TU Delft

Delft University of Technology

Document Version
Final published version

Licence
Dutch Copyright Act (Article 25fa)

Citation (APA)
Gaure, S., Koffas, S., Picek, S., & Rønjom, S. (2025). $L^2 \cdot M = C^2$ *Large Language Models Are Covert Channels*. Paper presented at ICASSP, Hyderabad, India. https://doi.org/10.1109/ICASSP49660.2025.10887756

Important note
To cite this publication, please use the final published version (if applicable).
Please check the document version above.

Copyright
In case the licence states “Dutch Copyright Act (Article 25fa)”, this publication was made available Green Open Access via the TU Delft Institutional Repository pursuant to Dutch Copyright Act (Article 25fa, the Taverne amendment). This provision does not affect copyright ownership.
Unless copyright is transferred by contract or statute, it remains with the copyright holder.

Sharing and reuse
Other than for strictly personal use, it is not permitted to download, forward or distribute the text or part of it, without the consent of the author(s) and/or copyright holder(s), unless the work is under an open content license such as Creative Commons.

Takedown policy
Please contact us and provide details if you believe this document breaches copyrights.
We will remove access to the work immediately and investigate your claim.

This work is downloaded from Delft University of Technology.

ICASSP 2025 - 2025 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) | 979-8-3503-6874-1/25/$31.00 ©2025 IEEE | DOI: 10.1109/ICASSP49660.2025.10887756

$\mathbf{L^2 \cdot M = C^2}$
Large Language Models Are Covert Channels

Simen Gaure[1], *Stefanos Koffas*[2], *Stjepan Picek*[3,2], *Sondre Rønjom*[1,4]

[1]Norwegian National Security Authority, Norway
[2]Cybersecurity Group, Delft University of Technology, The Netherlands
[3]Digital Security Group, Radboud University, The Netherlands
[4]University of Bergen, Norway

ABSTRACT

Large Language Models (LLMs) are susceptible to various attacks but can also improve the security of diverse systems. However, how well do open source LLMs behave as covertext distributions to, e.g., facilitate censorship-resistant communication? In this paper, we explore open-source LLM-based covert channels. We empirically measure the security vs. capacity of two open-source LLM models (Llama-7B and GPT-2) to assess their performance as covert channels. Although our results indicate that such channels are not likely to achieve high practical bitrates, we also show that the chance for an adversary to detect covert communication is low. To ensure our results can be used with the least effort as a general reference, we employ a conceptually simple and concise scheme and only assume public models.

1. INTRODUCTION

Model inference depends, among others, on temperature and internal randomness to create "liveliness". By exchanging internal randomness with binary ciphertexts (assuming a suitable encoding scheme), inference can be considered as a transformation of ciphertexts into specific domain languages (e.g., HTML) of the particular model, resembling format transforming encryption (FTE) [7]. C. Cachin formalizes a model for information-theoretic steganography for covertext distributions [2]. While there have been several attempts to embed covert messages into LLM responses [18, 6, 8], most of them altered the model distribution directly, thus altering the model itself. As such, we use concepts commonly explored in cryptography. We note that the idea of connecting machine learning (ML) and cryptography spans decades [16, 14], and, more recently, cryptographic concepts were used to provide stronger theoretical foundations for ML security [10, 1].

Inspired by Cachin's information-theoretic model for steganography [2], we provide a practical and heuristic analysis of partition-based covert channels using covertext distributions defined by LLMs. We extend Cachin's partition-based channel to a setting where the rate is continuously adjusted according to the token distribution of our LLM. Cachin's proposal is conceptually simple and theoretically sound, and its extension to LLM covertext distributions is straightforward. We believe that basing our experiments on the simplicity of Cachin's partition-based approach broadens the impact of our results Here, ciphertext bits are encoded into token distributions using a partitioning algorithm that adjusts the rate according to varying token distributions. In contrast to recent papers, our objective is to conduct practical experiments to shed light on the relationship between indistinguishability and channel rate for practical LLM covertext distributions.

2. PRELIMINARIES

Llama 2. We use Llama 2 with 7B parameters as it is open-source and "small" enough to be run by common users. The text in Llama consists of "tokens", words or parts of words. Internally, Llama operates on these tokens, which are represented by integers. In our case, there are $K = 32000$ different tokens. The text generation proceeds by feeding Llama the prompt. The model then produces a set of K real numbers, $(\ell_i)_{i=1}^K$. These are then scaled by a user-supplied "temperature" T, and converted to a discrete probability distribution, $q_i = \frac{\exp(\ell_i/T)}{\sum_j^K \exp(\ell_j/T)}$. The probabilities are sorted in descending order. Then, a user-supplied cutoff C is applied, by letting $c = 1 + \max\{n : \sum_i^n q_i < C\}$, and, $\tilde{p}_i = \begin{cases} q_i & \text{if } i \le c, \\ 0 & \text{if } i > c. \end{cases}$

The $\tilde{p}_i$'s are then scaled to have sum 1, $p_i = \tilde{p}_i / \sum_j \tilde{p}_j$ yielding a probability distribution (ϕ_i) over the tokens. A token w_t is then sampled from the distribution ϕ_t. This token, an integer, represents a word or part of a word, which is output from Llama as a text. The token is also fed back into Llama to change its state, yielding a new probability distribution ϕ_{t+1}. Then another token w_{t+1} is sampled from ϕ_{t+1}. This continues until an "end of sequence" (EOS) token is sampled, or a maximum length is reached.

GPT-2. Similarly, GPT-2 generates a stream of text. For this reason, we have also run experiments with an open-source

Authorized licensed use limited to: TU Delft Library. Downloaded on April 17,2026 at 07:22:45 UTC from IEEE Xplore. Restrictions apply.

version of GPT-2[1] to evaluate our approach's generalization.

Chosen Hyperparameters. The information an even with probability p is $-\log_2 p$. The entropy of a probability distribution $(p_i)_{i=1}^k$ is the expected information $-\sum_{i=1}^k p_i \log_2 p_i$. Entropy can describe the capacity of a communication channel. We will use LLM-generated text as a communication channel. The entropy of the token distributions is then the maximal bit rate we can achieve by encoding bits into token choices. In Llama, we can control the entropy of the token distributions with the "temperature" T and "top-p" C hyperparameters. Temperatures above ≈ 1.3 may result in logorrhea, where the model produces an unintelligible stream of tokens. Our experiments show that we can increase T above 1 without problems. We stick to $T = 1.1$, truncating at the $C = 0.95$ quantile. Our prompt consists of a "system" ("Answer like a play by Shakespeare") and a "user" part ("Write a story about a humorous encounter with The Hollow Men"). We are not interested in the actual tokens, only the distribution of probabilities, irrespective of which tokens they refer to. To collect some distributions, using our prompt and a random seed set from the system clock, we let our models produce 2200 responses., resulting in $\approx$ 1000000 distributions of average entropy ≈ 1.15 bits. The token distributions were saved and used for our analysis.

We have removed the distributions with more than 1024 non-zero probabilities ($\approx 0.6\%$). Such distributions have on average ≈ 5500 non-zero probabilities and ≈ 7.5 average entropy. We assume that a token distribution with many non-zero probabilities does not reflect any reasonable text generation but is possibly an artifact of incomplete model training and that the probabilities have not quite reached numerically zero.

3. ENCODING BITS BY SAMPLING TOKENS

3.1. Setup

Rather than using a (pseudo) random generator to choose tokens from the distributions, we will use the message bits. The message bits can be decoded from the generated text by a receiver running the LLM with the same hyperparameters and prompt. We use the generated text as a communication channel in a way that is hard to distinguish from using a random generator so that the communication is covert.

We assume the message bits are (pseudo) random, e.g., a ciphertext. A simple way to ensure this is to XOR each bit with a bit from a random number generator, e.g., AES in counter mode [5]. We assume an adversary knows our model and its parameters but not whether we encode some unknown bits in the tokens or if we sample tokens randomly. Then, the question is, could an adversary reject the null hypothesis that the tokens have been randomly drawn?

We partition the probabilities into two sets with almost equal sums. To encode a message bit, we use its value to choose between the two sets, and then we draw randomly from the chosen set. The receiver of the text, running the same model, can then recover the message bit by determining which of the two sets the token is drawn from.

Next, we formally describe our scheme. The tokens $T = \{1, 2, \ldots, K\}$ have probabilities $P = (p_i)_{i=1}^K$. We start out by finding sets T_0 and T_1 with $T_0 \cap T_1 = \emptyset$, and $T_0 \cup T_1 = T$, with corresponding probabilites P_0 and P_1 such that $\sum P_0 \approx \sum P_1$. That is, the tokens have been partitioned into two groups with approximately equal probability sums. If this cannot be done because no subset of P sums to something close to $\frac{1}{2}\sum P$, we cannot encode a bit in this token choice. If, however, we have found suitable T_0 and T_1, we pick T_0 to encode a 0-bit, and T_1 to encode a 1-bit. To measure how good the partitioning is, we use the *split entropy*, $h_s = -(p \log_2 p + (1-p)\log_2(1-p)$, where $p = \sum P_0$. As a hyperparameter for the encoding, we choose a *minimum split entropy* $H_s \leq 1$ and disallow partitions with $h_s < H_s$.

We continue this process with the chosen half of the tokens. That is, if we have chosen T_0, we try to split it into two halves T_{00} and T_{01} with approximately equal probability sums P_{00} and P_{01}. If successful, we encode another bit. We continue this process until we are left with a set $T_\perp$, which we cannot halve into two almost equal parts. We then draw a token randomly from $T_\perp$ according to the probabilities in $P_\perp$. The pseudocode is given in Algorithm 1 and is assumed to replace the random sampling of tokens in the LLM.

Algorithm 1 Use message bits to sample a token

```
function ENCODEBITS(bits, Q, H_s) → (number of bits encoded, token)
    ▷ Q is the vector of token probabilities, bits is a vector of message bits ◁
    P ← Q
    i ← 0
    while PARTITION(P, H_s) → P_0, P_1 do
        i ← i + 1
        if bits[i] = 0 then
            P ← P_0
        else
            P ← P_1
    ▷ The partitions are assumed to be indexed by indices of Q ◁
    idx ← <Sample index from P>
    return i, idx
```

As shown in Algorithm 2, the decoder uses the reverse process and splits T into two halves T_0 and T_1, similarly to the encoder. The decoder has received the generated text, knows the chosen token w, and decodes a 0-bit if $w \in T_0$, and a 1-bit if $w \in T_1$. It then tries to split again and may decode another bit. If unable to split, it goes on to the next token.

The split entropy has a relation to the security of the encoding system via [2, Theorem 2], stating that if the difference $|\sum P_1 - \sum P_0|$ is δ, the security of the scheme is $\frac{\delta^2}{\log_e 2}$. With a partition P_0, P_1 with $\sum P_0 = \frac{1}{2}(1-\delta)$ and $\sum P_1 = \frac{1}{2}(1+\delta)$, the split entropy is a function, $h(\delta) = -(\frac{1}{2}(1-\delta)\log_2(\frac{1}{2}(1-\delta)) + \frac{1}{2}(1+\delta)\log_2(\frac{1}{2}(1+\delta)))$. It is easily shown by applying L'Hôpital's rule twice, that $\lim_{\delta\to 0}(1-h(\delta))/\delta^2 = \frac{1}{2\log_e 2}$. For split entropies h_s close to 1, we therefore have $\frac{\delta^2}{\log_e 2} \approx$

[1] https://huggingface.co/openai-community/gpt2-xl

Authorized licensed use limited to: TU Delft Library. Downloaded on April 17,2026 at 07:22:45 UTC from IEEE Xplore. Restrictions apply.

Algorithm 2 Decode message bits from a token

```
function DECODEBITS(idx, Q, H_s) → decoded message bits
    ▷ idx is an index into the token probabilities Q ◁
    P ← Q
    bits ← ∅
    while PARTITION(P, H_s) → P_0, P_1 do
        if idx ∈ indices(P_0) then
            <append 0 to bits>
            P ← P_0
        else
            <append 1 to bits>
            P ← P_1
    return bits
```

$2(1-h_s)$. Thus, the security of our encoding in terms of [2] is $\approx 2(1-H_s)$.

3.2. Partitioning the P

We have used a simple, almost greedy algorithm to split $P \subset \mathbb{R}^+$ with $\sum P = 2\delta$. We choose a $k \in \mathbb{N}$ so that 2^k is not too large, e.g., $k = 10$. We sort $P = (p_i)_i$ in descending order. Then, we find an n such that $L = \sum_{i=1}^{n} p_i \approx \delta - \epsilon$ and $\sum_{i=1}^{n+k} p_i \approx \delta + \epsilon$, for some $\epsilon \geq 0$. For every subset S of $\{p_i\}_{i=n+1}^{n+k}$, we compute $s = L + \sum S$, and choose the S that minimizes $|s - \delta|$. We let $P_0 = \{p_i\}_{i=1}^{n} \cup S$, and $P_1 = P \setminus P_0$. With $p = \frac{1}{2\delta}\sum P_0$, we can compute the split's entropy, $h_s = -(p \log_2 p + (1-p)\log_2(1-p))$. We deem the split acceptable if $h_s \geq H_s$.

If we aim for encoding no more than 1 bit in each token, we should use a large k. Since we are partitioning the result again (P_0 or P_1), this is not necessarily optimal. A near-perfect split may make further splits harder. Our experiments show that lowering k to 3 improves the encoding's performance because then the chosen partition tends to be easier to partition again, leading to more encoded bits in a single token choice. However, the closer to 1 we choose H_s, the larger k should be. Based on preliminary experiments, we use $k = \lfloor -\log(1-H_s) - 0.5 \rfloor$, and clamp k between 2 and 16.

Allowing more inaccurate splits will increase the possible encoding bit rate by splitting more probability sets, but it will also increase the possibility of adversaries being able to discover that something is wrong with the generated text. Thus, we need to account for this trade-off.

4. DETECTING ANOMALOUS TOKEN SAMPLES

As above, we denote by ϕ_n the discrete distribution from which the nth token is sampled. We denote its K probabilities by $(p_k^n)_{k=1}^K$. If we manipulate the token sampling, we must ensure that it still looks like random sampling from the token distributions ϕ_n. A manipulation could, e.g., lead to a bias in the probabilities of the tokens that are selected or too many too small and too large probabilities. Such bias could make our covert channel less stealthy and more easily detected. Here, we describe a simple test that can be used to test if a sequence of tokens has the expected information content.

When we draw a token, it has a specific probability p_k^n, and we denote its information, a random variable, by $h_n = -\log p_k^n$. The entropy of ϕ_n is denoted by $H_n = \mathbb{E}(h_n) = -\sum_k p_k^n \log p_k^n$, the expectation of the random variable h_n. We denote by $D_n = H_n - h_n$ a random variable that measures the deviation from the expected information. When N tokens have been sampled, we form the random variable $D = \frac{1}{N}\sum_n D_n$. We have $\mu = \mathbb{E}(D) = 0$ and variance:

$$\begin{aligned}\sigma^2 = \operatorname{Var} D &= \operatorname{Var}\left(\frac{1}{N}\sum_n H_n - h_n\right)\\ &= \frac{1}{N^2}\sum_n \operatorname{Var} h_n\\ &= \frac{1}{N^2}\sum_n \mathbb{E}(h_n^2) - \mathbb{E}(h_n)^2.\end{aligned}$$

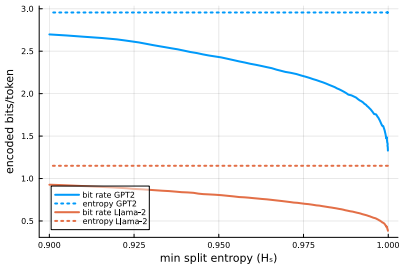
An actual realization of tokens then produces a sequence of tokens with information $\hat{h}_n$, and then a $\hat{D} = \frac{1}{N}\sum_n (H_n - \hat{h}_n)$, drawn from D. Now, D is an average of random variables D_n, and it is no surprise that it is close to being normally distributed. We can then test how close $\hat{D}$ is to $\mathbb{E}(D) = 0$ by computing a z-score, $z = |\hat{D}|/\sigma$, and compute a p-value as $p = 2(1 - \Phi(z))$, where Φ is the standard normal cdf. The p-value is the probability that we draw a d from D with $|d| \geq |\hat{D}|$. A p-value close to zero is unlikely if tokens are drawn properly from their distributions.

5. EXPERIMENTAL RESULTS

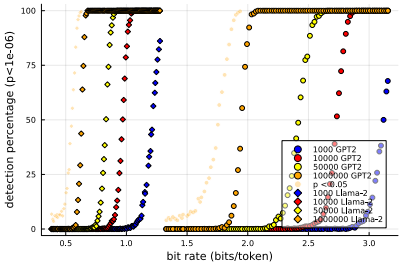
In our experiments, we encode a stream of random bits as presented in Section 3 and try to detect the presence of the resulting skewed sampling as shown in Section 4. In Figure 1a, the minimum split entropy H_s varies from 0.9 to 0.9999. We have plotted the resulting bit rates from encoding random bits in our token distributions. The dashed lines are the average entropy of the token distributions for each model. As H_s approaches 1, we see a dramatic drop in bit rate. There are fewer token distributions that can possibly be partitioned with such precision, and therefore, fewer tokens that can be used for encoding bits, resulting in a lower bit rate.

In Figure 1b, we have done 1000 runs of encoding random bitstreams for each H_s, with varying numbers of token distributions drawn from our Llama and GPT-2 collections. We have plotted the fraction with the p-value (from Section 4) below 10^{-6}. While this could be considered a somewhat arbitrary significance level, it clearly shows how the bit rate and the number of tokens influence the anomaly detection rate. For 10^6 tokens, we have added significance levels 0.001 and 0.05 as well. We see that for 1000 tokens, we can achieve a bit rate of ≈ 0.9 and ≈ 2.8 bits/token without a large detection probability for Llama-7B and GPT-2, respectively. Llama is a model with low entropy [20, 11] resulting in a channel of lower bandwidth. It is known [9] that distinguishing two finite

Authorized licensed use limited to: TU Delft Library. Downloaded on April 17,2026 at 07:22:45 UTC from IEEE Xplore. Restrictions apply.

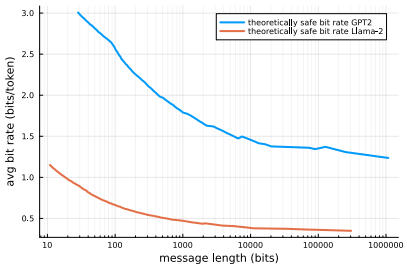


(a) Entropy and bit rate

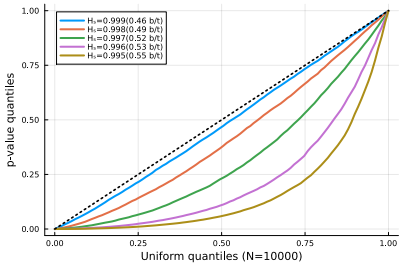


(b) Detection rate

Fig. 1: Bit rate and detection rate



(a) Theoretically safe bit rates



(b) QQ-plots for different bit rates on Llama-2

Fig. 2: Theoretical bounds

probability distributions with statistical distance ϵ can be done with $O(\epsilon^{-2})$ samples. So given a limit N to the number of tokens, we should have $\epsilon < N^{-1/2}$, for simplicity removing the O. Our partitioning of the probability distributions changes the probabilities a slight amount, and one partition has sum $p = 0.5 - \delta$ whereas the other has sum $1 - p$. This means that the actual probabilities are slightly changed, yielding a statistical distance at most δ. This translates to a minimum split entropy H_s and, thus, a theoretically safe bit rate. In Figure 2a, we have plotted these for our set of token distributions for both models. They show a lower bound for the safe bitrates, though since we have removed a constant factor by replacing $O(\epsilon^{-2})$ by ϵ^{-2}, the graph may be horizontally shifted, and should not be taken to be exact.

In Figure 2b, we used our collection of $\approx 10^6$ distributions for Llama-2 to encode random bits, made 10^4 p-values for $H_s \in 0.995..0.999$, and made QQ-plots against a uniform distribution. We see that the p-values do not become reasonably uniform until the bit rate drops below 0.5 bits/token. But even with a bit rate of 0.46, our p-values are not entirely uniform.

6. CONCLUSION

It becomes more challenging to recognize if some content is produced by LLMs or humans [3, 12] resulting in research efforts fighting disinformation, such as model watermarking [4, 13, 17, 19, 21]. However, the fundamental problem of identifying data produced by a model will likely remain a major challenge in many areas [12, 15]. Then, if it is computationally infeasible to distinguish synthetic from "organic" data, it is computationally infeasible to detect encrypted covert channels based on the same covertext distributions. We showed that LLMs can be used for covert communication with around 1 bit or 3 bits per word on average bit rate for Llama-7B and GPT-2, respectively. We conducted extensive experiments on the relationship between covert channel bandwidth and distinguishing probability based on a conceptually simple scheme adopted from the earlier fundamental work on secure covert channels in [2]. Using a simple and clean scheme that is applicable to different models allows us to conduct practical experiments that hopefully can be reused by others for both theoretical analysis and comparison with minimal effort. In practice, the detection probability analyzed in this paper is very conservative. It assumes that an adversary knows exactly which LLM is used and its parameters. If this is actually known by a real-world adversary, it is a reasonable assumption that they already know that hidden messages are encoded. If this strong assumption is relaxed, the hidden communication can be virtually impossible to detect. Our covert channel could be destroyed if the generated text is heavily altered through a rephrasing countermeasure [20]. Still, it has been shown that changing only a small fraction of tokens cannot always affect such a channel [20], making rephrasing an interesting research direction for the future.

7. ACKNOWLEDGEMENTS

This research was funded in part by The Research Council of Norway (project number 353785).

Authorized licensed use limited to: TU Delft Library. Downloaded on April 17,2026 at 07:22:45 UTC from IEEE Xplore. Restrictions apply.

8. REFERENCES

[1] Kasra Abbaszadeh, Christodoulos Pappas, Dimitrios Papadopoulos, and Jonathan Katz. Zero-knowledge proofs of training for deep neural networks. Cryptology ePrint Archive, Paper 2024/162, 2024.

[2] Christian Cachin. An information-theoretic model for steganography. *Information and Computation*, 2004.

[3] Souradip Chakraborty, Amrit Singh Bedi, Sicheng Zhu, Bang An, Dinesh Manocha, and Furong Huang. On the possibilities of ai-generated text detection, 2023.

[4] Miranda Christ, Sam Gunn, and Or Zamir. Undetectable watermarks for language models, 2023.

[5] Joan Daemen and Vincent Rijmen. *The Design of Rijndael*. Springer Berlin Heidelberg, 2002.

[6] Falcon Dai and Zheng Cai. Towards near-imperceptible steganographic text. In Anna Korhonen, David Traum, and Lluís Màrquez, editors, *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4303–4308, Florence, Italy, July 2019. Association for Computational Linguistics.

[7] Kevin P. Dyer, Scott E. Coull, Thomas Ristenpart, and Thomas Shrimpton. Protocol misidentification made easy with format-transforming encryption. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, CCS '13, page 61–72, New York, NY, USA, 2013. Association for Computing Machinery.

[8] Tina Fang, Martin Jaggi, and Katerina Argyraki. Generating steganographic text with LSTMs. In Allyson Ettinger, Spandana Gella, Matthieu Labeau, Cecilia Ovesdotter Alm, Marine Carpuat, and Mark Dredze, editors, *Proceedings of ACL 2017, Student Research Workshop*, pages 100–106, Vancouver, Canada, July 2017. Association for Computational Linguistics.

[9] Oded Goldreich. *Introduction to property testing*. Cambridge University Press, 2017.

[10] Shafi Goldwasser, Michael P. Kim, Vinod Vaikuntanathan, and Or Zamir. Planting undetectable backdoors in machine learning models : [extended abstract]. In *2022 IEEE 63rd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 931–942, 2022.

[11] Neel Jain, Khalid Saifullah, Yuxin Wen, John Kirchenbauer, Manli Shu, Aniruddha Saha, Micah Goldblum, Jonas Geiping, and Tom Goldstein. Bring your own data! self-supervised evaluation for large language models. *arXiv preprint arXiv:2306.13651*, 2023.

[12] Ganesh Jawahar, Muhammad Abdul-Mageed, and Laks V. S. Lakshmanan. Automatic detection of machine generated text: A critical survey, 2020.

[13] Travis Munyer, Abdullah Tanvir, Arjon Das, and Xin Zhong. Deeptextmark: A deep learning-driven text watermarking approach for identifying large language model generated text, 2024.

[14] Ronald L. Rivest. Cryptography and machine learning. In *Proceedings of the International Conference on the Theory and Applications of Cryptology: Advances in Cryptology*, ASIACRYPT '91, page 427–439, Berlin, Heidelberg, 1991. Springer-Verlag.

[15] Vinu Sankar Sadasivan, Aounon Kumar, Sriram Balasubramanian, Wenxiao Wang, and Soheil Feizi. Can ai-generated text be reliably detected?, 2024.

[16] L. G. Valiant. A theory of the learnable. *Commun. ACM*, 27(11):1134–1142, nov 1984.

[17] Lean Wang, Wenkai Yang, Deli Chen, Hao Zhou, Yankai Lin, Fandong Meng, Jie Zhou, and Xu Sun. Towards codable watermarking for injecting multi-bits information to llms, 2024.

[18] Zhong-Liang Yang, Xiao-Qing Guo, Zi-Ming Chen, Yong-Feng Huang, and Yu-Jin Zhang. Rnn-stega: Linguistic steganography based on recurrent neural networks. *IEEE Transactions on Information Forensics and Security*, 14(5):1280–1295, 2019.

[19] KiYoon Yoo, Wonhyuk Ahn, and Nojun Kwak. Advancing beyond identification: Multi-bit watermark for large language models, 2024.

[20] Or Zamir. Excuse me, sir? your language model is leaking (information), 2024.

[21] Hanlin Zhang, Benjamin L. Edelman, Danilo Francati, Daniele Venturi, Giuseppe Ateniese, and Boaz Barak. Watermarks in the sand: Impossibility of strong watermarking for generative models, 2023.

Authorized licensed use limited to: TU Delft Library. Downloaded on April 17,2026 at 07:22:45 UTC from IEEE Xplore. Restrictions apply.