# Efficient Provably Secure Linguistic Steganography via Range Coding

**Ruiyi Yan and Yugo Murawaki**
Graduate School of Informatics, Kyoto University
ruiyi@nlp.ist.i.kyoto-u.ac.jp, murawaki@i.kyoto-u.ac.jp

## Abstract

Linguistic steganography involves embedding secret messages within seemingly innocuous texts to enable covert communication. Provable security, which is a long-standing goal and key motivation, has been extended to language-model-based steganography. Previous provably secure approaches have achieved perfect imperceptibility, measured by zero Kullback-Leibler (KL) divergence, but at the expense of embedding capacity. In this paper, we attempt to directly use a classic entropy coding method (**range coding**) to achieve secure steganography, and then propose an efficient and provably secure linguistic steganographic method with a rotation mechanism. Experiments across various language models show that our method achieves around 100% entropy utilization (embedding efficiency) for embedding capacity, outperforming the existing baseline methods. Moreover, it achieves high embedding speeds (up to 1554.66 bits/s on GPT-2). The code is available at github.com/ryehr/RRC_steganography.

## 1 Introduction

Linguistic steganography, as a promising field in safeguarding information, refers to the art of concealing messages within texts. With rapid advancements in large language models (LLM) (Brown et al., 2020; Achiam et al., 2023; Anthropic, 2024), LM-based steganography methods (Ziegler et al., 2019; Wu et al., 2024; Yan et al., 2025) have dominated linguistic steganography, as leveraging LMs can create flexible text content, diverse genres, and consistent contexts, enabling high embedding capacity. Figure 1 illustrates how a sender (Alice) and a receiver (Bob) communicate using linguistic steganography.

Simmons's "Prisoners' Problem" (Simmons, 1984) illustrates a steganographic scenario, where Alice and Bob (steganographers) are trying to hatch an escape plan, and they can only communicate under the scrutiny of a warden, Eve (a steganalyzer), who can block communication if any suspicious content is detected. To evade detection, they embed their secret message into an innocent-looking carrier (steganographic content). Intuitively, steganographic content is expected to closely resemble normal content, leading to the concept of steganographic security. However, incorporating steganographic algorithms into the language model's prediction and sampling processes often introduces distributional distortions (Holub and Fridrich, 2013; Yang et al., 2019b). To address this challenge, recent work has explored approaches aimed at achieving *provable security* (Hopper et al., 2002, 2009) in steganography.

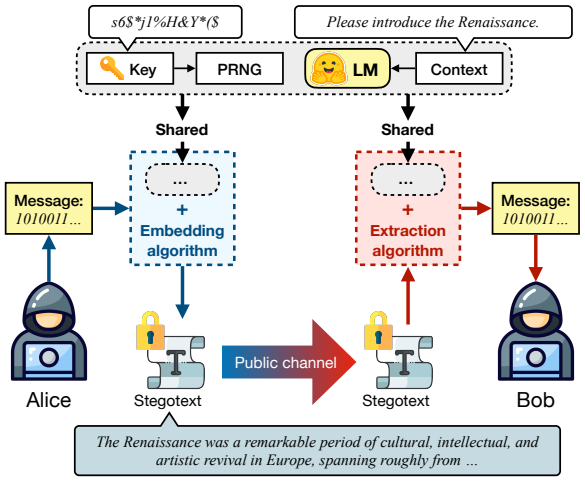


Figure 1: A schematic diagram of linguistic steganography, where PRNG refers to a pseudo-random number generator for controlling randomness and reproducibility. Alice embeds the secret message into a steganographic text (stegotext), and Bob extracts the secret message from the received stegotext.

Despite the progress of provably secure approaches, each method has its own limitations. ADG (Zhang et al., 2021) fails to strictly preserve the original probability distribution by grouping



*Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 890–907
July 2-7, 2026 

candidate tokens at each generative step. Meteor (Kaptchuk et al., 2021), which is based on arithmetic coding (AC) (Ziegler et al., 2019), inevitably distorts the original distribution when encoding intervals. Although iMEC (de Witt et al., 2023), Discop (Ding et al., 2023), and SparSamp (Wang et al., 2025) maintain the original probability distribution, the first two suffer from limited embedding capacity and slow embedding speeds. SparSamp, the current state-of-the-art method, still falls short of achieving full embedding efficiency (i.e., 100% entropy utilization). All of these methods process the secret message as discrete bits, which can result in losses in embedding efficiency or imperceptibility when incorporating some mechanisms for unique extraction.

These limitations motivate us to revisit a classical entropy-coding technique, range coding (RC; Martin, 1979) which encodes messages in decimal form rather than in bits, and explore whether it can achieve (i) preservation of the original probability distribution, (ii) full embedding efficiency, and (iii) high embedding speed.[1]

We begin by proposing a vanilla RC steganography (Section 3). To the best of our knowledge, it is the first attempt to embed a secret message **entirely in decimal form**. However, we discover its security issues: (1) distortion on probability distribution and (2) randomness reuse.

To address these issues, we present **rotation range-coding (RRC)** steganography, which incorporates a rotation mechanism (Section 4). This mechanism ensures zero KL divergence at every generative step and prevents reuse of randomness. The method is **training-free** and **plug-and-play**.

In addition, we provide theoretical analysis and proofs regarding zero KL divergence and computational security, supporting the **provable security** of our RRC steganography (Section 5). Additionally, the approximate 100% entropy utilization for embedding capacity is analyzed and empirically validated. The key advantage of RC-based steganography is *ensuring unique extraction internally, without requiring any trade-offs or external restrictions.*

Experimental results in various language models demonstrate that RRC steganography consistently achieves **the highest embedding efficiency** (i.e., entropy utilization) and **great embedding speed** (up to 1554.66 bits/s in GPT-2) compared to all provably secure baseline methods (Section 6). Experiments also show that RRC steganography has strong scalability and anti-steganalysis capacity.

[1]When all operations are carried out in decimals, the original distribution at each generative step is rescaled to a new range, without distortion caused by constructing discrete bits.

## 2 Background and Preliminaries

### 2.1 Language Model Basics

A language model (LM) has a vocabulary $\mathcal{V}$, a set of tokens. Consider a sequence of LM-generated $T$ tokens $\{s^{(t)}\} \in \mathcal{V}^T$. Tokens with negative indices, $[s^{(-N_p)}, \ldots, s^{(-1)}]$, represent a *prompt* of length $N_p$ and $[s^{(0)}, \ldots, s^{(T-1)}]$ are tokens generated by an LM in response to the prompt.

The next token prediction by an LM at position $t$, is a function whose input is a sequence of known tokens $[s^{(-N_p)}, \ldots, s^{(t-1)}]$ which consists of a prompt and the first $t-1$ LM-generated tokens. Then it outputs a logit vector, corresponding to each token in $\mathcal{V}$. These logits are then converted into a discrete probability distribution $\boldsymbol{p}^{(t)} = (p_1^{(t)}, \ldots, p_{|\mathcal{V}|}^{(t)})$ over the vocabulary, via a $\mathrm{softmax}$ operator (commonly). The next token is then sampled from $\boldsymbol{p}^{(t)}$ using either standard multinomial sampling, beam search, or other strategies.

### 2.2 LM-based Steganography

Alice (the sender) wants to communicate a secret message $m_s \sim U(\{0,1\}^l)$ with Bob (the receiver) by embedding it in a natural-language text $t_s$ (a stegotext). Alice and Bob have agreed on an embedding function $\mathcal{S}_{\text{emb}}$ and an extracting function $\mathcal{S}_{\text{ext}}$ that perform steganography, achieved by a language model, $\mathcal{M}$. These two functions are supposed to be invertible. In other words, $\mathcal{S}_{\text{emb}}(\mathcal{M}, m_s) = t_s$, $\mathcal{S}_{\text{ext}}(\mathcal{M}, t_s) = m'_s$.

### 2.3 Security of Steganography

Cachin (1998) first modeled steganographic security from the perspective of information theory, where given an object $\mathbf{x}$, the security of a stegosystem can be quantified by Kullback-Leibler (KL) divergence between the cover distribution (the channel distribution) $P_c$ and the stego distribution $P_s$,

$$D_{\text{KL}}(P_c||P_s) = \sum_{\mathbf{x}\in\mathcal{C}} P_c(\mathbf{x}) \log \frac{P_c(\mathbf{x})}{P_s(\mathbf{x})} \quad (1)$$

which typically measures how different the two distributions are. When $D_{\text{KL}}(P_c||P_s) = 0$, the stegosystem is considered to be *perfectly secure* in this perspective of information theory.

Benefiting from the explicit generative models that can predict probability distributions, the above

definition of steganographic security can be modeled into another goal, that is, steganography is indistinguishable from the normal generation process, i.e., random sampling (Ding et al., 2023).

In addition, from the perspective of computational security (Hopper et al., 2002; Katzenbeisser and Petitcolas, 2002), steganography is secure against chosen hiddentext attacks, if for all probabilistic polynomial time (PPT) adversary detection $\mathcal{A}_\mathcal{D}$, it holds:

$$|\Pr[\mathcal{A}_\mathcal{D}(x_s) = 1] - \Pr[\mathcal{A}_\mathcal{D}(x_c) = 1]| < \text{negl}(\lambda) \quad (2)$$

where $x_s$ is the stegotext, $x_c$ is the normally generated cover text, $\lambda$ is the security parameter of the shared key $K$ (usually the length of $K$), and $\text{negl}(\lambda)$ is a negligible function concerning $\lambda$.

### 2.4 Statistical Imperceptibility of LM-based Steganography

Following the previous formulation (Dai and Cai, 2019; Shen et al., 2020), statistical imperceptibility refers to the similarity between the true language model $\mathcal{M}^t$ in the monitored channel and $\mathcal{M}^s$, the language model $\mathcal{M}$ integrated with steganographic algorithms. Specifically, the total variation distance (TVD) is used to measure statistical imperceptibility. Consider the TVD between $\mathcal{M}^t$ and $\mathcal{M}^s$, i.e. $d(\mathcal{M}^t, \mathcal{M}^s)$, by triangle inequality:

$$d(\mathcal{M}^t, \mathcal{M}^s) \leq d(\mathcal{M}^t, \mathcal{M}) + d(\mathcal{M}, \mathcal{M}^s). \quad (3)$$

As $d(\mathcal{M}^t, \mathcal{M})$ is a criterion to measure the original language model, which is limited by the research on language models. Thus, $d(\mathcal{M}, \mathcal{M}^s)$ is the main focus of linguistic steganography.

According to Pinsker's inequality (Fedotov et al., 2003) and additivity of KL divergence, $d(\mathcal{M}, \mathcal{M}^s)$ can be further decomposed in each step, that is:

$$d(\mathcal{M}, \mathcal{M}^s) \leq \sqrt{\frac{\ln 2}{2} \sum_{t=1}^{\infty} D_{\text{KL}}(\boldsymbol{p}^{(t)} || \hat{\boldsymbol{p}}^{(t)})} \quad (4)$$

where $\boldsymbol{p}^{(t)}$ is the original probability distribution at $t^{th}$ step, and $\hat{\boldsymbol{p}}^{(t)}$ is transformed from $\boldsymbol{p}^{(t)}$ via sampling and encoding. Hence, linguistic steganography could aim to minimize $D_{\text{KL}}(\boldsymbol{p}^{(t)} || \hat{\boldsymbol{p}}^{(t)})$, in order to obtain relative near-imperceptibility. Some derivation is skipped here, as details are verified in (Dai and Cai, 2019; Shen et al., 2020; Fedotov et al., 2003).

Specifically, Figure 2 illustrates how distribution distortion can compromise imperceptibility.

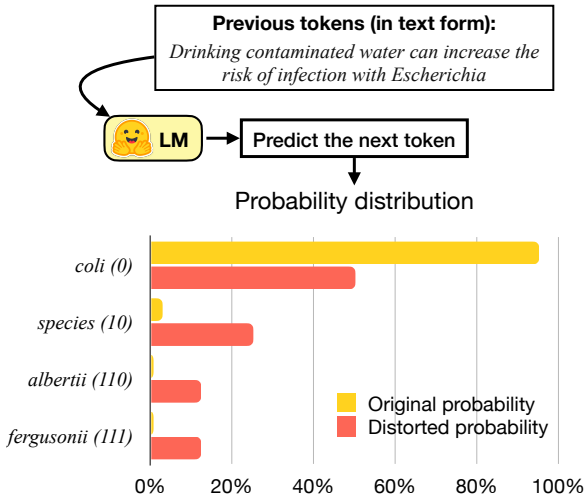


Figure 2: An example of how steganographic encoding (using Huffman coding (Yang et al., 2019b)) can alter the original top-4 probability distribution predicted by a language model. Specifically, a pair such as *coli (0)* denotes a candidate token along with its Huffman code, and the distorted probabilities are computed based on a random secret message.

In this low-entropy example, "*coli*" has an overwhelmingly high probability of being the next token, making any alternative token extremely unlikely. If steganography introduces distribution distortion (such as through Huffman encoding), it may create detectable patterns that could be exploited by a steganalysis detector.

### 2.5 Related Work with Zero KL Divergence

Achieving $D_{\text{KL}}(\boldsymbol{p}^{(t)} || \hat{\boldsymbol{p}}^{(t)}) = 0$ (for each $t$) is a desirable objective in steganography. Ding et al. (2023) proposed Discop, a representative zero-KL provably secure steganographic method. At each generative step, the message determines which *distribution copy* to sample. However, Discop must avoid overlaps between its rotated copies to keep extraction unique: whenever overlaps occur, it embeds fewer bits. Recently, Wang et al. (2025) introduced SparSamp, which embeds messages by combining them with pseudo-random numbers to obtain message-derived random numbers for sampling. To achieve uniquely extractable, SparSamp must sparsify its sampling grid, which leaves a small capacity gap. More related methods and details are shown in Appendix A.

## 3 Vanilla Range-Coding Steganography

In this section, we tentatively start by describing a simple *vanilla* version of range-coding (RC)

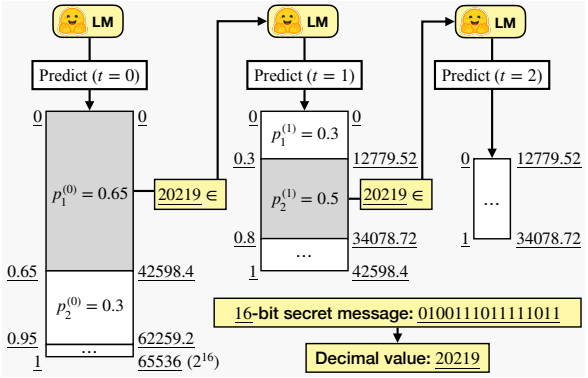


Figure 3: An example of procedures for embedding a 16-bit secret message into a text via the vanilla RC steganography. The interval is iteratively narrowed until it can **uniquely** represent the decimal value 20219.

steganography, which directly applies RC to linguistic steganography.

### 3.1 Embedding & Extraction

Figure 3 briefly illustrates how vanilla RC steganography embeds a message into a text. In range coding, all the information can be represented in decimals and ranges (intervals). In this example, the 16-bit message is interpreted as an integer whose decimal representation is 20219, and the interval is initialized as $[0, 2^{16}) = [0, 65536)$.

Algorithm 1 outlines how the sender (Alice) embeds the secret message $m_s$ into the text $t_s$ using vanilla RC steganography. Specifically, $m_s$ is first decimalized to $d_s$ in Line 1, and all subsequent procedures operate directly on $d_s$ rather than on a bitstream. In Line 2, the initial interval is set to $[0, 2^l)$, where $l$ is the length of $m_s$. During subsequent iterative processes (Lines 3–9), the interval is progressively narrowed at each step. The iteration ends when the midpoint of the interval is exactly rounded to $d_s$ (which ensures **uniqueness**).

Algorithm 2 outlines how the receiver (Bob) extracts the secret message $m_s$ from the received text $t_s$. The initial interval is narrowed according to each token received, and $d_s$ is the rounded value of the midpoint of the final interval. Finally, the extraction result $m_s$ is binarized from $d_s$.

### 3.2 Security Issues

For vanilla RC steganography, there can be two security issues:

*1) Distortion on probability distribution.* Taking the first generative step in Figure 3 as an example, the (softmax) probability of $\text{token}_1$ $(t = 0)$, $p_1^{(0)}$, is $0.65$, and its interval is $[0, 42598.4)$. Considering a

**Algorithm 1** Vanilla RC steganography (embed)

**Input:**
Context (initial previous tokens), $C$
Language model, $\mathcal{M}$
Message length, $l$
Secret message, $m_s$

**Output:**
Steganographic text, $t_s$

1: $d_s \leftarrow \text{bin2dec}(m_s)$; // `Decimalize`
2: $[L, R) \leftarrow [0, 2^l)$; // `Initialize interval`
3: **while** $\text{round}(\frac{L+R}{2}) \neq d_s$ **do**
4: $\quad \boldsymbol{p}^{(t)} \leftarrow \mathcal{M}(C)$; // `Predict probs`
5: $\quad \boldsymbol{c}^{(t)} \leftarrow 0 || \boldsymbol{p}^{(t)}.\text{cumsum}()$; // `Cumulate probs`
6: $\quad \boldsymbol{c}'^{(t)} \leftarrow L + (R - L) \times \boldsymbol{c}^{(t)}$; // `Rescale`
7: $\quad$ Select $\text{token}_i$ so that $d_s \in [\boldsymbol{c}'^{(t)}[i-1], \boldsymbol{c}'^{(t)}[i])$;
8: $\quad [L, R) \leftarrow [\boldsymbol{c}'^{(t)}[i-1], \boldsymbol{c}'^{(t)}[i])$;
9: $\quad C \leftarrow C || \text{token}_i$;
10: Detokenize $C$ into $t_s$;
11: **return** $t_s$

**Algorithm 2** Vanilla RC steganography (extract)

**Input:**
Context (initial previous tokens), $C$
Language model, $\mathcal{M}$
Message length, $l$
Steganographic text, $t_s$

**Output:**
Secret message, $m_s$

1: Tokenize $t_s$ into $S$;
2: $[L, R) \leftarrow [0, 2^l)$; // `Initialize interval`
3: **for** $t = 0, 1, \ldots, |S| - |C| - 1$ **do**
4: $\quad \boldsymbol{p}^{(t)} \leftarrow \mathcal{M}(S[: |C| + t])$; // `Predict probs`
5: $\quad \boldsymbol{c}^{(t)} \leftarrow 0 || \boldsymbol{p}^{(t)}.\text{cumsum}()$; // `Cumulate probs`
6: $\quad \boldsymbol{c}'^{(t)} \leftarrow L + (R - L) \times \boldsymbol{c}^{(t)}$; // `Rescale`
7: $\quad$ Select $\text{token}_i$ so that $\text{token}_i = S[|C| + t + 1]$;
8: $\quad [L, R) \leftarrow [\boldsymbol{c}'^{(t)}[i-1], \boldsymbol{c}'^{(t)}[i])$;
9: $d_s \leftarrow \text{round}(\frac{L+R}{2})$;
10: $m_s \leftarrow \text{dec2bin}(d_s).\text{zfill}(l)$; // `Binarize` & `Fill 0`
11: **return** $m_s$

random 16-bit secret message $m_s \sim U(\{0, 1\}^{16})$, so $d_s \sim U(\{0, 1, \ldots, 2^{16} - 1\})$ ($d_s$ is a **discrete** uniform random variable). Then, the steganographic sampled probability for $\text{token}_1$ $(t = 0)$ is $P(d_s \in [0, 42598.4)) = \frac{42599}{65536} \neq 0.65 = p_1^{(0)}$. Thus, distortion on probability distribution occurs and zero KL divergence or perfect security cannot hold. Even though it could be mitigated when lengthening $m_s$ ($l$ can be set greater than the tensor precision). However, similar to AC-based steganography, iterative interval narrowing still causes noticeable distortion in small intervals.

*2) Randomness reuse.* According to Kaptchuk et al. (2021), reusing randomness in multiple sampling events could expose features and bias to detec-

**Algorithm 3** Rotation RC steganography (embed)

**Input:**
Context (initial previous tokens), $C$
Pseudo-random number generator, PRNG
Language model, $\mathcal{M}$
Symmetric key (seed), $K$
Message length, $l$
Secret message, $m_s$

**Output:**
Steganographic text, $t_s$

1: $d_s^{(-1)} \leftarrow \text{bin2dec}(m_s)$; // `Decimalize`
2: PRNG.set_seed($K$);
3: $[L^{(-1)}, R^{(-1)}) \leftarrow [0, 2^l)$; // `Initialize interval`
4: **for** $t = 0, 1, \ldots$ **do**
5: $\quad \boldsymbol{p}^{(t)} \leftarrow \mathcal{M}(C)$; // `Predict probs`
6: $\quad \boldsymbol{c}^{(t)} \leftarrow 0||\boldsymbol{p}^{(t)}.\text{cumsum}()$; // `Cumulate probs`
7: $\quad \Delta^{(t-1)} = R^{(t-1)} - L^{(t-1)}$;
8: $\quad \boldsymbol{c'}^{(t)} \leftarrow L^{(t-1)} + \Delta^{(t-1)} \times \boldsymbol{c}^{(t)}$; // `Rescale`
9: $\quad o^{(t)} \leftarrow U(0,1).\text{sample}(\text{PRNG}^{(t)})$;
10: $\quad d_s^{(t)} \leftarrow L^{(t-1)} + (d_s^{(t-1)} - L^{(t-1)} + o^{(t)} \times \Delta^{(t-1)}) \bmod \Delta^{(t-1)}$; // `Rotate`
11: $\quad$ Select $\text{token}_i$ so that $d_s^{(t)} \in [\boldsymbol{c'}^{(t)}[i-1], \boldsymbol{c'}^{(t)}[i])$;
12: $\quad [L^{(t)}, R^{(t)}) \leftarrow [\boldsymbol{c'}^{(t)}[i-1], \boldsymbol{c'}^{(t)}[i])$;
13: $\quad C \leftarrow C||\text{token}_i$;
14: $\quad$ **if** $\frac{L^{(t)}+R^{(t)}}{2} - d_s^{(t)} \in (-0.5, 0.5]$ **then**
15: $\quad\quad$ **break**
16: Detokenize $C$ into $t_s$;
17: **return** $t_s$

**Algorithm 4** Rotation RC steganography (extract)

**Input:**
Context (initial previous tokens), $C$
Pseudo-random number generator, PRNG
Language model, $\mathcal{M}$
Symmetric key (seed), $K$
Message length, $l$
Steganographic text, $t_s$

**Output:**
Secret message, $m_s$

1: Tokenize $t_s$ into $S$;
2: PRNG.set_seed($K$);
3: $[L^{(-1)}, R^{(-1)}) \leftarrow [0, 2^l)$; // `Initialize interval`
4: $t_{\text{end}} \leftarrow |S| - |C| - 1$;
5: **for** $t = 0, 1, \ldots, t_{\text{end}}$ **do**
6: $\quad \boldsymbol{p}^{(t)} \leftarrow \mathcal{M}(S[:|C|+t])$; // `Predict probs`
7: $\quad \boldsymbol{c}^{(t)} \leftarrow 0||\boldsymbol{p}^{(t)}.\text{cumsum}()$; // `Cumulate probs`
8: $\quad \Delta^{(t-1)} \leftarrow R^{(t-1)} - L^{(t-1)}$;
9: $\quad \boldsymbol{c'}^{(t)} \leftarrow L^{(t-1)} + \Delta^{(t-1)} \times \boldsymbol{c}^{(t)}$; // `Rescale`
10: $\quad$ Select $\text{token}_i$ so that $\text{token}_i = S[|C|+t+1]$;
11: $\quad [L^{(t)}, R^{(t)}) \leftarrow [\boldsymbol{c'}^{(t)}[i-1], \boldsymbol{c'}^{(t)}[i])$;
12: $\text{mid}^{(t_{\text{end}})} \leftarrow (L^{(t_{\text{end}})} + R^{(t_{\text{end}})})/2$;
13: **for** $t = t_{\text{end}}, \ldots, 1, 0$ **do**
14: $\quad o^{(t)} \leftarrow U(0,1).\text{sample}(\text{PRNG}^{(t)})$;
15: $\quad \text{mid}^{(t-1)} \leftarrow L^{(t-1)} + (\text{mid}^{(t)} - L^{(t-1)} - o^{(t)} \times \Delta^{(t-1)}) \bmod \Delta^{(t-1)}$; // `Rotate reversely`
16: $d_s^{(-1)} \leftarrow \text{round\_half\_down}(\text{mid}^{(-1)})$;
17: $m_s \leftarrow \text{dec2bin}(d_s^{(-1)}).\text{zfill}(l)$; //`Binarize` & `Fill 0`
18: **return** $m_s$

tors. Therefore, only using the bits of the message as the randomness or encrypting the message with a pseudo-random cipher, as in a public-key solution, is insecure because multiple samplings will be forced to reuse randomness.

## 4 Rotation Range-Coding Steganography

Considering the security issues discussed above, we propose a rotation range-coding (RRC) steganographic method. Instead of directly using the constant $d_s$, our proposed rotation mechanism updates $d_s^{(t-1)}$ to $d_s^{(t)}$ at each time step $t$ (initial $d_s = d_s^{(-1)}$), with the following objectives:

- To transform the discrete uniform random variable $d_s$ to a **continuous** uniform random variable $d_s^{(t)}$ at each $t$, thereby preserving the original probability distribution and ensuring zero KL divergence.
- To introduce *fresh* randomness at each $t$, thereby preventing the reuse of randomness.

### 4.1 Embedding of RRC Steganography

Algorithm 3 outlines the embedding procedures of RRC steganography (Algorithm 3 shares several steps with Algorithm 1). Inspired by other provably secure methods, we employ a pseudo-random number generator (PRNG) and a symmetric key $K$ to generate pseudo-random numbers for the following sampling (Line 2), ensuring reproducibility and correct extraction. Note that the pseudo-random numbers generated by PRNG are used to control the sampling of the offset $o \sim U(0,1)$ at each $t$ (Line 9), and then $o$ is used to rotate $d_s^{(t-1)}$ to $d_s^{(t)}$ (Line 10). Besides, the termination condition in RRC steganography is $\frac{L^{(t)}+R^{(t)}}{2} - d_s^{(t)} \in (-0.5, 0.5]$ (Lines 14–15), which is tailored for ***unique extraction*** *and avoiding generating unnecessary tokens*.

### 4.2 Extraction of RRC Steganography

Algorithm 4 outlines how the receiver (Bob) extracts the secret message $m_s$ from the steganographic text $t_s$. As Alice and Bob have agreed on PRNG and symmetric key $K$, Bob can synchronize each $t$-time rotation with Alice and reproduce each range of $d_s^{(t)}$ according to each interval $[L^{(t)}, R^{(t)})$ at each $t$. Based on the termination condition of the embedding algorithm, the end time $t_{\text{end}} = |S| - |C| - 1$, and $\text{mid}^{(t_{\text{end}})} =$

$(L^{(t_{\text{end}})} + R^{(t_{\text{end}})})/2$, there is:

$$\text{mid}^{(t_{\text{end}})} - d_s^{(t_{\text{end}})} \in (-0.5, 0.5]$$

$$d_s^{(t_{\text{end}})} \in [\text{mid}^{(t_{\text{end}})} - 0.5, \text{mid}^{(t_{\text{end}})} + 0.5).$$

Considering linear transformation and the rotation in embedding, for each $t$ there is (in Line 15):

$$\text{mid}^{(t-1)} = L^{(t-1)} + (\text{mid}^{(t)} - L^{(t-1)} - o^{(t)} \times \Delta^{(t-1)}) \bmod \Delta^{(t-1)}$$

$$d_s^{(t)} \in [\text{mid}^{(t)} - 0.5, \text{mid}^{(t)} + 0.5)$$

where $\Delta^{(t-1)} = R^{(t-1)} - L^{(t-1)}$. After iteration, as $d_s^{(-1)} \in \mathbb{Z}$, there is (Line 16):

$$d_s^{(-1)} \in [\text{mid}^{(-1)} - 0.5, \text{mid}^{(-1)} + 0.5)$$

$$d_s^{(-1)} = \text{round\_half\_down}(\text{mid}^{(-1)})$$

where round_half_down means that when a number is exactly halfway between two possible rounded values (e.g., 2.5), round_half_down rounds toward the smaller rounded values (e.g., 2).

Therefore, in RRC steganography, $d_s^{(-1)}$ can be computed **uniquely** by Bob, and then $m_s$ is binarized from $d_s^{(-1)}$ (Line 17).

## 5 Analysis of RRC Steganography

### 5.1 Proof of Zero KL Divergence

Considering rotation, we first introduce Proposition 1 (the rigorous proof is shown in Appendix B.1), and then explain how the original probability distribution is preserved (Proposition 2).

**Proposition 1.** $d_s^{(t)} \sim U(L^{(t-1)}, R^{(t-1)})$.

**Proposition 2.** *In Line 11 (Algorithm 3), the selected probability of each* $\text{token}_i$ *equals* $p_i^{(t)}$.

*Proof.* Considering the interval construction from $\boldsymbol{p}^{(t)}$ to $\boldsymbol{c}'^{(t)}$ (Lines 5–8 in Algorithm 3), $\boldsymbol{p}^{(t)} = (p_1^{(t)}, \ldots, p_{|\mathcal{V}|}^{(t)})$, $\boldsymbol{c}^{(t)} = (c_o^{(t)}, c_1^{(t)}, \ldots, c_{|\mathcal{V}|}^{(t)}) = (0, \sum_{k=1}^{1} p_k, \ldots, \sum_{k=1}^{|\mathcal{V}|} p_k)$, and $\boldsymbol{c}'^{(t)} = L^{(t-1)} + \Delta^{(t-1)} \times \boldsymbol{c}^{(t)} = (L^{(t-1)}, L^{(t-1)} + \Delta^{(t-1)} \times \sum_{k=1}^{1} p_k, \ldots, R^{(t-1)})$. According to Proposition 1, $d_s^{(t)} \sim U(L^{(t-1)}, R^{(t-1)})$, so the probability density function is:

$$f_{d_s^{(t)}}(x) = \begin{cases} \frac{1}{\Delta^{(t-1)}}, & \text{if } x \in [L^{(t-1)}, R^{(t-1)}), \\ 0, & \text{otherwise} \end{cases}$$

$$P\left(d_s^{(t)} \in [\boldsymbol{c}'^{(t)}[i-1], \boldsymbol{c}'^{(t)}[i])\right)$$
$$= \int_{\boldsymbol{c}'^{(t)}[i-1]}^{\boldsymbol{c}'^{(t)}[i]} f_{d_s^{(t)}}(x)\, dx = \frac{\boldsymbol{c}'^{(t)}[i] - \boldsymbol{c}'^{(t)}[i-1]}{\Delta^{(t-1)}}$$
$$= \frac{(\Delta^{(t-1)} \times \sum_{k=1}^{i} p_k) - (\Delta^{(t-1)} \times \sum_{k=1}^{i-1} p_k)}{\Delta^{(t-1)}}$$
$$= \sum_{k=1}^{i} p_k - \sum_{k=1}^{i-1} p_k = p_i.$$

Therefore, the probability of selecting $\text{token}_i$ is exactly $p_i^{(t)}$, the original LM distribution. □

As RRC steganography does not change the original predicted probability by the LM for each token, the KL divergence between the original and the steganographic probability distribution is zero. RRC steganography possesses perfect statistical imperceptibility (according to Section 2.4).

### 5.2 Computational Security

From the perspective of computational security (Section 2.3 and Equation 2), we prove our RRC steganography is a secure method, for which details are shown in Appendix B.3.

### 5.3 Embedding Capacity

First, we consider when the embedding ends according to the proposed termination condition, there is a proposition (its proof is shown in Appendix B.2):

**Proposition 3.** $\Delta^{(t)} \leq 1$ *is a sufficient condition for the embedding termination* $\frac{L^{(t)}+R^{(t)}}{2} - d_s^{(t)} \in (-0.5, 0.5]$ *(Lines 14–15 in Algorithm 3).*

Then, given the initial interval $[L^{(-1)}, R^{(-1)}) = [0, 2^n)$ and the interval length $\Delta^{(-1)} = 2^n$, the interval length at $t$ is:

$$\Delta^{(t)} = 2^n \cdot \prod_{i=0}^{t} p_{\text{output}}^{(i)} \tag{5}$$

and there is:

$$\frac{\Delta^{(t)}}{\Delta^{(t-1)}} = p_{\text{output}}^{(t)} \tag{6}$$

where $p_{\text{output}}^{(i)}$ is the probability of the output token $(i = 0, 1, \ldots, t)$.

According to Proposition 3, when $\Delta^{(t)} \leq 1$, the embedding iteration ends. Considering information theory and Equation 5, the interval shrinkage rate is determined by the entropy of the probability distribution. The average amount of information per iteration is $H^{(t)}$, and the total amount of information is

required to cover $n$ bits of the initial interval. Therefore, the number of loops is satisfied: $\sum_{i=0}^{t} H^{(t)} \geq n$ where $H^{(t)} = -\sum_{i=1}^{|\mathcal{V}|} p_i^{(t)} \log_2 p_i^{(t)}$, and let the average entropy is $H_{\text{avg}}$, so that the loop number (which is exactly the number of the generated tokens) is: $N_{\text{token}} \approx \frac{n}{H_{\text{avg}}}$. Thus, the embedding capacity (bits per token) can be represented as: $\frac{n}{N_{\text{token}}} \approx H_{\text{avg}}$. RRC steganography can achieve approximate **100% utilization of entropy**.

### 5.4 Complexity

RC-based steganography (including vanilla RC steganography and RRC steganography) requires updating probability intervals after each step, resulting in a time complexity of $O(|\mathcal{V}|)$ for each generative step. The complexity can be computed following the approaches used in AC steganography (Ziegler et al., 2019).

## 6 Experiments

To validate the security and efficiency of our RRC steganography, we evaluated it compared to a series of methods toward provable security in this era, including arithmetic coding (AC) (Ziegler et al., 2019), ADG (Zhang et al., 2021), Meteor (Kaptchuk et al., 2021), iMEC (de Witt et al., 2023), Discop (Ding et al., 2023), and SparSamp (Wang et al., 2025).

### 6.1 Setup

To validate the generalizability of our steganographic method, we implemented it using three language models of various scales: GPT-2 (Radford et al., 2019),[2] OPT-1.3b (Zhang et al., 2022),[3] and Llama-2-7b (Touvron et al., 2023).[4]

For each language model and steganographic method, 1,000 samples were generated using 1,000 different initial contexts. These contexts consist of the first 10 words from sequences randomly selected from the C4 dataset.[5]

All the experiments were conducted with top-$p$ ($p = 1.0$) sampling, i.e., encoding the entire vocabulary $\mathcal{V}$, and 1.0 temperature. Experiments were implemented in Python 3.12.7 with Torch 2.5.0, and accelerated by using RTX 6000 Ada Generation GPUs. Besides, considering the precision limitation of the tensor, we imported Python's `decimal` module for computing with sufficient precision. Otherwise without an enough precision, errors or incorrect extractions could occur.

[2] https://huggingface.co/openai-community/gpt2
[3] https://huggingface.co/facebook/opt-1.3b
[4] https://huggingface.co/meta-llama/Llama-2-7b-hf
[5] https://huggingface.co/datasets/allenai/c4

### 6.2 Metrics

**Avg (Max) KLD**, *a security metric*, refers to the average (or maximum) value of the KL divergence $D_{\text{KL}}(\boldsymbol{p}^{(t)} || \hat{\boldsymbol{p}}^{(t)})$ in all steps, which indicates the average (or maximum) degree to the original distribution by steganography (Ding et al., 2023). Specifically, in $D_{\text{KL}}(\boldsymbol{p}^{(t)} || \hat{\boldsymbol{p}}^{(t)})$, $\boldsymbol{p}^{(t)}$ is the probability distribution of the original candidate pool, and $\hat{\boldsymbol{p}}^{(t)}$ is the probability distribution of the modified (steganographic) candidate pool at each $t$.

**Embedding capacity** refers to the average number of bits embedded per generated token.

**Entropy utilization (embedding efficiency)** refers to the ratio of embedding capacity to the average entropy over all steps.

**Embedding speed** refers to the average seconds required to embed a single secret bit.

### 6.3 Main Results

Table 1 presents the average results across various metrics for GPT-2. Besides, the results for OPT-1.3b and Llama-2-7b are shown in Tables 4 and 5 in Appendix C.1. Both Meteor and Discop were evaluated in two configurations: sorted and unsorted. For each metric, the best-performing result is highlighted in **bold**, while the second-best is indicated with underline. The embedded secret message was a randomly generated 128-bit sequence. In addition, multinomial sampling generation (random sampling) was also carried out for comparison. The key findings from these experiments are as follows:

*1)* As analyzed in Wang et al. (2025), iMEC, Discop and SparSamp are probability-preserving. The zero KL divergence of RRC steganography is proved in Section 5.1, thus these methods and our RRC steganography can achieve 0 KL divergence.

*2)* Our RRC steganography empirically achieves around 100% entropy utilization, which complies with the theoretical analysis in Section 5.3, which denotes the **100% embedding efficiency**. Besides, the entropy utilization of our method is steadily superior to other baseline methods when implemented in different language models.

*3)* Our RRC steganography achieves a highly competitive embedding speed, which is the **fastest** in GPT-2 (up to 1554.66 bits/s). However, in the other two language models, our method obtains the second-fastest speeds.

| Method | Avg / Max KLD (bits/token) ↓ | Capacity (bits/token) ↑ | Entropy (bits/token) | Utilization (%) ↑ | Speed (bits/s) ↑ |
|---|---|---|---|---|---|
| Multinomial sampling | 0 / 0 | N/A | 5.86 | N/A | N/A |
| AC | 1.95E-03 / 3.01E-02 | 5.86 | 5.87 | 99.83 | 1025.36 |
| ADG | 1.60E-04 / 1.57E-03 | 4.81 | 5.89 | 81.60 | 36.45 |
| Meteor w/o sort | 4.22E-02 / 1.16E-01 | 4.17 | 5.79 | 71.96 | 950.27 |
| Meteor w/ sort | 4.11E-02 / 1.16E-01 | 4.77 | 5.81 | 82.08 | 25.25 |
| iMEC | **0 / 0** | 4.16 | 5.83 | 71.44 | 27.30 |
| Discop w/o sort | **0 / 0** | 2.31 | 5.90 | 39.31 | 218.34 |
| Discop w/ sort | **0 / 0** | 5.58 | 5.86 | 95.17 | 44.30 |
| SparSamp | **0 / 0** | 5.74 | 5.93 | 96.76 | 1267.82 |
| RRC steganography (ours) | **0 / 0** | **5.93** | 5.93 | **99.98** | **1554.66** |

Table 1: Quantitative comparison with previous steganographic methods on GPT-2.

| Message length (bits) | 32 | 64 | 128 | 256 | 512 | 1024 | 2048 | 4096 | 8192 |
|---|---|---|---|---|---|---|---|---|---|
| Utilization (%) ↑ | 99.86 | 99.19 | 99.98 | 99.89 | 99.77 | 99.93 | 100.35 | **100.43** | 99.99 |
| Speed (bits/s) ↑ | 1390.23 | 1496.88 | 1554.66 | **1572.40** | 1475.68 | 1511.83 | 1191.88 | 880.82 | 604.76 |
| Running time (s) | 0.023 | 0.043 | 0.082 | 0.163 | 0.374 | 0.677 | 1.718 | 4.650 | 13.546 |

Table 2: Average results on utilization, speed and running time of RRC steganography across various message lengths $l$ on GPT-2.

Additionally, Appendix C.5 presents ablation studies comparing vanilla RC (Section 3) and provable secure RRC steganography (Section 4).

### 6.4 Scalability of RRC Steganography

Steganography based on range coding has a distinct characteristic, that is, it embeds the entire secret message using decimal values, rather than embedding it bit by bit. In other words, the minimum unit of embedding is the complete $l$-bit message itself. If the embedding process is not completed, the message is considered not embedded at all. Therefore, scalability should be considered, as it reflects how well RRC steganography can support the secret message with various lengths.

Table 2 lists the average utilization, speed, and running time when RRC steganography embeds the secret message with various lengths on GPT-2. Tables 6 and 7 (in Appendix C.2) show results conducted in OPT-1.3b and Llama-2-7b. The number of generated texts for each message length is 1000. From this table, we can find that:

*1)* RRC steganography maintains steady entropy utilization around 100%.[6]

*2)* When the message length varies from 64 to 1024 bits, the embedding speed is steadily around 1500 bits per second.

*3)* Our method supports messages with significantly higher bit lengths (with 8192 bits not representing an upper limit), enabled by the scalable precision of Python's `decimal` module.

### 6.5 Anti-steganalysis Capacity

In this section, we evaluated the ability of our method to evade Eve's detection using steganalysis techniques, specifically through a fine-tuned discriminator. The discriminators used for detection were fine-tuned versions of the pretrained BERT (Devlin et al., 2019) and RoBERTa (Conneau et al., 2019) models, respectively. Table 8 in Appendix C.3 presents the steganalysis accuracies for steganographic texts generated by three different language models. Accuracies around 50% indicate that the *steganalysis methods do not perform better than random guessing* in detecting texts.

### 6.6 Human Evaluation

We randomly mixed the steganographic texts and cover texts (250 samples per category) and asked three human evaluators to judge whether each text was machine-generated with embedded hidden information. As shown in Table 3, the consistently low detection scores indicate that human evaluators struggle to distinguish steganographic texts from cover texts, providing strong evidence for the perceptual imperceptibility of RRC steganography. Notably, statistical classifiers already outperform

[6]The stability outperforms SparSamp whose utilization is sensitive to message length and the maximum length is only 1023 (according to the data disclosed in Wang et al. (2025)).

| Model | Accuracy | Precision | Recall | F1 |
|---|---|---|---|---|
| GPT-2 | 47.8% | 47.8% | 47.2% | 47.5% |
| OPT-1.3B | 46.6% | 46.2% | 41.2% | 43.6% |
| Llama-2-7B | 50.6% | 50.6% | 52.4% | 51.5% |

Table 3: Human evaluation results across different models.

humans at this discrimination task, as they can aggregate subtle statistical cues that are imperceptible to human readers.

## 7 Conclusion

In this paper, we explore the use of a relatively simple and classical approach, range coding (RC), to achieve provably secure steganography as well as high embedding efficiency and speed. However, two key security challenges arise: (1) distortion of the probability distribution and (2) reuse of randomness. To address these issues, we propose rotation range-coding (RRC) steganography, and provide theoretical explanations and proofs for it. RRC empirically outperforms the baseline methods in both embedding efficiency and capacity, while also achieving competitive embedding speed. Moreover, RRC steganography is training-free, model-agnostic, and straightforward to implement, making it a practical foundation for future extensions to multi-modal steganography and for deployment in real-world, privacy-preserving communication.

## Limitations

In the symmetric steganographic system based on RRC steganography, Alice and Bob must agree on the secret message length $l$ before the steganographic communication, which is used to initialize the interval $[0, 2^l)$ for both sides and fill "0" in extraction (Line 17 in Algorithm 4).

For the analysis of embedding capacity or embedding efficiency (Section 5.3) of RRC steganography, we only explain an approximate 100% entropy utilization for it without rigorous theoretical proofs, but experiments can empirically prove that its utilization is approximate 100%.

## Ethical Considerations

While steganography has legitimate applications such as embedding copyright information, it can also be misused for disinformation or to evade censorship. This dual-use nature underscores the need for effective monitoring and regulation. We emphasize that future research should develop effective detection and monitoring systems in parallel, ensuring responsible use of steganography.

## Acknowledgments

We express our gratitude to the anonymous reviewers for their valuable and insightful comments. This work was supported by JST SPRING, Grant Number JPMJSP2110.

## References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Anthropic. 2024. The claude 3 model family: Opus, sonnet, haiku. https://www-cdn.anthropic.com/de8ba9b01c9ab7cbabf5c33b80b7bbc618857627/Model_Card_Claude_3.pdf.

Berk Atil, Sarp Aykent, Alexa Chittams, Lisheng Fu, Rebecca J. Passonneau, Evan Radcliffe, Guru Rajan Rajagopal, Adam Sloan, Tomasz Tudrej, Ferhan Ture, Zhe Wu, Lixinyu Xu, and Breck Baldwin. 2025. Non-determinism of "deterministic" llm settings. *Preprint*, arXiv:2408.04667.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, and 12 others. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Christian Cachin. 1998. An information-theoretic model for steganography. In *Information Hiding*, pages 306–318, Berlin, Heidelberg. Springer Berlin Heidelberg.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Unsupervised

cross-lingual representation learning at scale. *CoRR*, abs/1911.02116.

Falcon Dai and Zheng Cai. 2019. Towards near-imperceptible steganographic text. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4303–4308, Florence, Italy. Association for Computational Linguistics.

Christian Schroeder de Witt, Samuel Sokota, J Zico Kolter, Jakob Nicolaus Foerster, and Martin Strohmeier. 2023. Perfectly secure steganography using minimum entropy coupling. In *The Eleventh International Conference on Learning Representations*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Jinyang Ding, Kejiang Chen, Yaofei Wang, Na Zhao, Weiming Zhang, and Nenghai Yu. 2023. Discop: Provably secure steganography in practice based on "distribution copies". In *2023 IEEE Symposium on Security and Privacy (SP)*, pages 2238–2255.

A.A. Fedotov, P. Harremoes, and F. Topsoe. 2003. Refinements of pinsker's inequality. *IEEE Transactions on Information Theory*, 49(6):1491–1498.

Vojtěch Holub and Jessica Fridrich. 2013. Digital image steganography using universal distortion. In *Proceedings of the First ACM Workshop on Information Hiding and Multimedia Security*, IH&MMSec '13, page 59–68, New York, NY, USA. Association for Computing Machinery.

Nicholas Hopper, Luis von Ahn, and John Langford. 2009. Provably secure steganography. *IEEE Transactions on Computers*, 58(5):662–676.

Nicholas J. Hopper, John Langford, and Luis von Ahn. 2002. Provably secure steganography. In *Advances in Cryptology — CRYPTO 2002*, pages 77–92, Berlin, Heidelberg. Springer Berlin Heidelberg.

Gabriel Kaptchuk, Tushar M. Jois, Matthew Green, and Aviel D. Rubin. 2021. Meteor: Cryptographically secure steganography for realistic distributions. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, CCS '21, page 1529–1548, New York, NY, USA. Association for Computing Machinery.

Stefan Katzenbeisser and Fabien AP Petitcolas. 2002. Defining security in steganographic systems. In *Security and Watermarking of Multimedia Contents IV*, volume 4675, pages 50–56. SPIE.

Diederik P. Kingma and Jimmy Ba. 2017. Adam: A method for stochastic optimization. *Preprint*, arXiv:1412.6980.

Tri Van Le. 2003. Efficient provably secure public key steganography. Cryptology ePrint Archive, Paper 2003/156.

G Nigel N Martin. 1979. Range encoding: an algorithm for removing redundancy from a digitised message. In *Proc. Institution of Electronic and Radio Engineers International Conference on Video and Data Recording*, volume 2.

Jumon Nozaki and Yugo Murawaki. 2022. Addressing segmentation ambiguity in neural linguistic steganography. In *Proceedings of the 2nd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 12th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 109–116, Online only. Association for Computational Linguistics.

Chao Pan, Donghui Hu, Yaofei Wang, Kejiang Chen, Yinyin Peng, Xianjin Rong, Chen Gu, and Meng Li. 2025. Rethinking prefix-based steganography for enhanced security and efficiency. *IEEE Transactions on Information Forensics and Security*, 20:3287–3301.

Yuang Qi, Kejiang Chen, Kai Zeng, Weiming Zhang, and Nenghai Yu. 2025. Provably secure disambiguating neural linguistic steganography. *IEEE Transactions on Dependable and Secure Computing*, 22(3):2430–2442.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, and 1 others. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

J. Rissanen and G. G. Langdon. 1979. Arithmetic coding. *IBM Journal of Research and Development*, 23(2):149–162.

Jiaming Shen, Heng Ji, and Jiawei Han. 2020. Near-imperceptible neural linguistic steganography via self-adjusting arithmetic coding. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 303–313, Online. Association for Computational Linguistics.

Gustavus J. Simmons. 1984. *The Prisoners' Problem and the Subliminal Channel*, pages 51–67. Springer US, Boston, MA.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, and 1 others. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Yaofei Wang, Gang Pei, Kejiang Chen, Jinyang Ding, Chao Pan, Weilong Pang, Donghui Hu, and Weiming Zhang. 2025. SparSamp: Efficient provably secure

steganography based on sparse sampling. In *The 34th USENIX Security Symposium*.

Jiaxuan Wu, Zhengxian Wu, Yiming Xue, Juan Wen, and Wanli Peng. 2024. Generative text steganography with large language model. In *Proceedings of the 32nd ACM International Conference on Multimedia*, MM '24, page 10345–10353, New York, NY, USA. Association for Computing Machinery.

Lingyun Xiang, Shuanghui Yang, Yuhang Liu, Qian Li, and Chengzhang Zhu. 2020. Novel linguistic steganography based on character-level text generation. *Mathematics*, 8(9).

Ruiyi Yan, Chenhui Chu, Zhongliang Yang, and Yugo Murawaki. 2025. A comprehensive survey on linguistic steganography: Methods, countermeasures, evaluation, and challenges.

Ruiyi Yan and Yugo Murawaki. 2025. Addressing tokenization inconsistency in steganography and watermarking based on large language models. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 7076–7098, Suzhou, China. Association for Computational Linguistics.

Ruiyi Yan, Tian Song, and Yating Yang. 2024a. A near-imperceptible disambiguating approach via verification for generative linguistic steganography. In *2024 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 1638–1643.

Ruiyi Yan, Tian Song, and Yating Yang. 2024b. Tokenfree: A tokenization-free generative linguistic steganographic approach with enhanced imperceptibility. In *2024 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 449–455.

Ruiyi Yan, Yating Yang, and Tian Song. 2023. A secure and disambiguating approach for generative linguistic steganography. *IEEE Signal Processing Letters*, 30:1047–1051.

Kuan Yang, Kejiang Chen, Weiming Zhang, and Nenghai Yu. 2019a. Provably secure generative steganography based on autoregressive model. In *Digital Forensics and Watermarking*, pages 55–68, Cham. Springer International Publishing.

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

Zhong-Liang Yang, Si-Yu Zhang, Yu-Ting Hu, Zhi-Wen Hu, and Yong-Feng Huang. 2021. Vae-stega: Linguistic steganography based on variational auto-encoder. *IEEE Transactions on Information Forensics and Security*, 16:880–895.

Siyu Zhang, Zhongliang Yang, Jinshuai Yang, and Yongfeng Huang. 2021. Provably secure generative linguistic steganography. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3046–3055, Online. Association for Computational Linguistics.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. 2022. OPT: Open pre-trained transformer language models. *Preprint*, arXiv:2205.01068.

Zachary Ziegler, Yuntian Deng, and Alexander Rush. 2019. Neural linguistic steganography. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1210–1215, Hong Kong, China. Association for Computational Linguistics.

# A Related Work

In this section, we introduce the existing attempts to provably secure steganography, and analyze their characteristics or limitations.

## A.1 Arithmetic Coding (AC) & Meteor

Arithmetic coding (AC) is a form of entropy encoding used in lossless data compression (Rissanen and Langdon, 1979). A steganographic method that first adopts AC is proposed by Le (2003). Then, AC is applied in deep generative model and image generation to the field of provably secure steganography (Yang et al., 2019a). Following these works, in the field of linguistic steganography, researchers have presented a series of variant methods, especially including the original AC-based steganography (Ziegler et al., 2019), the AC-based method with a self-adjusting mechanism (Shen et al., 2020), and Meteor (Kaptchuk et al., 2021).

The sort of these AC-based steganographic methods commonly encounter a problem, that is, the precision limitation results in distortion in the original probability distribution at each generative step. Specifically, when the original probabilities are encoded into binary-based intervals, the selected probability for each token always has the form of $2^{-p^{(t)}}$, where $p^{(t)}$ is the precision at time $t$. It is almost impossible to maintain the original distribution perfectly, thus introducing distortion.

To mitigate this distortion, one method is using a higher initial precision, and even Python's

`decimal` module can be used here to support a precision that is higher than the precision of the tensor. However, during the iteration of AC-based steganography, the external interval is narrowed and expanded many times, and when the external interval is small, $p^{(t)}$ is also small. Therefore, as the precision is AC-based methods are changed all the times and cannot be controlled well, distortion on probability distribution is inevitable.

Besides, even though Meteor addresses some problems that basic AC-based steganography suffers, Meteor suffers from limited embedding capacity. The reason is that, as Meteor does not narrow the interval successively and only considers each generated symbol separately, thus it cannot fully utilize the entropy. And Meteor does not address the probability-distortion problem that arises in AC-based methods.

### A.2 Adaptive Dynamic Grouping (ADG)

Zhang et al. (2021) proposed a grouping-based steganographic method called adaptive dynamic grouping (ADG). At each time step, it dynamically groups the probability distribution of all tokens of the vocabulary into $2^r$ groups with approximately the same probability sum, and then numbers them $0, 1, \ldots, 2^r - 1$. All tokens in each group represent the same message bits of length $r$. Then, they match the first $r$ bits from the message to be embedded and converts them to a decimal number in $\{0, 1, \ldots, 2^r - 1\}$, and performs random sampling from the normalized distribution of its corresponding group to obtain the next token. In their assumptions, ADG can theoretically achieve perfect security (no probability distortion) if and only if the grouping is perfectly balanced.

However, the problem is that since the vocabulary-size probability distribution is discrete, the requirement is almost impossible to satisfy. In most cases, the actual distribution used to embed the message is a modified distribution, which is different from the original distribution.

### A.3 Iterative Minimum Entropy Coupling (iMEC)

de Witt et al. (2023) analyzed information-theoretic steganography through the lens of *minimum entropy coupling*. They investigated how much information about a fixed-length secret message can be inferred by the sender and receiver through the selection of tokens, aiming to maximize and accumulate this information until the entire message is determined. They demonstrated that achieving perfect steganographic security is equivalent to solving a coupling problem, and that maximizing transmission efficiency under perfect security corresponds to solving a minimum entropy coupling problem. Their proposed iMEC scheme fully exploits the theoretical properties of coupling and minimum entropy coupling. As a result, the method preserves the original probability distribution and achieves provably perfect security.

However, iMEC does have a certain bit error rate. In addition, to achieve minimum entropy coupling and enhance the embedding rate, a considerable amount of computational complexity, specifically $O(|\mathcal{V}| \log |\mathcal{V}|)$, is necessary to couple the probabilities. Low computation efficiency of iMEC makes it difficult to be practically utilized in a vocabulary-size situation.

### A.4 Distribution Copies (Discop)

Ding et al. (2023) proposed a provably secure steganographic method based on *distribution copies* (Discop). In this method, several distribution copies are generated by rotating all intervals by specific displacements. At each time step, the message determines which distribution copy to sample from. Discop also employs an iterative method based on the Huffman tree to further enhance the capacity. Experimental results demonstrated a high utilization rate of entropy. However, the complexity of creating a Huffman tree ($O(|\mathcal{V}|)$ complexity) at each step could not be efficient when a vocabulary-size candidate pool is encoded.

Discop works by creating several rotated *copies* of the model's probability distribution and picking one copy according to the secret bits. This design must let the receiver uniquely tell which copy is used, without ever changing the model's original probabilities. In practice, those rotated copies often overlap on the same token: if the random number falls into an overlap (a *disputed range*), the receiver cannot be sure which copy was chosen. When that happens, the embedding process has to step back and embed fewer bits at that position to keep extraction unambiguous, which directly reduces the per-step payload. These overlaps become more likely whenever one token is much more probable than the rest, because adding more copies pushes more of the token intervals on top of each other. As a result, the achievable rate is effectively governed by the dominance of the most probable token (not by the full entropy of the distribution), and over

long texts it only approaches a stricter ceiling rather than the ideal limit. The paper itself notes that, empirically, Discop reaches roughly 92%–95% of its stated theoretical limit because of these disputed ranges and the need to back off to smaller embeddings when they occur.

### A.5 Sparse Sampling (SparSamp)

Wang et al. (2025) proposed SparSamp, an efficient and provably secure steganographic method based on sparse sampling. SparSamp embeds messages by combining them with pseudo-random numbers to generate message-derived randomness for sampling. This approach introduces only $O(1)$ additional computational complexity per sampling step, ensuring high computational efficiency.

SparSamp embeds bits by turning the random numbers used for sampling into *message-driven* numbers. If two different message-driven numbers land on the same token, the receiver cannot tell which message was used. The paper calls this an inherent conflict: longer message chunks raise capacity but also raise the chance of such collisions; shorter chunks reduce collisions but waste headroom. To keep extraction unique without changing the distribution, SparSamp deliberately *spreads out* (sparsifies) the allowable random numbers (i.e., it increases the gap between neighboring positions), so different messages are unlikely to pick the same token. That spacing is the price of uniqueness and inevitably leaves a gap to 100% entropy utilization.

## B Propositions and Proofs

### B.1 Proof of Proposition 1

**Proposition 1.** $d_s^{(t)} \sim U(L^{(t-1)}, R^{(t-1)})$.

*Proof.* Considering Lines 9–10 in Algorithm 3, as $d_s^{(t)} = L^{(t-1)} + (d_s^{(t-1)} - L^{(t-1)} + o^{(t)} \times \Delta^{(t-1)}) \bmod \Delta^{(t-1)}$, and $o^{(t)} \in U(0,1)$, we let $A = d_s^{(t-1)} - L^{(t-1)}$ and $B = o^{(t)} \times \Delta^{(t-1)}$. Considering $X = (A+B) \bmod \Delta^{(t-1)}$, for any $x \in [0, \Delta^{(t-1)})$, there is:

$$P(X \le x) = P((A+B) \bmod \Delta^{(t-1)} \le x).$$

Let $A = k\Delta^{(t-1)} + r$, where $k \in \mathbb{Z}$ and $r \in [0, \Delta^{(t-1)})$. Considering periodicity of modulo operations, there is

$$(A+B) \bmod \Delta^{(t-1)} = (r+B) \bmod \Delta^{(t-1)}.$$

*Case 1:* $r + B \le \Delta^{(t-1)}$. There are $X = r + B$ and $P(B \le \Delta^{(t-1)} - r) = \frac{\Delta^{(t-1)} - r}{\Delta^{(t-1)}}$.

*Case 2:* $r + B > \Delta^{(t-1)}$. There are $X = r + B - \Delta^{(t-1)}$ and $P(B > \Delta^{(t-1)} - r) = \frac{r}{\Delta^{(t-1)}}$.

Therefore, for any $x \in [0, \Delta^{(t-1)})$, there is

$$P(X \le x) = \frac{x}{\Delta^{(t-1)}}$$

which means $X \sim U(0, \Delta^{(t-1)})$.

As $d_s^{(t)} = L^{(t-1)} + X$, $d_s^{(t)} \sim U(L^{(t-1)}, L^{(t-1)} + \Delta^{(t-1)})$, thus $d_s^{(t)} \sim U(L^{(t-1)}, R^{(t-1)})$. □

### B.2 Proof of Proposition 3

**Proposition 3.** $\Delta^{(t)} \le 1$ *is a sufficient condition for the embedding termination* $\frac{L^{(t)}+R^{(t)}}{2} - d_s^{(t)} \in (-0.5, 0.5]$ *(Lines 14–15 in Algorithm 3).*

*Proof.* If $\Delta^{(t)} = R^{(t)} - L^{(t)} \le 1$:

$$L^{(t)} \le d_s^{(t)} < R^{(t)} \le L^{(t)} + 1$$

$$\frac{L^{(t)} + R^{(t)}}{2} \in (L^{(t)}, L^{(t)} + 0.5] \subset (L^{(t)}, d_s^{(t)} + 0.5]$$

$$\frac{L^{(t)} + R^{(t)}}{2} \in [R^{(t)} - 0.5, R^{(t)}) \subset (d_s - 0.5, R^{(t)})$$

$$\frac{L^{(t)} + R^{(t)}}{2} \in (d_s - 0.5, d_s + 0.5]$$

$$\frac{L^{(t)} + R^{(t)}}{2} - d_s^{(t)} \in (-0.5, 0.5].$$

□

### B.3 Proof of Computational Security of RRC Steganography

**Proposition 4.** *For all probabilistic polynomial time (PPT) adversary detection* $\mathcal{A}_\mathcal{D}$*,*

$$|Pr[\mathcal{A}_\mathcal{D}(x_s) = 1] - Pr[\mathcal{A}_\mathcal{D}(x_c) = 1]| < negl(\lambda)$$

*where* $x_s$ *is the stegotext,* $x_c$ *is the normally generated cover text,* $\lambda$ *is the security parameter of the shared key* $K$ *(usually the length of* $K$*), and* $negl(\lambda)$ *is a negligible function concerning* $\lambda$*.*

*Proof.* We prove the computational indistinguishability (security) between stegotext and normal text using a **hybrid argument**. Define the following hybrid distributions:

- $\text{Hyb}_0$ – real steganographic generation: using the secret message $m_s$ and $K$.
- $\text{Hyb}_1$ – modified steganographic generation: initialized $d_s^{(-1)}$ as a uniform random value over $[0, 2^l)$ instead of $m_s$, with other steps unchanged.

- $\text{Hyb}_2$ – normal generation: sample tokens directly from language model $\mathcal{M}$ until the sequence length matches the expected stego length, without interval operations.

**Step 1: computational indistinguishability (security) between $\text{Hyb}_0$ and $\text{Hyb}_1$**

- In $\text{Hyb}_0$, $d_s^{(-1)} = \text{bin2dec}(m_s)$.
- In $\text{Hyb}_1$, $d_s^{(-1)} \sim \text{Uniform}(0, 2^l)$.

At $t = 0$, the rotation operation updates $d_s^{(0)}$:

$$d_s^{(0)} = (d_s^{(-1)} + o^{(0)} \times 2^l) \bmod 2^l.$$

Since $o^{(0)} \sim U(0,1)$ implies $o^{(0)} \times 2^l \sim \text{Uniform}(0, 2^l)$, we have:

- If $d_s^{(-1)}$ is fixed, $d_s^{(0)} \sim \text{Uniform}(0, 2^l)$.
- If $d_s^{(-1)}$ is uniform, $d_s^{(0)} \sim \text{Uniform}(0, 2^l)$ still holds.

For $t \geq 1$, $d_s^{(t)} \sim \text{Uniform}(L^{(t-1)}, R^{(t-1)})$ which is independent of initialization. Thus, the token sequence distributions of $\text{Hyb}_0$ and $\text{Hyb}_1$ are identical (statistically indistinguishable). For any PPT distinguisher $\mathcal{A}_\mathcal{D}$:

$$|\Pr[\mathcal{A}_\mathcal{D}(\text{Hyb}_0) = 1] - \Pr[\mathcal{A}_\mathcal{D}(\text{Hyb}_1) = 1]| = 0.$$

Initialization differences are eliminated by the first rotation step.

**Step 2: computational indistinguishability (security) between $\text{Hyb}_1$ and $\text{Hyb}_2$**

$\text{Hyb}_1$ uses random $d_s^{(-1)}$ and PRNG, but its conditional token distribution matches the normal generation. The stopping time $T$ is stochastic, but:

- Token distributions conditioned on history match normal generation.
- Termination depends on internal uniform variables $d_s^{(t)}$ and $o^{(t)}$, which are inaccessible to distinguishers observing only token sequences.

Consider $\text{Hyb}_2$ (normal generation): Sample tokens stepwise from $\mathcal{M}$ for length $L = \mathbb{E}[T]$ (polynomially bounded since $T = O(l)$, efficiently sampleable).

For any fixed sequence $S = [s^{(0)}, \ldots, s^{(k-1)}]$, its probability in $\text{Hyb}_1$ is: $\Pr_{\text{Hyb}_1}[S] = (\prod_{t=0}^{k-1} p^{(t)}(s^{(t)})) \times \Pr[\text{terminate at } k|S] \times \Pr[\text{no early termination}|S]$.

In $\text{Hyb}_2$ (fixed length $k$): $\Pr_{\text{Hyb}_2}[S] = \prod_{t=0}^{k-1} p^{(t)}(s^{(t)})$.

Though an extra factor $\Pr[\text{terminate}|S]$ exists:

- This factor depends on interval lengths determined by $S$, which is bounded in $[0, 1]$, and introduces no bias due to language model smoothness.
- In computational settings, distinguishers cannot efficiently compute $\Pr[\text{terminate}|S]$ (requires internal state or LM details), and the factor does not alter core conditional distributions.

Crucially, the cryptographically secure PRNG ensures its outputs $o^{(t)}$ are computationally indistinguishable from true randomness. If a valid PPT distinguisher $\mathcal{A}_\mathcal{D}$ exists for $\text{Hyb}_1$ and $\text{Hyb}_2$, we build an adversary $\mathcal{A}'$ to break PRNG security:

- $\mathcal{A}'$ simulates stego generation but uses a challenge randomness source (PRNG or true random) for $o^{(t)}$.
- if $\mathcal{A}_\mathcal{D}$ succeeds, $\mathcal{A}'$ breaks PRNG security, which leads to a contradiction against the cryptographically secure PRNG.

Thus:

$$|\Pr[\mathcal{A}_\mathcal{D}(\text{Hyb}_1) = 1] - \Pr[\mathcal{A}_\mathcal{D}(\text{Hyb}_2) = 1]| \leq \text{negl}_{\text{PRNG}}(\lambda).$$

**Step 3: computational indistinguishability (security) between $\text{Hyb}_0$ and $\text{Hyb}_2$**

By the hybrid argument:

$$\begin{aligned}&|\Pr[\mathcal{A}_\mathcal{D}(\text{Hyb}_0) = 1] - \Pr[\mathcal{A}_\mathcal{D}(\text{Hyb}_2) = 1]| \\ &\leq |\Pr[\mathcal{A}_\mathcal{D}(\text{Hyb}_0) = 1] - \Pr[\mathcal{A}_\mathcal{D}(\text{Hyb}_1) = 1]| \\ &+ |\Pr[\mathcal{A}_\mathcal{D}(\text{Hyb}_1) = 1] - \Pr[\mathcal{A}_\mathcal{D}(\text{Hyb}_2) = 1]| \\ &\leq 0 + \text{negl}_{\text{PRNG}}(\lambda) = \text{negl}_{\text{PRNG}}(\lambda).\end{aligned}$$

As $\text{Hyb}_0$ is real steganographic generation, $\text{Hyb}_2$ is normal generation (length-matched), for any PPT distinguisher $\mathcal{A}_\mathcal{D}$, there is:

$$|\Pr[\mathcal{A}_\mathcal{D}(x_s) = 1] - \Pr[\mathcal{A}_\mathcal{D}(x_c) = 1]| < \text{negl}(\lambda)$$

where $\text{negl}(\lambda) = \text{negl}_{\text{PRNG}}(\lambda)$, which is negligible.

□

| Method | Avg / Max KLD (bits/token) ↓ | Capacity (bits/token) ↑ | Entropy (bits/token) | Utilization (%) ↑ | Speed (bits/s) ↑ |
|---|---|---|---|---|---|
| Multinomial sampling | 0 / 0 | N/A | 4.59 | N/A | N/A |
| AC | 1.85E-03 / 1.13E-02 | 4.64 | 4.65 | 99.81 | 352.09 |
| ADG | 1.38E-04 / 1.61E-03 | 3.45 | 4.64 | 74.20 | 25.29 |
| Meteor w/o sort | 2.80E-02 / 8.34E-02 | 3.13 | 4.54 | 69.03 | 410.79 |
| Meteor w/ sort | 2.77E-02 / 8.12E-02 | 3.65 | 4.52 | 80.76 | 46.02 |
| iMEC | **0 / 0** | 3.24 | 4.61 | 70.24 | 19.78 |
| Discop w/o sort | **0 / 0** | 1.92 | 4.67 | 41.08 | 154.25 |
| Discop w/ sort | **0 / 0** | 4.39 | 4.63 | 94.71 | 31.94 |
| SparSamp | **0 / 0** | 4.35 | 4.53 | 96.08 | **852.36** |
| RRC steganography (ours) | **0 / 0** | **4.70** | 4.67 | **100.67** | 750.41 |

Table 4: Quantitative comparison with previous steganographic methods on OPT-1.3b.

| Method | Avg / Max KLD (bits/token) ↓ | Capacity (bits/token) ↑ | Entropy (bits/token) | Utilization (%) ↑ | Speed (bits/s) ↑ |
|---|---|---|---|---|---|
| Multinomial sampling | 0 / 0 | N/A | 3.46 | N/A | N/A |
| AC | 6.92E-04 / 9.90E-03 | 3.53 | 3.52 | 100.33 | 104.37 |
| ADG | 1.81E-04 / 3.90E-03 | 2.41 | 3.54 | 68.14 | 21.15 |
| Meteor w/o sort | 1.24E-02 / 4.00E-02 | 2.42 | 3.50 | 69.14 | 98.71 |
| Meteor w/ sort | 1.21E-02 / 4.19E-02 | 2.89 | 3.53 | 81.84 | 50.26 |
| iMEC | **0 / 0** | 2.48 | 3.43 | 72.35 | 10.30 |
| Discop w/o sort | **0 / 0** | 1.50 | 3.49 | 42.99 | 127.61 |
| Discop w/ sort | **0 / 0** | 3.33 | 3.48 | 95.72 | 26.13 |
| SparSamp | **0 / 0** | 3.38 | 3.44 | 98.12 | **326.12** |
| RRC steganography (ours) | **0 / 0** | **3.57** | 3.52 | **101.41** | 146.24 |

Table 5: Quantitative comparison with previous steganographic methods on Llama-2-7b.

# C Experimental Details & Supplementary Information

## C.1 Supplementary Main Results

Tables 4 and 5 report the average results across various metrics for OPT-1.3b and Llama-2-7b. The findings are consistent with those in Table 1 (GPT-2), where our RRC steganography achieves the highest embedding capacity and embedding utilization among all methods. In contrast, for the two larger language models, our method attains the second-fastest speed, behind SparSamp, which is particularly optimized for efficiency.

In addition, some utilization results exceeding 100% appear due to sampling bias from finite data. Similar instances of over-100% utilization have also been reported in recent work (Pan et al., 2025).

## C.2 Supplementary Results of Scalability

Tables 6 and 7 list the average utilization, speed, and running time when RRC steganography embeds the secret message with various lengths (up to 8192 bits) on OPT-1.3b and Llama-2-7b. Similarly to the results of 2, RRC steganography can achieve steady entropy utilization around 100%.

## C.3 Supplementary Results of Comparison

Table 8 reports steganalysis accuracies across three steganalysis models, three language models for steganography, and two steganographic methods (our RRC and the current state-of-the-art baseline, SparSamp). The experimental settings for steganalysis follow the details in Appendix C.4.

Table 9 presents the median perplexities under various language models for steganography and two steganographic methods (with multinomial sampling included for reference). The embedded secret message was a random 128-bit sequence, with 1,000 samples per group. Median values are reported because average perplexities are heavily skewed by extreme outliers.

Overall, both methods yield near-random steganalysis accuracies (about 50%) and comparable perplexity scores (close to those of multinomial sampling), due to the provable security property shared by RRC steganography and SparSamp. Be-

| Message length (bits) | 32 | 64 | 128 | 256 | 512 | 1024 | 2048 | 4096 | 8192 |
|---|---|---|---|---|---|---|---|---|---|
| Utilization (%) ↑ | 100.09 | 99.26 | **100.67** | 100.35 | 100.30 | 100.02 | 100.14 | 100.14 | 100.08 |
| Speed (bits/s) ↑ | 685.32 | 745.03 | **750.41** | 720.93 | 701.11 | 588.33 | 403.20 | 248.71 | 184.42 |
| Running time (s) | 0.047 | 0.086 | 0.171 | 0.355 | 0.730 | 1.740 | 5.079 | 16.469 | 44.421 |

Table 6: Average results on utilization, speed and running time of RRC steganography across various message lengths $l$ on OPT-1.3b.

| Message length (bits) | 32 | 64 | 128 | 256 | 512 | 1024 | 2048 | 4096 | 8192 |
|---|---|---|---|---|---|---|---|---|---|
| Utilization (%) ↑ | **102.01** | 100.29 | 101.41 | 100.52 | 100.25 | 100.31 | 100.35 | 100.17 | 100.04 |
| Speed (bits/s) ↑ | 142.67 | 144.63 | **146.24** | 142.00 | 126.51 | 92.65 | 79.38 | 63.21 | 46.10 |
| Running time (s) | 0.224 | 0.443 | 1.143 | 1.803 | 4.047 | 11.052 | 25.800 | 64.800 | 177.701 |

Table 7: Average results on utilization, speed and running time of RRC steganography across various message lengths $l$ on Llama-2-7b.

| Steganalysis model | GPT-2 | | OPT-1.3b | | Llama-2-7b | |
|---|---|---|---|---|---|---|
| | RRC (ours) | SparSamp | RRC (ours) | SparSamp | RRC (ours) | SparSamp |
| bert-base-uncased | 49.3% | 49.9% | 48.1% | 48.6% | 49.7% | 50.9% |
| roberta-base | 51.1% | 50.5% | 50.1% | 49.7% | 51.3% | 50.5% |
| roberta-large | 48.9% | 49.2% | 52.6% | 51.8% | 47.7% | 47.6% |

Table 8: Comparison of steganalysis accuracies against RRC steganography and SparSamp steganography under cases where models for steganography vary and models for steganalysis vary.

| Method | GPT-2 | OPT-1.3b | Llama-2-7b |
|---|---|---|---|
| Multinomial sampling | 81.05 | 74.09 | 74.99 |
| SparSamp | 83.56 | 73.18 | 73.37 |
| RRC steganography (Ours) | 82.79 | 72.66 | 75.24 |

Table 9: Comparison of median perplexity between RRC steganography and SparSamp steganography under cases where models for steganography vary.

| Method | Avg / Max KLD (bits/token) ↓ | Capacity (bits/token) ↑ | Entropy (bits/token) | Utilization (%) ↑ | Speed (bits/s) ↑ |
|---|---|---|---|---|---|
| Implemented in GPT-2 | | | | | |
| Vanilla RC steganography | 1.67E-14 / 5.73E-12 | 5.94 | 5.93 | 100.20 | 1605.31 |
| RRC steganography | 0 / 0 | 5.93 | 5.93 | 99.98 | 1554.66 |
| Implemented in OPT-1.3b | | | | | |
| Vanilla RC steganography | 1.46E-14 / 4.13E-12 | 4.70 | 4.69 | 100.19 | 770.49 |
| RRC steganography | 0 / 0 | 4.70 | 4.67 | 100.67 | 750.41 |
| Implemented in Llama-2-7b | | | | | |
| Vanilla RC steganography | 7.26E-13 / 1.99E-12 | 3.55 | 3.55 | 99.91 | 153.64 |
| RRC steganography | 0 / 0 | 3.57 | 3.52 | 101.41 | 146.24 |

Table 10: Comparison between vanilla RC steganography and RRC steganography in various metrics.

sides, we emphasize that lower perplexity does not imply higher imperceptibility, due to the conflict between perceptual and statistical imperceptibility (the Psic Effect) in steganography (Yang et al., 2021).

### C.4 Steganalysis

We generated 5,000 pairs of cover texts (via multinomial sampling) and steganographic texts, respectively implemented on GPT-2, OPT-1.3b, and Llama-2-7b. In each pair, the lengths (token num-

ber) of two texts are the same. The initial contexts for generation are the first 10 words from sequences randomly selected from the C4 dataset. For each experimental group, 5,000 texts are split in a 6:2:2 ratio to create the training, validation, and test sets.

For fine-tuning BERT or RoBERTa models, we use Adam (Kingma and Ba, 2017) as the optimizer with a learning rate of $5 \times 10^{-5}$. The batch size is set to 2048, and the discriminator is trained for 20 epochs, running time of the whole training process is approximately 5 minutes.

### C.5 Ablation on Rotation Mechanism

To quantify the effect of the rotation mechanism, we implemented a variant without rotation (*vanilla RC*). As shown in Table 10 (where the experimental setups follow Section 6.1), we can find that, in vanilla RC steganography, even though it can obtain the higher embedding speed compared to RRC steganography, 0 KL divergence cannot be achieved. Most importantly, theoretical security cannot be achieved. Therefore, the provable security of vanilla RC steganography cannot be ensured as shown in the analysis in Section 3.2.

## D Discussion on Practical Issues

One practical issue of linguistic steganography is **tokenization inconsistency**. The stegotext $t_s$ generated by $\mathcal{S}_{\text{emb}}(\mathcal{M}, m_s)$ is essentially a sequence of tokens. Before transmission, the sender must detokenize this sequence using a tokenizer to produce the final stegotext. Consequently, during extraction $\mathcal{S}_{\text{ext}}(\mathcal{M}, t_s)$, any tokenization inconsistency may cause extraction to fail or yield an incorrect secret message. This problem can be avoided only in a few tokenizer-free linguistic steganographic approaches (Xiang et al., 2020; Yan et al., 2024b). Recently, several disambiguation methods have been proposed to mitigate this issue (Nozaki and Murawaki, 2022; Yan et al., 2023, 2024a; Qi et al., 2025; Yan and Murawaki, 2025). These methods are orthogonal to our proposed RRC steganography and can be compatible, since they operate on candidate pools prior to steganographic processing.

Another practical issue of linguistic steganography is **hardware indeterminism** (Atil et al., 2025), whereby LLM outputs and the probability distribution of the next token given the same context can vary across different hardware settings. Such variability poses serious risks for reliable message extraction in real-world deployments, since even small shifts in probability values may result in incorrect extraction. This issue underscores the importance of explicitly addressing hardware-level robustness as an additional design dimension for practical steganographic systems, alongside imperceptibility and capacity, in future research.

## E Samples of Texts

We present examples of stegotexts generated by RRC steganography and non-steganographic texts generated by multinomial sampling. Each stegotext embeds a 128-bit random secret message. The initial context is "`Occasionally when I get some free time, I'll do.`" Following the approach of Ziegler et al. (Ziegler et al., 2019), we terminate the generation process once the proposed method has finished embedding the message. In the following examples, for each language model, the non-steganographic text has the same token number as that of the corresponding stegotext.

**Texts generated by GPT-2**

**Stegotext:**
Occasionally when I get some free time, I'll do something that uses all of those sensors scanned at the bottom of the computer - search for something. But I don't know how to do that

**Non-steganographic text:**
Occasionally when I get some free time, I'll do some stretch and the lights rise over me, and will do some real experimenting. I didn't really think about art where

**Texts generated by OPT-1.3b**

**Stegotext:**
Occasionally when I get some free time, I'll do flat screen, planner style arrangement cards. It really highlights that the cards are supposed to be focused

**Non-steganographic text:**
Occasionally when I get some free time, I'll do that in survival. It's loads more fun to just play around and search ways to build a nest and be creative in

Texts generated by Llama-2-7b

**Stegotext:**
Occasionally when I get some free time, I'll do a quick Google search on a random topic that interests me (if I have one free not sitting in front of a computer screen!), and just see where my curiosity takes me. The first thing

**Non-steganographic text:**
Occasionally when I get some free time, I'll do a Google Searche for ""badminton"". It is refreshing not to find the many images that can be found with another more popular global pastime. My experiment of searching finds pictures