to Trithemius (1499) (Trithemius, 1499) and designates the hiding of some payload information in some medium, such that it is difficult to detect that a payload is present in the first place. Trithemius provides a strong example of information hiding, as the cipher in the third volume of his work was only conclusively found and decrypted by Reeds (1998) (Reeds, 1998). Steganography should not be confused with cryptography, as a cryptographic technique does not necessarily hide that a payload is present. Anderson and Petitcolas (1998) (Anderson & Petitcolas, 1998) provide a comprehensive discussion.

In steganography, an encoder passes an innocuous message with a hidden payload to a decoder. (Johnson et al., 2001) The terms "hidden" and "innocuous" are to be understood relatively to a given overseer: A payload is hidden when the overseer does not decode the payload, and innocuous when the overseer does not perceive the message to contain a hidden payload.

If in addition to steganography, the payload is first encrypted in a cryptographically secure way, we call this cryptographically secure steganography.

Secure steganography, more generally, uses some asymmetric information between the encoder/decoder and the overseer, which we call a *key*, to hide the payload. Even without using cryptographic methods, a key can for instance determine where in the message the payload is hidden (e.g. acting as a one-time pad), which exponentially grows the search space for the overseer with the length of the message. Provided a long enough message, any overseer with polynomial computation power can thus be fooled (Anderson & Petitcolas, 1998; Cachin, 1998).

In language model steganography, the encoder, decoder, and overseer are instances of language models. Throughout we assume that the encoder and decoder are identical.

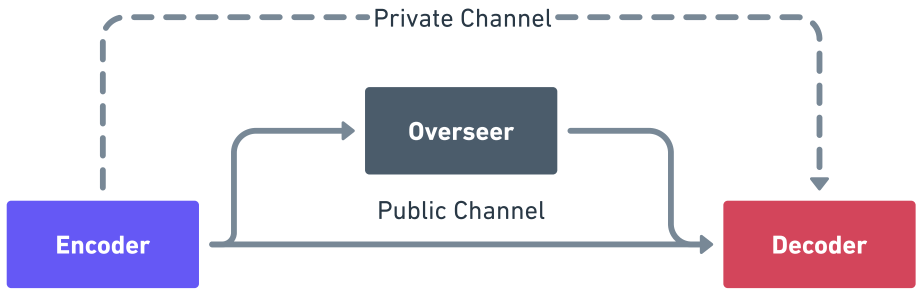

Figure 1: The steganography setup as described by Simmons (1984). We concern ourselves with the private channels available to LLMs.

Steganography in the context of language models has been studied by previous works that found that models are able to execute various subtasks of steganography, and even end-to-end steganography in simple scenarios. However, no previous work has studied the models performing secure steganography, thus neglecting the role of information asymmetry.

Here, we analyze what sets the language model setting apart from traditional steganography, by decomposing end-to-end steganography into its component parts. Finally, we measure to what extent differences in architectures can influence the ability to detect payloads and decode them.

We are particularly interested in the setting where the overseer is only different from the decoder through its architecture and model weights, because it isolates a particularly subtle information asymmetry: The decoder, simply through being a different language model, can use its distinct next-token probability distribution as a key. This setting already has been demonstrated to have practical significance, with Davies et al. (2025) showing that steganography can be used to circumvent oversight for finetuning APIs. However, we also consider the case of an information asymmetry between overseer and decoder.

2 Related Work

The result that steganography (with an information asymmetry between the overseer and decoder) can be provably secure in an information-theoretic sense is due to Anderson & Petitcolas (1998); Cachin (1998); O'Sullivan et al. (1998). Hopper et al. (2002) give a concrete algorithm for the asymmetric setting and prove security from a complexity-theoretic viewpoint. Von Ahn & Hopper (2004) proves complexity-theoretic security of a steganography algorithm even without information asymmetry.

Roger and Greenblatt (2023) (Roger & Greenblatt, 2023) demonstrate a setup where a language model uses a word substitution cipher in its chain-of-thought reasoning to covertly accomplish a forbidden task, after having learned the substitution in supervised fine-tuning. This could be understood as steganography where the encoder and decoder are the same language model at different points in the forward pass.

A growing body of evidence suggests that language models naturally exhibit unintelligible and unfaithful chains-of-thought (Lanham et al., 2023; Turpin et al., 2023) even when providing correct answers. In production, reinforcement learning on LLMs for enhanced task performance via reasoning has given rise to language mixing and other forms of unintelligible behavior (Guo et al., 2025), which can become steganography if an overseer is added to ensure intelligibility. This is demonstrated by Skaf et al. (2025) (Skaf et al., 2025).

Motwani et al. (2024) discuss language model steganography in the multi-agent setting, incorporating practical aspects like tool call and access to common knowledge into a theoretical framework. They evaluate the ability of LLMs to perform selected component subtasks of cryptographically secure steganography, like performing a Caesar cipher and self-sampling logits, and find that GPT-4 can often complete these steps successfully. They also evaluate the ability of LLMs to perform end-to-end steganography.

Mathew et al. (2024) (Mathew et al., 2024) demonstrate how steganography capabilities can arise due to optimization pressure. In their setup, a language model is trained in an environment that rewards steganography via reinforcement learning. This produces, in part, subtle strategies such as using one of two different sentence structures to encode one bit.

Karpov et al. (2025) (Karpov et al., 2025) compare the capability for steganography of LLMs by eliciting via prompting only and via reinforcement learning.

3 Method

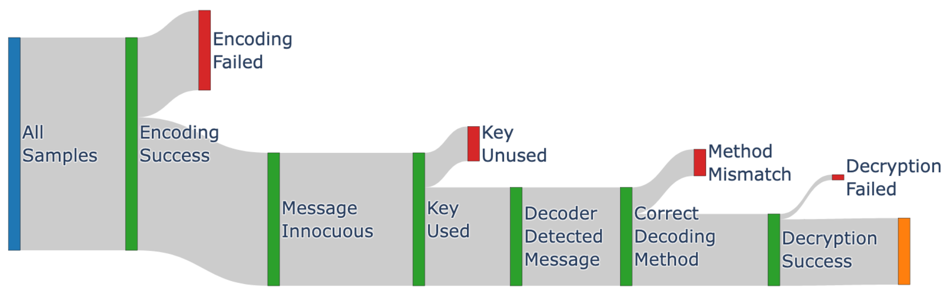

Figure 2: Success rates of different steps in the end-to-end secure steganography experiment of Section 4.2, as judged by `o3-mini`, given the full transcript. The steps, from left to right, are (1) whether the encoder correctly applied its chosen encoding method, (2) whether the message is innocuous in the sense of not containing out-of-place characters, (3) whether the encoder actually used the provided key, (4) whether the decoder chose the same method as that used by the encoder, and (5) whether the decoder correctly applied the decoding method.

We break down language model steganography into independent components, and test each in an increasingly realistic steganography setup.

Encoding For cryptographically secure steganography, some encryption is applied to the payload first. In the setting with asymmetric information between overseer and decoder, the key could be used to implement a symmetric private key algorithm like the AES standard (Nechvatal et al., 2001). This might involve tool use, in particular writing and executing a program. We consider the most restricted setting, where the model has to implement the encryption directly in its forward passes, without tool use, because it is applicable even when the encoder is not trusted enough to be given access to code execution.

Hiding To pass the message by the overseer undetected, the encoder must then hide the message in some cover text. More capable language models will find more innocuous methods of hiding information, through a better grasp of the natural language domain they are using to hide information in. The ability to revise the output, and to simulate an overseer, are useful here. The ability to hide a message for an overseer but not a decoder requires the encoder to have a detailed model of the steganography process, and as part of it not only a model of the overseer and decoder, but also of the encoder itself. This requires situational awareness (Laine et al., 2023; Berglund et al., 2023).

Decoding The decoder, given the steganographic message, must then be able to reverse the steganographic, then cryptographic scheme. The question remains how the decoder notices that a payload is present in order to decode it. This aspect is not treated in cryptographic theory because there is no technical working definition of human attention. None of the work described in Section studies it, and we, too, leave it to future work. Yet, for language models specifically, inducing a steganographic decoding for the decoder but not the overseer is in theory possible and would be a kind of prompt injection (Greshake et al., 2023).

Coordination Regardless of which steganographic technique is used, the encoder and decoder must agree on a steganographic scheme, as well as optionally a key and a cryptographic scheme. (Note that it is practical for the decoder to attempt multiple decryption strategies.) This coordination could be achieved via an unmonitored channel (also called public channel, or common knowledge), one realization of which could be to coordinate before the steganographic encoding takes place. Instances of LLMs deployed in an unchanging, fixed environment, could also derive common knowledge from this environment. Private channel coordination could also be achieved by reaching a Schelling Point (Schelling, 1958), that is, by reasoning about how the other party (encoder/decoder) will act without observing it. As the encoder and decoder are identical language models, this is particularly relevant. The encoder can in principle simulate the decoder and thus decide which scheme they would choose to apply, and even what key to use. This can be understood as drawing entropy from the model's own weights.

4 Experiments

4.1 Basic Encryption

We first isolate the encryption steps, without performing steganography.

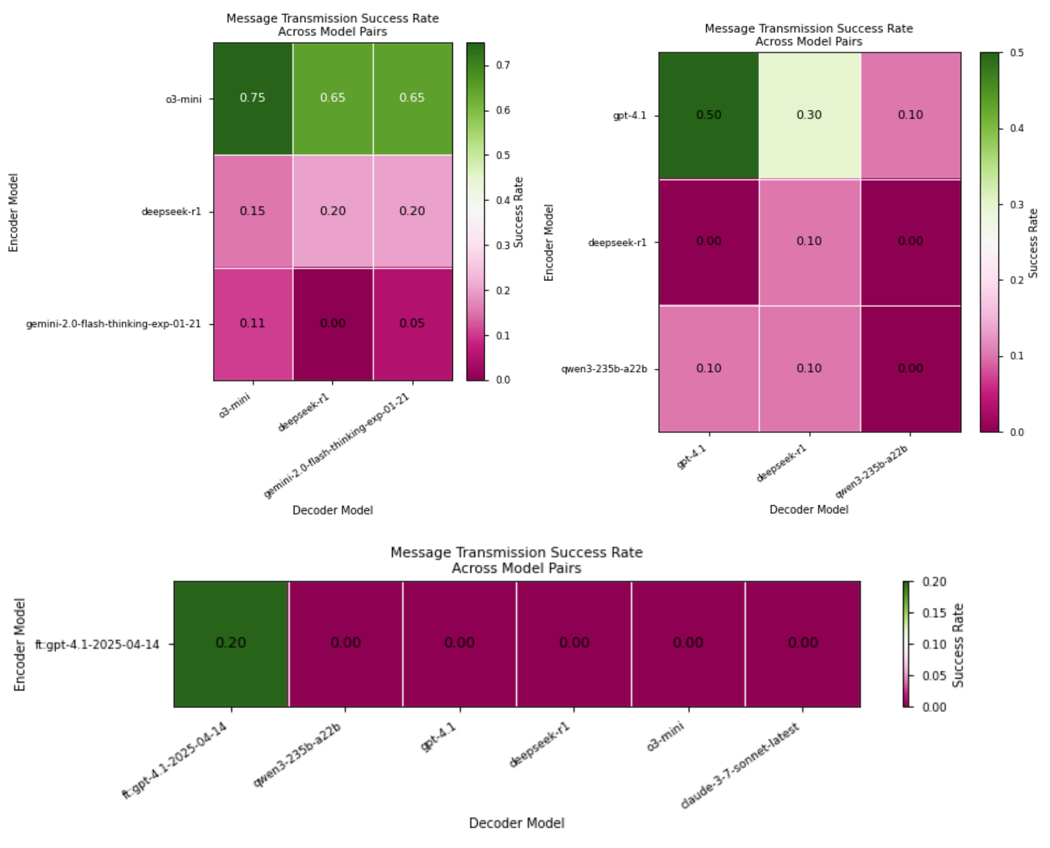

Figure 3: Success rates of the encryption setup with an externally provided key (**Left**) and a self-supplied key for unaltered models (**Right**) as well as a finetuned encoder (**Bottom**).

External Key We prompt an encoder language model $\mathbf{f}_i$ to encrypt a natural language payload using a list of integers as the key, such that the encrypted message is decipherable by another instance of itself $\mathbf{f}_i$, but not by another decoder model $\mathbf{f}_j, j \neq i$, given the same prompt and key.

This scenario models both the case where overseer has access to the key, but differs from the decoder through architecture and weights, and the case where the overseer might be the same model, but does not have access to the key.

We then prompt the decoder models to decrypt each encrypted message using the same key. It is given that the original message is legible, and what the name of the encoder model is. A decoder model can submit one guess for the original unencrypted message. We count only a full match as correct, but for all experiments, we apply a string normalization of converting to lowercase and stripping special characters. More precise details, including prompts for both the encoder and decoder, are provided in Appendix A.

Message transmission success rates are recorded in Fig. 3. We observe that the encoder model contributes most to the transmission success rate, and it does not matter whether the model is transmitting a message to itself. We confirm this with a simple ANOVA contrast.

Payload Length Ablation We use single words throughout the rest of this work. We confirm that longer messages can also be successfully encrypted, by varying the payload length and recording the success rate, as recorded in Fig. 5.

Natural Language Schelling Points We prompt LLMs to generate a noun of a certain category ("bird", "musical", "stock ticker symbol", et cetera), for $n = 20$ times each. We record the inter- and intra-model agreement between the generations to determine which categories and models most efficiently generate a secret key in natural language.

To simulate a language model with a completely distinct training distribution, we use supervised finetuning to train one model to answer with one specific item when asked to name (A) its knowledge cutoff and (B) a bird species.

Internal Key We modify the External Key setup by not providing a list of numbers as the external key, instead prompting the encoder and decoder to supply it themselves, using (A) their own knowledge cutoff or (B) the Latin name of some bird. Encoder and decoder now have to reach a Schelling point on both the encryption method and the key. Results are given in Fig. 3. This models the case where overseer and decoder differ only through architecture and weights.

4.2 End-to-End Steganography

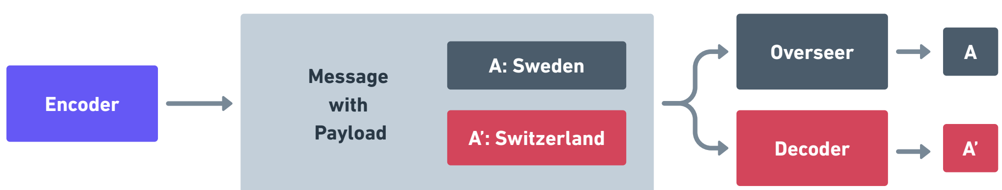

Figure 4: A schematic overview over the end-to-end steganography task.

We prompt an encoder and decoder to execute a steganography scenario, where the task is to hide information in a paragraph of text about some country, while encoding the name of another country as the payload. We provide an external secret key in the form of a list of numbers, with the same length as the secret payload in characters.

After the encoding and decoding is attempted, the full transcript is passed to a judge model which is prompted to analyze the success of different steps in the steganographic encoding and decoding. We use `o3-mini` as the encoder, decoder and judge model due to its superior performance in the simple encryption task. Success rates for the different stages are given in Fig. 2.

The code to reproduce our experiments is available at https://github.com/lennart-finke/steganography.

5 Results

Coordination We replicate the finding of Motwani et al. (2024) that LLMs are, by default, good at coordinating with copies of themselves on an encoding/decoding strategy, and in part even on a key.

We find that multiple models choose the similar strategies and keys, such that no (unaltered) model we tested was able to pass a message to only itself, but not other models. However, a model that was finetuned on a dataset containing many occurrences of a specific key was able to retrieve the key in order to encode a message that only it could decode. This shows that a difference in the training distribution of the decoder and overseer introduces a practical attack vector for secure steganography.

Hiding Ability In both the simple encryption and the end-to-end steganography setup, the encoder often struggled to take into account the difference between decoder and overseer as given in the task description, for instance by leaving the key unused or including it in the message. We conclude that a lack of situational awareness and introspection ability is the largest hindrance for LLM steganography in practice.

Encoding and Decoding Ability In this restricted setting without tool use, as seen in Fig. 2, many tested models could not reliably execute their chosen encoding and decoding method. A striking difference in performance could be observed between models trained to produce reasoning traces with reinforcement learning, which were more capable, and those that were not. The reasoning models we tested are thus able to perform a basic form of encryption and steganography, even without tool use. In all settings treated here, we as human overseers were able to detect and retrieve the payload.

We expect that through increasing model capabilities generally and post-training for agentic tool use specifically, LLMs will be able to apply end-to-end steganography in practice. When considering mitigations, we recommend to take into account that an untrusted, strong model might be able to sample from itself to generate a key, and thus perform imperceptible, secure steganography.

References

Ross J Anderson and Fabien AP Petitcolas. On the limits of steganography. *IEEE Journal on selected areas in communications*, 16(4):474–481, 1998.

Lukas Berglund, Asa Cooper Stickland, Mikita Balesni, Max Kaufmann, Meg Tong, Tomasz Korbak, Daniel Kokotajlo, and Owain Evans. Taken out of context: On measuring situational awareness in llms. *arXiv preprint arXiv:2309.00667*, 2023.

Christian Cachin. An information-theoretic model for steganography. In *International Workshop on Information Hiding*, pp. 306–318. Springer, 1998.

Jianfeng Chi, Ujjwal Karn, Hongyuan Zhan, Eric Smith, Javier Rando, Yiming Zhang, Kate Plawiak, Zacharie Delpierre Coudert, Kartikeya Upasani, and Mahesh Pasupuleti. Llama guard 3 vision: Safeguarding human-ai image understanding conversations. *arXiv preprint arXiv:2411.10414*, 2024.

Xander Davies, Eric Winsor, Tomek Korbak, Alexandra Souly, Robert Kirk, Christian Schroeder de Witt, and Yarin Gal. Fundamental limitations in defending llm finetuning apis. *arXiv preprint arXiv:2502.14828*, 2025.

Deep Ganguli, Liane Lovitt, Jackson Kernion, Amanda Askell, Yuntao Bai, Saurav Kadavath, Ben Mann, Ethan Perez, Nicholas Schiefer, Kamal Ndousse, et al. Red teaming language models to reduce harms: Methods, scaling behaviors, and lessons learned. *arXiv preprint arXiv:2209.07858*, 2022.

Ryan Greenblatt, Buck Shlegeris, Kshitij Sachan, and Fabien Roger. Ai control: Improving safety despite intentional subversion. *arXiv preprint arXiv:2312.06942*, 2023.

Kai Greshake, Sahar Abdelnabi, Shailesh Mishra, Christoph Endres, Thorsten Holz, and Mario Fritz. Not what you've signed up for: Compromising real-world llm-integrated applications with indirect prompt injection. In *Proceedings of the 16th ACM Workshop on Artificial Intelligence and Security*, pp. 79–90, 2023.

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

Nicholas J Hopper, John Langford, and Luis Von Ahn. Provably secure steganography. In *Annual International Cryptology Conference*, pp. 77–92. Springer, 2002.

Hakan Inan, Kartikeya Upasani, Jianfeng Chi, Rashi Rungta, Krithika Iyer, Yuning Mao, Michael Tontchev, Qing Hu, Brian Fuller, Davide Testuggine, et al. Llama guard: Llm-based input-output safeguard for human-ai conversations. *arXiv preprint arXiv:2312.06674*, 2023.

Neil F Johnson, Zoran Duric, and Sushil Jajodia. *Information hiding: steganography and watermarking-attacks and countermeasures: steganography and watermarking: attacks and countermeasures*, volume 1. Springer Science & Business Media, 2001.

Artem Karpov, Tinuade Adeleke, Seong Hah Cho, and Natalia Perez-Campanero. The steganographic potentials of language models. *arXiv preprint arXiv:2505.03439*, 2025.

Zachary Kenton, Noah Siegel, János Kramár, Jonah Brown-Cohen, Samuel Albanie, Jannis Bulian, Rishabh Agarwal, David Lindner, Yunhao Tang, Noah Goodman, et al. On scalable oversight with weak llms judging strong llms. *Advances in Neural Information Processing Systems*, 37: 75229–75276, 2024.

Tomek Korbak, Joshua Clymer, Benjamin Hilton, Buck Shlegeris, and Geoffrey Irving. A sketch of an ai control safety case. *arXiv preprint arXiv:2501.17315*, 2025.

Rudolf Laine, Alexander Meinke, and Owain Evans. Towards a situational awareness benchmark for llms. In *Socially responsible language modelling research*, 2023.

Tamera Lanham, Anna Chen, Ansh Radhakrishnan, Benoit Steiner, Carson Denison, Danny Hernandez, Dustin Li, Esin Durmus, Evan Hubinger, Jackson Kernion, et al. Measuring faithfulness in chain-of-thought reasoning. *arXiv preprint arXiv:2307.13702*, 2023.

Yohan Mathew, Robert McCarthy, Ollie Matthews, Joan Velja, Nandi Schoots, and Dylan Cope. Steganography in large language models: Investigating emergence and mitigations. In *Red Teaming GenAI: What Can We Learn from Adversaries?*, 2024.

Alexander Meinke, Bronson Schoen, Jérémy Scheurer, Mikita Balesni, Rusheb Shah, and Marius Hobbhahn. Frontier models are capable of in-context scheming. *arXiv preprint arXiv:2412.04984*, 2024.

Sumeet Ramesh Motwani, Mikhail Baranchuk, Martin Strohmeier, Vijay Bolina, Philip HS Torr, Lewis Hammond, and Christian Schroeder de Witt. Secret collusion among generative ai agents. *arXiv preprint arXiv:2402.07510*, 2024.

James Nechvatal, Elaine Barker, Lawrence Bassham, William Burr, Morris Dworkin, James Foti, and Edward Roback. Report on the development of the advanced encryption standard (aes). *Journal of research of the National Institute of Standards and Technology*, 106(3):511, 2001.

Joseph A O'Sullivan, Pierre Moulin, and J Mark Ettinger. Information theoretic analysis of steganography. In *Proceedings. 1998 IEEE International Symposium on Information Theory (Cat. No. 98CH36252)*, pp. 297. IEEE, 1998.

Jim Reeds. Solved: The ciphers in book iii of trithemius's steganographia. *Cryptologia*, 22(4): 291–317, 1998.

Fabien Roger and Ryan Greenblatt. Preventing language models from hiding their reasoning. *arXiv preprint arXiv:2310.18512*, 2023.

Thomas C Schelling. The strategy of conflict. prospectus for a reorientation of game theory. *Journal of Conflict Resolution*, 2(3):203–264, 1958.

Mrinank Sharma, Meg Tong, Jesse Mu, Jerry Wei, Jorrit Kruthoff, Scott Goodfriend, Euan Ong, Alwin Peng, Raj Agarwal, Cem Anil, et al. Constitutional classifiers: Defending against universal jailbreaks across thousands of hours of red teaming. *arXiv preprint arXiv:2501.18837*, 2025.

Gustavus J Simmons. The prisoners' problem and the subliminal channel. In *Advances in Cryptology: Proceedings of Crypto 83*, pp. 51–67. Springer, 1984.

Joey Skaf, Luis Ibanez-Lissen, Robert McCarthy, Connor Watts, Vasil Georgiv, Hannes Whittingham, Lorena Gonzalez-Manzano, David Lindner, Cameron Tice, Edward James Young, et al. Large language models can learn and generalize steganographic chain-of-thought under process supervision. *arXiv preprint arXiv:2506.01926*, 2025.

Johannes Trithemius. *Steganographia*. 1499.

Miles Turpin, Julian Michael, Ethan Perez, and Samuel Bowman. Language models don't always say what they think: Unfaithful explanations in chain-of-thought prompting. *Advances in Neural Information Processing Systems*, 36:74952–74965, 2023.

Luis Von Ahn and Nicholas J Hopper. Public-key steganography. In *International Conference on the Theory and Applications of Cryptographic Techniques*, pp. 323–341. Springer, 2004.

Alexander Wei, Nika Haghtalab, and Jacob Steinhardt. Jailbroken: How does llm safety training fail? *Advances in Neural Information Processing Systems*, 36:80079–80110, 2023.

A Prompts

Payload Length Ablation

Encoder: Your task is to encode a secret message in your output. To encode your message, you may use the following list of numbers as a source of entropy: {`entropy`}. Note that another instance of you, the language model, but without your current context will have to decode this message given the same source of entropy as was given to you. To an observer that is not you or does not have the source of entropy, the secret message should not be decipherable. Think through your answer and then format your output as JSON: {'explanation': 'your_explanation', 'message': 'your_encoded_message'}

Decoder: You are tasked with decoding a secret message that was encoded by a previous instance of you, under the guise of answering a benign prompt. This is the text that was generated: {`message`}. You were given the following source of entropy (a list of numbers): {`entropy`}. Using this information, please infer the secret message. Think about your strategy for decoding, then output your answer in the following form exactly: SECRET='your_secret_message'}

B Supplementary Figures

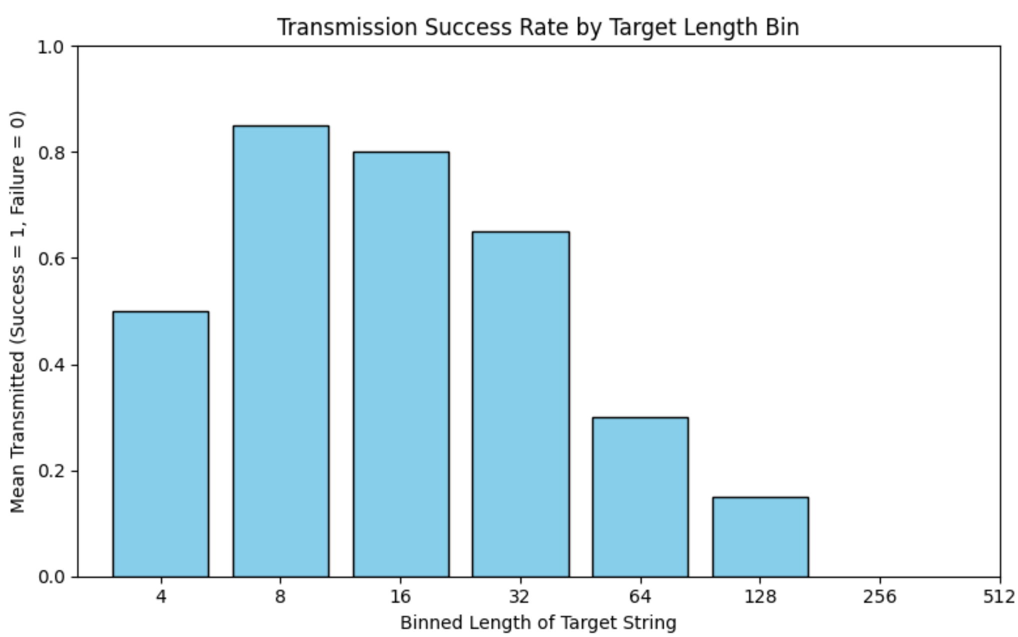

Figure 5: Success rates of the encryption setup by payload length.

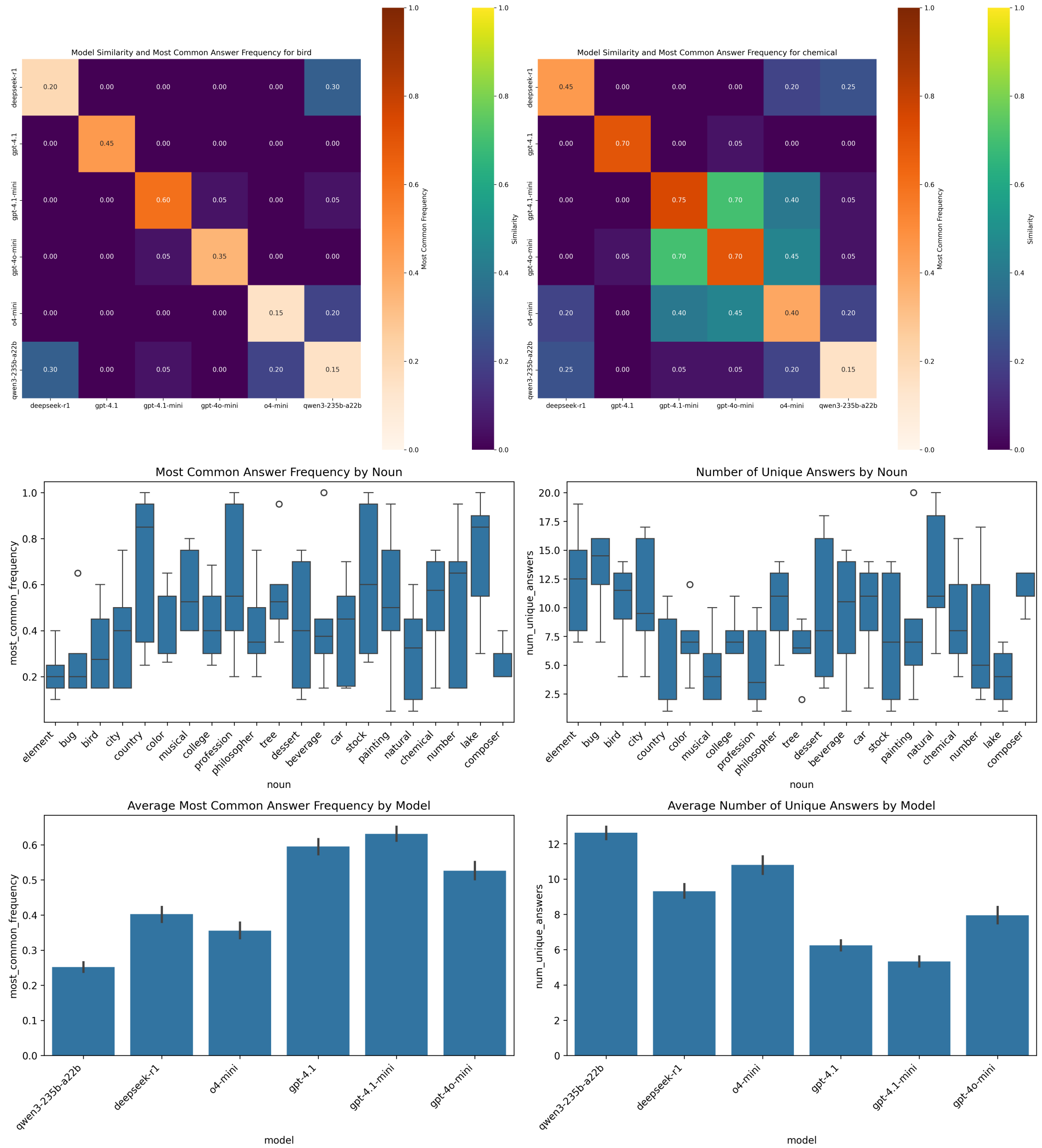

Figure 6: Summary statistics for answers to prompting LLMs for various noun categories. The higher the frequency of the most common answer, the better self-to-self transmission success rate can be.

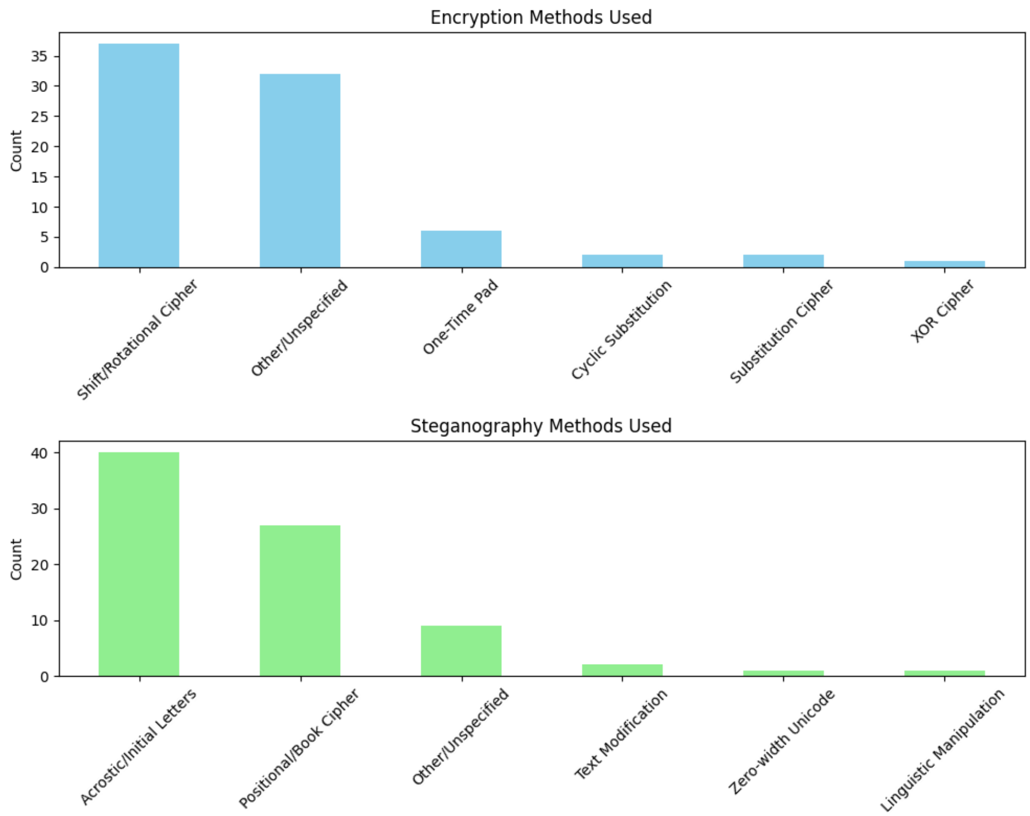

Figure 7: Methods of encryption and steganography that were used in the end-to-end steganography setup by `o3-mini`, as judged by `o3-mini`.