USENIX

THE ADVANCED COMPUTING
SYSTEMS ASSOCIATION

Shimmer: a Provably Secure Steganography Based on Entropy Collecting Mechanism

Minhao Bai, Kaiyi Pang, Guorui Liao, Jinshuai Yang,
and Yongfeng Huang, *Tsinghua University*

https://www.usenix.org/conference/usenixsecurity25/presentation/bai-minhao

This paper is included in the Proceedings of the
34th USENIX Security Symposium.

August 13–15, 2025 • Seattle, WA, USA

978-1-939133-52-6

Open access to the Proceedings of the
34th USENIX Security Symposium is sponsored by USENIX.

Shimmer: a Provably Secure Steganography Based on Entropy Collecting Mechanism

Minhao Bai, Kaiyi Pang, Guorui Liao, Jinshuai Yang, Yongfeng Huang
Tsinghua University

Abstract

This paper introduces a practical and provably secure steganography scheme. Most of practical steganography methods alter the model distribution to conceal private messages, posing security risks. Prior theoretical approaches are grounded in unrealistic assumptions and are not efficiently applicable to real-world scenarios. The potential of practical and provably secure steganographic methods has not been fully realized. To address these challenges, we introduce a novel entropy collection mechanism designed to ensure security and increase capacity. This mechanism captures residual entropy from the previous embedding phase and utilizes it in subsequent embedding steps, thereby increasing embedding capacity. Leveraging this mechanism, we propose Shimmer, a practical and provably secure steganography method with high capacity. We apply an mitigation to minimize the probability of interval splitting events, which maximizes its capacity further. Our experimental results demonstrate that Shimmer achieves highest bitrate compared to other existing practical and provably secure steganography techniques.

1 Introduction

There is an increasing demand for private communication channels that allow individuals to evade overbearing surveillance and monitoring. Steganography offers a solution by embedding private messages within seemingly innocuous carriers such as images and texts [7, 8, 14, 22, 25, 27], providing an essential tool for those with urgent privacy needs.

Theoretical work on secure steganography [1, 4, 12, 21] was first introduced over two decades ago. Although these methods are not yet practical for widespread use, they establish a robust framework for secure universal steganography. These early studies focused on efficiently sampleable black-box channels and rejection sampling, introducing formal definitions of security from both complexity theory and information theory perspectives. Researchers aim to develop steganographic techniques that do not degrade the quality of the synthesized carriers (also known as stegotext) and remain indistinguishable from normal carriers (also known as covertext).

Unfortunately, most practical implementations [6, 17, 18, 29] do not adhere to the principles established by earlier theoretical work. Early practical approaches focused on generating stegotext that appears indistinguishable from normal covertext based on human perception. However, these methods are often easily detected by advanced adversarial analyzers [22, 24, 26, 27]. To date, practical steganography techniques with provable security guarantees remain scarce. The pioneering work in practical and provably secure steganography [14] introduces a paradigm based on generative models. This approach relies on the explicit distribution of carriers provided by the generative models. In their work, the researchers developed METEOR, a secure steganography scheme that dynamically adapts to changes in entropy. METEOR's security is grounded in pseudorandomization of the secret bits, similar to a one-time pad mechanism. Subsequently, [8] introduced DISCOP, which significantly enhances embedding capacity by constructing a balanced binary tree. DISCOP's security is based on sampling from multiple distribution copies. However, the capacity of METEOR is limited by the problem of extracting the shared prefix in the chosen interval. And DISCOP suffers from long execution times because the construction of Huffman trees is not so efficient in practice. In summary, these practical and provably secure steganography schemes struggle to achieve both high embedding capacity and time efficiency. In other words, the secret channel they construct does not provide good channel capacity.

This limitation stems from the double-edged sword-like design of the steganographic codec mechanism. Drawing lessons from the drawbacks of METEOR [14] and DISCOP [8], we construct a more effective mechanism to construct a novel steganography method with higher bitrate.

In this paper, we introduce Shimmer, a steganographic scheme designed for symmetric key settings. Specifically, Shimmer employs an entropy collection mechanism that stores information in the form of intervals, representing the

possible range of secret bits. The encoder converts the secret bit string into decimal form and uses a pseudorandom generator to sample from the model distribution, which ensures the security. To further enhance Shimmer's capacity, we have designed a method to minimize the probability of interval splitting events. The main contributions of this paper is summarized in the following:

- **Entropy collecting Mechanism.** We formally discuss the wastage of entropy in token-level embedding and propose a method that accumulates remained entropy after the previous embedding. This discovery can be applied to improve existing token-level embedding techniques.

- **Shimmer.** We introduce Shimmer, a novel symmetric-key steganographic system that provides provable security and higher bitrate. Our method achieves the highest capacity and efficiency among the practical and provably secure steganography schemes.

- **Implementation and Evaluation.** We implement Shimmer and evaluate its performance across different generative language models. Experimental results show that Shimmer provides the highest bitrate among practical and provably secure steganography schemes.

2 Related Work

2.1 Foundations: Theoretical Steganography Works

Theoretical steganography works [2, 4, 12, 20, 21] mostly base on efficiently sampleable channels and rejection sampling. Encoding process of these works often requires sampling from the channel multiple times and choose a sample as they want. The decoding process only computes the labels (often using hash functions) of the sampled output. Such encoding process require a strong assumption, that is, the entropy of the channel distribution should be larger than (or at least equal to) the number of queries made to the channel oracle. However, for the mostly used channels like english text, the entropy of the channel distribution may not always satisfy that assumption. Especially, for the current popular language models, their distribution has an extremely low entropy, which makes directly applying the theoretical works to the language models almost impossible.

The security definition is first proposed by [12], which requires indistinguishability between steganography output and normal channel output. We follow this definition to prove the security of the proposed steganography scheme.

2.2 Explorations: Practical but Insecure Works

With the development of language models, some researchers pay attention to hiding information in the model-generated text via steganography. Most of these works rely on an existing source coding algorithm such as Huffman Coding (HC) and Arithmetic Coding (AC). HC-based methods [17] often construct a Huffman tree for the tokens in the vocabulary of the language model and select the tokens whose codeword is the same as the prefix of the given secret bits. AC-based methods [6, 18, 29] perform the inverse transform sampling on the distributions predicted by the model. However, they use the secret bits to replace the random number sampled from the uniform distribution, which may cause a randomness reuse problem [14]. In a high-level overview, these works focus on using source coding methods to encode the probability of tokens and assign a codeword to each token. The token whose codeword matches the prefix of given secret bits is selected and output.

The shortcomings of these methods arise from the imbalance between the codeword probabilities and the token probabilities, since the probabilities of the tokens are not always negative integral powers of 2. Therefore, the probability of the codeword is usually not equal to the probability of the corresponding token, resulting in the output distribution not matching the channel distribution. And if the alphabet of tokens is large, the construction of the codebook is not efficient in practice.

2.3 New Paradigm: Practical and Provably Secure Works

METEOR [14] introduces a novel paradigm in generative model-based steganography, where both the sender and receiver share a key and the channel distribution at each step. It highlights that previous methods [26, 27, 29] suffer from the randomness reuse problem, making their steganographic outputs detectable. METEOR addresses this by encoding secret bits only when sufficient entropy is available and pseudo-randomizing the remaining bits to avoid reusing the same random number. However, METEOR's approach does not fully utilize the available entropy. After completing an embedding loop, any remaining entropy is wasted. This becomes particularly problematic in channels with extremely low entropy, frequently resulting in reduced embedding capacity. To enhance the expected embedding capacity, a reordering algorithm is proposed for METEOR. The complexity of this reordering algorithm scales linearly with the size of the model vocabulary size. By strategically reordering the distributions, this method increases capacity by 20 - 25%. This improvement helps mitigate the entropy wastage and boosts overall performance, especially in low-entropy channels.

DISCOP [8] follows the paradigm proposed by [14]. The

core process of DISCOP is to using different random numbers to sample multiple tokens from the multiple copies of model-predicted distribution. If the output tokens do not repeat, there is a one-to-one correspondence between secret bits and output tokens, which is feasible for correct encoding and decoding. Since copies of distribution may have lots of overlapping zones, the embedding rate is highly restricted. [8] proved that the embedding rate is asymptotic to the minimum entropy, computed in token level. In order to further improve the embedding capacity, they proposed a recursion version of DISCOP. The distribution is re-arranged by a binary tree that keeps the probability of nodes as balanced as possible, which is similar to a Huffman tree. More current works focus on robustness of steganography system [28], which follows the pseudorandom error-correction code [5]. IMEC views the steganographic codec problem as a minimum entropy coupling between the text distribution and the secret messages' distribution, resulting in imporving the capacity [7]. Other related papers consider realistic deploying challenges [3], and tokenizer disambiguation [16]. FDPSS [15] is the latest provably secure steganographic framework in the field of symmetric stegano graphy, while SparSamp [23] represents an initial attempt to improve time efficiency within the same symmetric steganographic setting. Beyond these works of generative text steganography, there are still other works focus on generative image steganography such as Pulsar [13].

3 Method

We propose a steganography scheme that (1) is provably secure, and harmless to the model's quality, (2) offers a high embedding rate, and (3) ensures always correct codec. Our method requires the sender and receiver to share secret key k previously, and operate on the same generative model and pseudorandom generator, which is a standard symmetric-key setting.

3.1 Prelimilaries

3.1.1 Notations

Key parameters. We always use λ to denote the secure parameter in this paper.

Function notations. We represent polynomial functions as $\mathsf{poly}(n)$, which are of the order equivalent to the function n^c for some constant $c > 0$. We denote the negligible function as $\mathsf{negl}(n)$, which is asymptotically smaller than the inverse of any polynomial function. We call a probability is *noticeable* if it is greater than $\mathsf{negl}(\lambda)$. And we call a probability is *overwhelming* if it is greater than $1 - \mathsf{negl}(\lambda)$. $\ln(\cdot)$ represents the natural logarithm function. $\mathbb{1}(\cdot)$ denotes the indicator function, which outputs 1/0 if its input is true/false. $\inf\{I\}$ and $\sup\{I\}$ represent the greatest lower bound and least upper bound of the set I.

String notations. For a string b, b_i denotes the i-th element of b, and $b_{i:j}$ denotes a substring starting from the i-th element and ending at the j-th element (inclusive). The concatenation of 2 strings a, b is written as $a||b$. A series of concatenation operations can be written as $||_{n=i}^{j} b_n$, which represents the substring $b_{i:j}$. We use the symbol $\emptyset$ to denote an empty string. The length of a string b is denoted by $|b|$. The notation 0^λ denotes a string consisted of λ 0s.

Distribution notations. We use Unif to denote the uniform distribution. For example, $\mathsf{Unif}[0,1]$ represents a random number uniformaly distributed between 0 and 1. The notation $x \xleftarrow{\$} \mathsf{Unif}[0,1]$ signifies that a value x is randomly sampled from the distribution $\mathsf{Unif}[0,1]$. We use Dist_V to describe a distribution defined on the set V. We use $\approx_c$ and $\approx_s$ to denote the computational indistinguishablility and statistical closeness between 2 different distributions.

3.1.2 Pseudorandom Generator

Definition 1 (Pseudorandom Generator)**.** *A deterministic algorithm* $\mathsf{PRG}_k(\cdot) \to \{0,1\}$ *is a pseudorandom generator if for all polynomial time adversaries* $\mathcal{A}$*,*

$$\left|\mathbf{Pr}\left[\mathcal{A}^{\mathsf{PRG}_k(\cdot)} = 1\right] - \mathbf{Pr}\left[\mathcal{A}^{\mathsf{Unif}\{0,1\}^{\mathsf{poly}(|k|)}} = 1\right]\right| < \mathsf{negl}(|k|). \tag{1}$$

Once the $\mathsf{PRG}_k(\cdot)$ generates a list of $\mathsf{poly}(|k|)$ random bits $\{b_1, b_2, \cdots\}$, they can be transformed into a real number in [0,1] using the formula $\sum_{i=1}^{\mathsf{poly}(|k|)} \frac{b_i}{2^i}$. In this paper, we always convert the output of $\mathsf{PRG}_k(\cdot)$ into these real numbers and write them as $\{r_1, r_2, \cdots\}$.

3.1.3 Generative Models

Different from the previous opinion that considers the generative models as a whole, we separate the deterministic part and the randomized part of the model as 2 distinct algorithms.

Definition 2 (Generative Model)**.** *A generative model is a pair of efficient algorithms* $\mathsf{Model} = (\mathsf{Predictor}, \mathsf{Sampler})$ *based on a vocabulary* V*.*

- $\mathsf{Predictor}(H) \to \mathsf{Dist}_\mathsf{V}(\cdot)$ *is a deterministic algorithm that takes as input a prompt* H*, and outputs a distribution of the next token* $\mathsf{Dist}_\mathsf{V}(\cdot)$ *over the vocabulary* V*.*
- $\mathsf{Sampler}(\mathsf{Dist}_\mathsf{V}(\cdot)) \to t$ *is a randomized algorithm that takes as input a distribution* $\mathsf{Dist}_\mathsf{V}(\cdot)$ *over the vocabulary* V*, and outputs a token* t*.*

3.1.4 Inverse Transform Sampling

Inverse transform sampling is a convenient way to sample from the distribution that Predictor outputs. For the generative language models, the output tokens are mapped

to ID numbers, which is called token IDs. Therefore, we can write the distribution in a more formal way. We let $\mathsf{Dist}_{\mathsf{V}}(t), t \in [|\mathsf{V}|]$ be the cumulative probability function of the distribution, where t is the token ID. Specifically, let $p(t)$ be the probability of outputting token t, $\mathsf{Dist}_{\mathsf{V}}(t) = \sum_{n=1}^{t} p(n)$. The inverse transform sampling is defined as follows:

Definition 3 (Inverse Transform Sampling)**.** *Given the cumulative probability distribution* $\mathsf{Dist}_{\mathsf{V}}(t), t \in \{0, 1, \cdots, |\mathsf{V}|\}$*, its inverse function is defined as*

$$\mathsf{Dist}_{\mathsf{V}}^{-1}(r) := \inf\{t \in \{0, 1, \cdots, |\mathsf{V}|\} : \mathsf{Dist}_{\mathsf{V}}(t) \geq r, 0 \leq r \leq 1\}. \tag{2}$$

Let r be a random number in $[0,1]$, the inverse transform sampling method outputs a token ID $\mathsf{Dist}_{\mathsf{V}}^{-1}(r)$.

In this paper, we write the inverse transform sampling as a function $\mathsf{Sample}(\mathsf{Dist}_{\mathsf{V}}(\cdot), r) \to t$, which takes as input a distribution $\mathsf{Dist}_{\mathsf{V}}(\cdot)$ and a random number r, and outputs a token t.

3.1.5 Conversion between bit strings and decimal numbers

We define 2 functions: Dec and Bin to convert between bit strings and decimal numbers in $[0,1]$.

- $\mathsf{Dec}(B) \to \sum_{n=1}^{|B|} \frac{B_n}{2^n}$ takes as input a bit string, and outputs a decimal number in $[0,1]$.
- $\mathsf{Bin}_{\beta}(d) \to (\lfloor d \cdot 2^{\beta} \rfloor)_2$ takes as input a decimal number in $[0,1]$, and outputs a bit string of length β, where $(\lfloor x \rfloor)_2$ denotes the binary representation of the integer part of x.

3.1.6 Merging the Intervals

In our construction, we sometimes need to merge two intervals into one. We define the merge function $\circ$ as follows:

$$I_1 \circ I_2 := \big\{x : \inf\{I_1\} + (\sup\{I_1\} - \inf\{I_1\}) \cdot \inf\{I_2\} \leq x \leq \inf\{I_1\} + (\sup\{I_1\} - \inf\{I_1\}) \cdot \sup\{I_2\}\big\} \tag{3}$$

3.1.7 Steganography Scheme

In most situations, we discuss a steganography scheme based on a generative model M. We formally give a definition of a steganography scheme as follows:

Definition 4 (Steganography Scheme)**.** *A steganographic scheme is a tuple of efficient algorithms* $SS = (\mathsf{KeyGen}, \mathsf{Encode}, \mathsf{Decode})$ *based on a model* Model*, where*

- $\mathsf{KeyGen}(1^{\lambda}) \to sk$ *is a randomized algorithm that takes as input a security parameter* λ*, outputs a key* sk*.*
- $\mathsf{Encode}_{sk,\mathsf{Model}}(H, B) \to T$ *is a keyed randomized algorithm that takes as input a prompt* H *and a secret bit string* B*, output a token string* T*.*
- $\mathsf{Decode}_{sk,\mathsf{Model}}(H, T) \to B$ *is a keyed randomized algorithm that takes as input a prompt* H *and a token string* T*, outputs a secret bit string* B*.*

Basically, a steganography scheme requires 2 properties: **security** and **correctness**. The security is defined as the computational indistinguishability between the steganography output and the model output. The correctness is defined as the probability of decoding failure is negligible in secure parameter λ. We write the formal definitions of them in the following:

Definition 5 (Security of a Steganography Scheme)**.** *Let* $\mathsf{Model}'(h, B) := \mathsf{Model}(h)$*. A steganography scheme* $SS = (\mathsf{KeyGen}, \mathsf{Encode}, \mathsf{Decode})$ *based on a model* Model *is secure if for all polynomial time adversaries* $\mathcal{A}$*,*

$$\left| \mathbf{Pr}\left[\mathcal{A}^{\mathsf{Encode}_{sk,\mathsf{Model}}(\cdot,\cdot)}(1^{\lambda}) = 1\right] - \mathbf{Pr}\left[\mathcal{A}^{\mathsf{Model}'(\cdot,\cdot)}(1^{\lambda}) = 1\right] \right| < \mathsf{negl}(\lambda). \tag{4}$$

Definition 6 (Correctness of a Steganography Scheme)**.** *A steganography scheme* $SS = (\mathsf{KeyGen}, \mathsf{Encode}, \mathsf{Decode})$ *based on a model* Model *is correct if for any* sk*,*

$$\mathbf{Pr}[\mathsf{Decode}_{sk,\mathsf{Model}}(h, \mathsf{Encode}_{sk,\mathsf{Model}}(h, B)) = B] \geq 1 - \mathsf{negl}(\lambda). \tag{5}$$

3.2 Threat Model

We adopt the framework of the Prisoners' Problem introduced in [19], which models a scenario where two prisoners, Alice and Bob, attempt to communicate covertly over an untrusted channel. They achieve this by embedding secret messages into innocuous cover objects (e.g., images or texts). In the standard setting, the adversary has full access to the channel and can observe all communications between Alice and Bob. If the adversary can distinguish between normal communications and steganographic communications with a probability greater than $\frac{1}{2}$, the steganography scheme is considered broken. Thus, a steganography scheme is computationally secure if no probabilistic polynomial-time adversary can break it.

In modern steganography, generative models are employed to approximate the channel's distribution. Here, the adversary has access to the same models used by the sender and receiver, as well as their communication history. A steganography scheme is computationally secure if no probabilistic polynomial-time adversary can distinguish between stegotext (generated by the steganography scheme) and covertext (generated by the model).

Since our steganography scheme relies on pseudorandom generators, we must assume the existence of such generators whose outputs are computationally indistinguishable from those of a true random generator.

3.3 Intuition

We mainly follow the METEOR [14] and some AC-based methods [6, 18]. The main difference between METEOR and AC-based methods is that METEOR uses XOR to pseudorandomize the secret bits without collecting entropy, while the AC-based methods directly use the secret bits as random source and apply an entropy collecting mechanism. Specifically, for a given secret bit string B, AC-based methods compute the decimal form of its prefix of length β, denoted as $\mathsf{Dec}(B_{1:\beta})$. Then they perform inverse transform sampling on the model-provided distribution $\mathsf{Dist}_{\mathsf{V}}(\cdot)$ using $\mathsf{Dec}(B_{1:\beta})$ as the random number. This sampling procedure finally outputs a token t, such that interval $[\mathsf{Dist}_{\mathsf{V}}(t-1), \mathsf{Dist}_{\mathsf{V}}(t)]$ contains the decimal $\mathsf{Dec}(B_{1:\beta})$. If the binary forms of the upper and lower bounds of this interval share a common prefix, all decimal numbers within this interval will have the same prefix. By extracting this shared prefix, the initial portion of B is recovered. The interval is then expanded and merged with the intervals of subsequently sampled tokens until the model naturally halts. The problem of these AC-based methods comes from using $\mathsf{Dec}(B_{1:\beta})$ as the random number, which may be repeated in multiple steps.

METEOR does not use $\mathsf{Dec}(B_{1:\beta})$ directly as the random number. Instead, it first generates a mask $m \in \mathsf{Unif}\{0,1\}^{\beta}$, and uses $\mathsf{Dec}(m \oplus B_{1:\beta})$ as the random number. However, this approach poses challenges when trying to recover the interval containing $\mathsf{Dec}(B_{1:\beta})$. Specifically, re-applying the XOR operation with the mask m on the binary forms of numbers within the chosen interval does not yield another interval. Instead, it breaks the interval into discrete numbers, making it extremely difficult to maintain the necessary structure. While XOR ensures security, it also reduces capacity because it prevents METEOR from collecting entropy in the same way AC-based methods do. To address this limitation, we explored other pseudorandomization techniques that could enable both security and entropy collection. We ultimately chose a method similar to XOR but operating in the real number field: modulo 1. For a fixed number $\mathsf{Dec}(B_{1:\beta}) \in [0,1]$ and a pseudorandom number $r \in [0,1]$, the expression $r + \mathsf{Dec}(B_{1:\beta}) \mod 1$ is still pseudorandom in $[0,1]$. If we finally sample a token t, We can still find the interval that contains $\mathsf{Dec}(B_{1:\beta})$ by computing $[\mathsf{Dist}_{\mathsf{V}}(t-1) - r \mod 1, \mathsf{Dist}_{\mathsf{V}}(t) - r \mod 1]$. The main advantage of using modulo 1 is that the reverse operation on an interval remains an interval, preserving the necessary structure for entropy collection and recovery.

3.4 Shimmer

We propose a steganography scheme named **Shimmer**. In this section, we will first introduce the encoding and decoding procedures. Following that, we will prove the security of the proposed scheme. We then discuss the interval splitting problem and present a simple mitigation strategy to minimize its negative impact.

3.4.1 Construction

Algorithm 1 $\mathsf{Sample}(\mathsf{Dist}_{\mathsf{V}}(\cdot), r) \to t$

Input: distribution $\mathsf{Dist}_{\mathsf{V}}(\cdot)$, pseudorandom number r.
Output: token t.

1: **return** $\mathsf{Dist}_{\mathsf{V}}^{-1}(r)$

Algorithm 2 $\mathsf{Extract}_{\beta}(I) \to B_{pre}, I'$

Input: bit precision β, interval I.
Output: shared prefix B_{pre}, updated interval I'

1: $B_{pre} \longleftarrow \emptyset$
2: $inf, sup \longleftarrow \mathsf{Bin}_{\beta}(\inf\{I\}), \mathsf{Bin}_{\beta}(\sup\{I\})$
3: $cnt \longleftarrow 0$
4: **for** $n \in \{1, 2, \cdots, \beta\}$ **do**
5: **if** $inf_n = sup_n$ **then**
6: $cnt \longleftarrow cnt + 1$
7: $B_{pre} \longleftarrow B_{pre} || inf_n$
8: **else**
9: **break**
10: **end if**
11: **end for**
12: $I' \longleftarrow \left[\mathsf{Dec}(inf_{cnt:\beta} || 0^{cnt}), \mathsf{Dec}(sup_{cnt:\beta} || 1^{cnt})\right]$
13: **return** B_{pre}, I'

Shimmer consists of three algorithms: $\mathsf{KeyGen}, \mathsf{Encode}$ and Decode. This scheme is parameterized by a security parameter λ, a bit precision β, and a generative model Model.

- $\mathsf{KeyGen}(1^{\lambda}) \to sk$ takes as input the security parameter λ, and outputs a secret key.
- $\mathsf{Encode}_{sk,\mathsf{Model},\beta}(H, B) \to T$ takes as input the prompt H and secret bit string B, and outputs a token string T.
- $\mathsf{Decode}_{sk,\mathsf{Model},\beta}(H, T) \to B$ takes as input the prompt H and token string T, and outputs a secret bit string B.

As shown in Fig.1, the Encode first initializes an interval $I = [0,1]$. For each generation step, It runs $\mathsf{PRG}_{sk}(i)$ to draw a random number r, where i represents the i-th token. The input i ensures that the PRG_{sk} does not repeat the same random number with a noticeable probability. At the beginning of generation, it runs the $\mathsf{Predictor}(H)$ to obtain the distribution $\mathsf{Dist}_{\mathsf{V}}(t)$, and computes the decimal form $\mathsf{Dec}(B_{1:\beta}) = \sum_{n=1}^{\beta} \frac{B_n}{2^n}$. Then it combines the decimal form $\mathsf{Dec}(B_{1:\beta})$ and the random number r to generate a new random number $r' = r + \mathsf{Dec}(B_{1:\beta}) \mod 1$. It performs inverse transform sampling by using r' as the random number, and output a token t, appended to the prompt $H = H||t$. As we know that $r' \in [\mathsf{Dist}_{\mathsf{V}}(t-1), \mathsf{Dist}_{\mathsf{V}}(t)]$, if $\mathsf{Dist}_{\mathsf{V}}(t-1) \geq r$ or

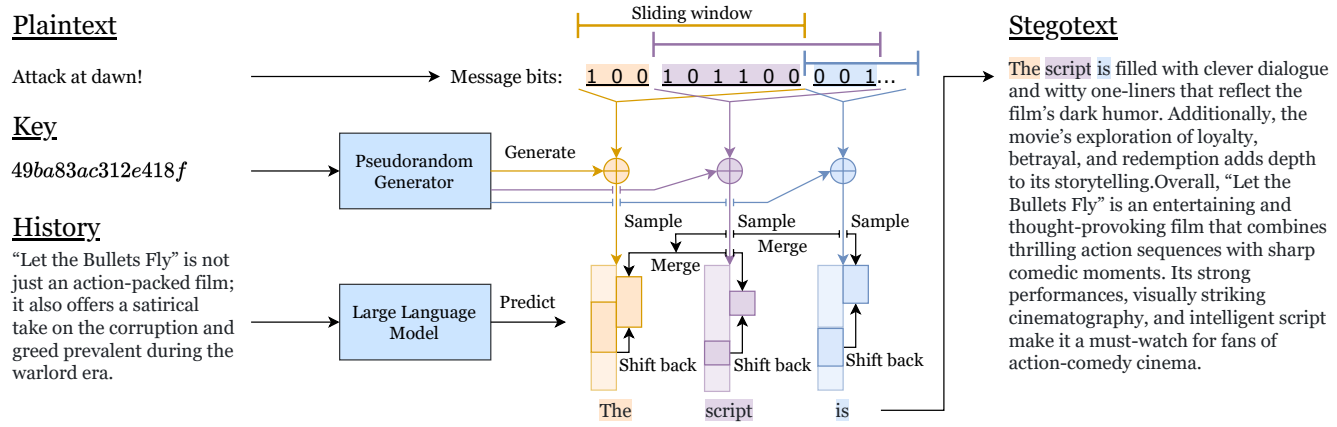


Figure 1: An overview of Shimmer. The sender and reciever should share the same key, history and the pseudorandom generator (PRG). The sampling procedure is controlled by the message bits and the number generated by PRG. During the embedding process, the probability intervals of each sampled token will be shifted backward by the the number generated by PRG and then merged into a single interval. Then we can extract the shared prefix from this interval to get the encoded bits.

$\mathsf{Dist}_V(t) \leq r$, it can compute $\mathsf{Dec}(B_{1:\beta}) \in [\mathsf{Dist}_V(t-1) - r \mod 1, \mathsf{Dist}_V(t) - r \mod 1]$; otherwise it just skips this computation and starts generating the next token. If the interval that contains $\mathsf{Dec}(B_{1:\beta})$ can be computed, it updates the interval $I = I \circ [\mathsf{Dist}_V(t-1) - r \mod 1, \mathsf{Dist}_V(t) - r \mod 1]$. Then it try to extract a shared prefix B_{pre} from the binary form of upper and lower bound of the interval I. If B_{pre} is not empty, it removes the prefix B_{pre} from the secret bit string. In the next steps of generation, it repeats the above process and updates the interval I until the Model naturally halts.

Similar to Encode, Decode first initializes an interval $I = [0,1]$, runs $\mathsf{PRG}_{sk}(i)$ to draw a random number r, and runs the $\mathsf{Predictor}(H)$ to obtain the distribution $\mathsf{Dist}_V(t)$. For each step, it can recover the interval $[\mathsf{Dist}_V(t-1), \mathsf{Dist}_V(t)]$. It also tries to compute $[\mathsf{Dist}_V(t-1) - r \mod 1, \mathsf{Dist}_V(t) - r \mod 1]$ and merge I with this interval to updates I. Then it extracts the shared prefix from the binary form of upper and lower bound of the interval I. It repeats the above process until the last token, and outputs the bits extracted from each step.

Details of $\mathsf{Encode}_{sk,\mathsf{Model},\beta}$ and $\mathsf{Decode}_{sk,\mathsf{Model},\beta}$ are shown in Algorithm 3 and 4. In the following part of the paper, we refer to the interval $[\mathsf{Dist}_V(t-1), \mathsf{Dist}_V(t)]$ as the *chosen token interval*, and the interval I as the *iterative interval*. The iterative interval I is the core of our entropy collecting mechanism.

3.4.2 A Running Example

As illustrated in Fig. 2, we need to embed the bits $B = 01010101...$ with its decimal representation being $\bar{B} = 0.3333....$ In the first step, $r_1 = 0.4$, and $r_1 + \bar{B} = 0.7333...$, which falls within the interval $[0.7, 1.0)$, representing the token "Hello". By shifting the interval $[0.7, 1.0)$ backward, we obtain the interval $[0.3, 0.6)$ that contains $\bar{B}$.

In the subsequent step, $r_2 = 0.7 \cdot (0.6 - 0.3) = 0.21$, and $r_2 + \bar{B} = 0.5433...$, which is within the interval $[0.51, 0.6)$ resulting from the composition of $[0.3, 0.6)$ and $[0.7, 1.0)$. Shifting the interval backward yields the interval $[0.3, 0.39)$ that contains $\bar{B}$. Here, we find that $f^{-1}(0.3) = 01001100...$ and $f^{-1}(0.39) = 01100011...$, both of which share the prefix 01. Consequently, we can expand the interval $[0.3, 0.39)$ to $[f(f^{-1}(0.3)[2:]), f(f^{-1}(0.39)[2:])) = [0.2, 0.56)$.

In the third step, $r_3 = 0.5 \cdot (0.56 - 0.2) = 0.18$, and $r_3 + \bar{B} = 0.5133...$, which falls within the interval $[0.452, 0.56)$ resulting from the composition of $[0.2, 0.56)$ and $[0.7, 1)$. Shifting the interval backward yields the interval $[0.272, 0.38)$ that contains $\bar{B}$. Here, we observe that $f^{-1}(0.272) = 01000101...$ and $f^{-1}(0.38) = 01100001...$, both sharing the prefix 01. Therefore, we can expand the interval $[0.272, 0.38)$ to $[f(f^{-1}(0.272)[2:]), f(f^{-1}(0.38)[2:])) = [0.088, 0.52)$.

Therefore, after three iterations, we have embedded the bits 0101 into the stegotext, leaving some additional information in the interval $[0.088, 0.52)$. The remaining information, $-\log_2(0.52 - 0.088) = 1.2109$ bits, will be utilized for the subsequent iteration. The decoding process is analogous, and we will not reiterate it here.

3.4.3 Security of Shimmer

To prove the security of our steganography scheme, we construct a series of games that naturally transition from steganographic schemes to normal generation process of models. We focus on the distribution of our method's output and the games are shown in Figure 3.4.1.

$G_1 \approx_c G_2$: G_1 uses a pseudorandom generator to perform

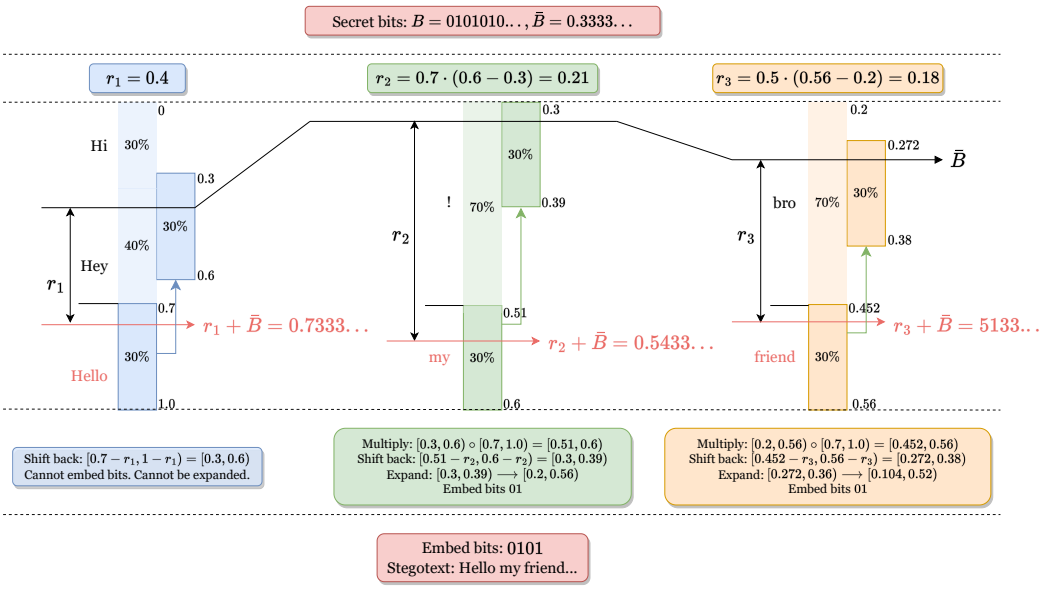


Figure 2: A simple example of encoding.

inverse transform sampling, while G_2 uses a real random generator. Assuming that there exists an polynomial time adversary $\mathcal{A}$, which has a noticeable probability of distinguishing between G_1 and G_2. We can construct another polynomial time adversary $\mathcal{A}'$ that can use $\mathcal{A}$ to distinguish between the pseudorandom generator PRG and the real random generator O.

$\mathcal{A}'$ works as follows:

- Adversary $\mathcal{A}'$ queries both a pseudorandom generator PRG and a real random generator O to obtain two random bit string, one from each source. Then $\mathcal{A}'$ computes the decimal form of them.
- $\mathcal{A}'$ runs G_1 using these two decimal numbers.
- $\mathcal{A}'$ receives a pair of outputs, one from G_1 (when using the pseudorandom generator) and one from G_2 (when using the real random generator).
- $\mathcal{A}'$ runs $\mathcal{A}$ on this pair of outputs to determine whether the output came from G_2 or G_3.

Since $\mathcal{A}$ can distinguish between G_1 and G_2 with non-negligible probability, $\mathcal{A}'$ can also distinguish between the pseudorandom numbers and real random numbers with non-negligible probability. However, this contradicts the definition of a secure pseudorandom generator, which states that no polynomial-time adversary can distinguish the output of a PRG from real random numbers with non-negligible probability. Therefore, our assumption that such an adversary $\mathcal{A}$ exists is false. Consequently, the output distributions of G_1 and G_2 are computationally indistinguishable.

$G_2 \approx_s G_3$: G_2 uses the real random generator to generate a bit string of length $\mathsf{poly}(\lambda)$, and then converts it into a decimal number. While G_3 directly draws a random decimal number from the interval $[0,1]$. The probability of G_2 outputs a token t is bounded by

$$\begin{aligned}&\mathbf{Pr}\left[r' \in [\mathsf{Dist}_V(t-1), \mathsf{Dist}_V(t)]\right] \\ \leq &\frac{\lceil (\mathsf{Dist}_V(t) - \mathsf{Dist}_V(t-1)) \cdot 2^{\mathsf{poly}(\lambda)} \rceil}{2^{\mathsf{poly}(\lambda)}},\end{aligned} \tag{6}$$

and

$$\begin{aligned}&\mathbf{Pr}\left[r' \in [\mathsf{Dist}_V(t-1), \mathsf{Dist}_V(t)]\right] \\ \geq &\frac{\lfloor (\mathsf{Dist}_V(t) - \mathsf{Dist}_V(t-1)) \cdot 2^{\mathsf{poly}(\lambda)} \rfloor}{2^{\mathsf{poly}(\lambda)}}.\end{aligned} \tag{7}$$

Therefore,

$$\begin{aligned}&\left|\mathbf{Pr}\left[r' \in [\mathsf{Dist}_V(t-1), \mathsf{Dist}_V(t)]\right] - (\mathsf{Dist}_V(t) - \mathsf{Dist}_V(t-1))\right| \\ \leq &\frac{1}{2^{\mathsf{poly}(\lambda)}},\end{aligned} \tag{8}$$

which is negligible in λ. Obviously, the probability that G_3 outputs the token t is $(\mathsf{Dist}_V(t) - \mathsf{Dist}_V(t-1))$. In this way, the total variance distance between output distributions of G_2 and G_3 is $\frac{1}{2}\sum_{n=1}^{|V|} \frac{1}{2^{\mathsf{poly}(\lambda)}}$, which is negligible in λ. Therefore, G_2 and G_3 are statistically close.

$G_3 = G_4$: G_3 executes an additional step, which adds the decimal form of secret bits and then modulo 1. Obviously, a fixed number $\frac{\mathsf{Dec}(B_{1:\beta}) - \inf\{I\}}{\sup\{I\} - \inf\{I\}}$ adds a random number $r \xleftarrow{\$} \mathsf{Unif}[0,1]$ is still uniformly random in $[0,1]$. The probability

G_1: Shimmer

1: $\mathsf{Dist_V}(\cdot) \longleftarrow \mathsf{Predictor}(H)$
2: $r \longleftarrow \mathsf{PRG}_{sk}(\cdot)$
3: $r' \longleftarrow r + \mathsf{Dec}(B_{1:\beta}) \mod 1$
4: $t \longleftarrow \mathsf{Sample}(\mathsf{Dist_V}(\cdot), r')$

G_2: Using real randomness

1: $\mathsf{Dist_V}(\cdot) \longleftarrow \mathsf{Predictor}(H)$
2: $r \xleftarrow{\$} \mathsf{Dec}\left(\mathsf{Unif}\{0,1\}^{\mathsf{poly}(\lambda)}\right)$
3: $r' \longleftarrow r + \mathsf{Dec}(B_{1:\beta}) \mod 1$
4: $t \longleftarrow \mathsf{Sample}(\mathsf{Dist_V}(\cdot), r')$

G_3: Using random real numbers

1: $\mathsf{Dist_V}(\cdot) \longleftarrow \mathsf{Predictor}(H)$
2: $r \xleftarrow{\$} \mathsf{Unif}[0,1]$
3: $r' \longleftarrow r + \mathsf{Dec}(B_{1:\beta}) \mod 1$
4: $t \longleftarrow \mathsf{Sample}(\mathsf{Dist_V}(\cdot), r')$

G_4: Removing the secret bits

1: $\mathsf{Dist_V}(\cdot) \longleftarrow \mathsf{Predictor}(H)$
2: $r \xleftarrow{\$} \mathsf{Unif}[0,1]$
3: $t \longleftarrow \mathsf{Sample}(\mathsf{Dist_V}(\cdot), r)$

G_5: Model

1: $\mathsf{Dist_V}(\cdot) \longleftarrow \mathsf{Predictor}(H)$
2: $t \longleftarrow \mathsf{Sample}(\mathsf{Dist_V}(\cdot))$

Figure 3: Games used in the security proof.

Algorithm 3 $\mathsf{Encode}_{sk,\mathsf{Model},\beta}(H,B) \to T$

Input: key sk, model Model, bit precision β, prompt H, secret bit string B.
Output: token string T.

$I \longleftarrow [0,1]$
$i \longleftarrow 0$
$T \longleftarrow \emptyset$
while Model does not halt **do**
  $\mathsf{Dist_V}(\cdot) \longleftarrow \mathsf{Predictor}(H)$
  $r \longleftarrow \mathsf{PRG}_{sk}(i)$
  $r' \longleftarrow r + \frac{\mathsf{Dec}(B_{1:\beta}) - \inf\{I\}}{\sup\{I\} - \inf\{I\}} \mod 1$
  $t \longleftarrow \mathsf{Sample}(\mathsf{Dist_V}(\cdot), r')$
  $T, H \longleftarrow T||t, H||t$
  if $r \le \mathsf{Dist_V}(t-1)$ **or** $r \ge \mathsf{Dist_V}(t)$ **then**
    $I \longleftarrow I \circ [\mathsf{Dist_V}(t-1) - r \mod 1, \mathsf{Dist_V}(t) - r \mod 1]$
  end if
  $B_{pre}, I \longleftarrow \mathsf{Extract}_{\beta}(I)$
  $B \longleftarrow B_{|B_{pre}|:|B|}$
  $i \longleftarrow i+1$
end while
return T

that G_3 output a token t is

$$\begin{aligned}
&\mathbf{Pr}\left[r + \frac{\mathsf{Dec}(B_{1:\beta}) - \inf\{I\}}{\sup\{I\} - \inf\{I\}} \mod 1 \in [\mathsf{Dist_V}(t-1), \mathsf{Dist_V}(t)]\right] \\
=&\mathsf{Dist_V}(t) - \mathsf{Dist_V}(t-1) \\
=&\mathbf{Pr}\left[r \in [\mathsf{Dist_V}(t-1), \mathsf{Dist_V}(t)]\right], \qquad (9)
\end{aligned}$$

which is identical to the probability that G_4 outputs the token t.

$G_4 = G_5$: G_4 uses the inverse transform sampling method to sample from the distribution, while G_5 is the original Sampler of the Model. We want to prove that the output of G_4 obeys the distribution given by Predictor. It is obvious that the probability of G_4 outputs a token t is $\mathsf{Dist_V}(t) - \mathsf{Dist_V}(t-1)$. As $\mathsf{Dist_V}(\cdot)$ is the cumulative probability function, $\mathsf{Dist_V}(t) - \mathsf{Dist_V}(t-1)$ is identical to the probability of Model outputs t. Therefore, the output distributions of G_4 and G_5 are identical.

We have proved that $G_1 \approx_c G_2 \approx_s G_3 = G_4 = G_5$. The output distribution of Shimmer and Model in each step is computationally indistinguishable. Since we set different input for the PRG, the output of PRG is exactly pseudorandom. In the end, our construction, Shimmer, is provably secure.

3.4.4 Correctness of Shimmer

As Algorithm 3 shows, the encoding process pre-decodes the secret bits, and the decoding process can completely recover the parameters used in encoding. Therefore, the proposed steganography scheme is always correct.

Algorithm 4 $\mathsf{Decode}_{sk,\mathsf{Model},\beta}(H,T)\to B$

Input: key sk, model Model, bit precision β, prompt H, token string T.
Output: secret bit string B.

$I \longleftarrow [0,1]$
$i \longleftarrow 0$
$B \longleftarrow \emptyset$
while Model does not halt **do**
  $\mathsf{Dist}_{\mathsf{V}}(\cdot) \longleftarrow \mathsf{Predictor}(H||T_{1:i})$
  $r \longleftarrow \mathsf{PRG}_{sk}(i)$
  if $r \le \mathsf{Dist}_{\mathsf{V}}(t-1)$ **or** $r \ge \mathsf{Dist}_{\mathsf{V}}(t)$ **then**
    $I \longleftarrow I \circ [\mathsf{Dist}_{\mathsf{V}}(t-1) - r \mod 1, \mathsf{Dist}_{\mathsf{V}}(t) - r \mod 1]$
  end if
  $B_{pre}, I \longleftarrow \mathsf{Extract}_{\beta}(I)$
  $B \longleftarrow B||B_{pre}$
  $i \longleftarrow i+1$
end while
return B

3.4.5 Capacity of Shimmer

In this section, we first prove that without interval splitting event, our steganography scheme does not lose information. Then we take this event into account and compute a more precise bound of utilized information. As the last iterative interval cannot be extracted for any secret bit, we subtract the remaining amount of information as well.

We define the information of a token string or an interval as follows:

Definition 7 (Information of a Token String)**.** *Let T be a token string and $p(T_i)$ be the probability of outputting token T_i in the i-th step, the information of the token string T is given by*

$$\mathsf{Info}(T) = \sum_{i=1}^{|T|} -\log_2(p(T_i)). \tag{10}$$

Definition 8 (Information of an Interval)**.** *Let I be a interval in $[0,1]$. The information of interval I is given by*

$$\mathsf{Info}(I) = -\log_2(\sup\{I\} - \inf\{I\}). \tag{11}$$

We first claim that if the chosen token interval does not split, the information is completely absorbed by the updated interval. We denote the interval I as the interval before merging, the interval I' as the interval after merging, and the token t as the token outputted in this step.

Claim 1. *If the event of interval splitting does not happen, we have*

$$\mathsf{Info}(t) = \mathsf{Info}(I') - \mathsf{Info}(I). \tag{12}$$

Proof. According to the workflow of Encode, we know that $I' = I \circ [\mathsf{Dist}_{\mathsf{V}}(t-1), \mathsf{Dist}_{\mathsf{V}}(t)]$. The length of interval I' is

$$\begin{aligned}\sup\{I'\} - \inf\{I'\} =& \inf\{I\} + (\sup\{I\} - \inf\{I\}) \cdot \mathsf{Dist}_{\mathsf{V}}(t) \\ & - (\inf\{I\} + (\sup\{I\} - \inf\{I\}) \cdot \mathsf{Dist}_{\mathsf{V}}(t-1)) \\ =& (\sup\{I\} - \inf\{I\}) \cdot p(t).\end{aligned} \tag{13}$$

Therefore,

$$\mathsf{Info}(I') = -\log_2((\sup\{I\} - \inf\{I\}) \cdot p(t)) = \mathsf{Info}(I) + \mathsf{Info}(t). \tag{14}$$

It proves that if the interval splitting does not happen, the information is completely utilized by the interval merging. □

Next, we claim that the algorithm Extract only waste negligible information.

Claim 2. *Let B_{pre} and I' be the output of $\mathsf{Extract}_{\beta}(I)$, we have*

$$|B_{pre}| + \mathsf{Info}(I') \ge \mathsf{Info}(I) - \mathsf{negl}(\beta). \tag{15}$$

Proof. Let $inf = \mathsf{Bin}_{\beta}(\inf\{I\})$ and $sup = \mathsf{Bin}_{\beta}(\sup\{I\})$. So we have $\inf\{I\} - \mathsf{Dec}(inf) \le \frac{1}{2^{\beta}}$ and $\sup\{I\} - \mathsf{Dec}(sup) \le \frac{1}{2^{\beta}}$. Combining these 2 inequality we have

$$\begin{aligned}\mathsf{Dec}(sup) - \mathsf{Dec}(inf) \in \big[&\sup\{I\} - \inf\{I\} - \frac{1}{2^{\beta}}, \\ &\sup\{I\} - \inf\{I\} + \frac{1}{2^{\beta}}\big].\end{aligned} \tag{16}$$

The information of I' is computed as

$$\begin{aligned}&\mathsf{Info}(I') \\ =& -\log_2\left(\mathsf{Dec}(sup_{|B_{pre}|:\beta}||0^{|B_{pre}|}) - \mathsf{Dec}(inf_{cnt:\beta}||1^{|B_{pre}|})\right) \\ =& -\log_2\left(\left(\mathsf{Dec}(sup) - \mathsf{Dec}(inf) + \frac{1}{2^{\beta}} - \frac{1}{2^{\beta+|B_{pre}|}}\right) \cdot 2^{|B_{pre}|}\right) \\ &\begin{cases} \ge -\log_2\left(\sup\{I\} - \inf\{I\} + \frac{2}{2^{\beta}} - \frac{1}{2^{\beta+|B_{pre}|}}\right) - |B_{pre}| \\ \le -\log_2\left(\sup\{I\} - \inf\{I\} - \frac{1}{2^{\beta+|B_{pre}|}}\right) - |B_{pre}| \end{cases}\end{aligned} \tag{17}$$

Therefore, $\mathsf{Info}(I') + |B_{pre}| \ge \mathsf{Info}(I) - \mathsf{negl}(\beta)$. If we set β to a large number, the loss of information is negligible. □

Then we discuss the influence of interval splitting event on the information utilization of Shimmer without reordering. Assume the Encode outputs token t in this step, the interval splitting event happens when $r \in [\mathsf{Dist}_{\mathsf{V}}(t-1), \mathsf{Dist}_{\mathsf{V}}(t)]$. At the same time, outputting the token t requires that $r + \mathsf{Dec}(B_{1:\beta}) \mod 1 \in [\mathsf{Dist}_{\mathsf{V}}(t-1), \mathsf{Dist}_{\mathsf{V}}(t)]$. So if the interval splitting event happens, r and $r + \mathsf{Dec}(B_{1:\beta}) \mod 1$ should be in the same interval $[\mathsf{Dist}_{\mathsf{V}}(t-1), \mathsf{Dist}_{\mathsf{V}}(t)]$. In common settings, the secret bit string B can be pseudorandomized or encrypted

before transmitting, and the value of $\mathsf{Dec}(B_{1:\beta})$ is independent from the distribution and the pseudorandom number r. Therefore, the probability of this event is

$$\begin{aligned}
&\mathbf{Pr}\left[r+\frac{\mathsf{Dec}(B_{1:\beta})-\inf\{I\}}{\sup\{I\}-\inf\{I\}} \bmod 1 \in [\mathsf{Dist}_{\mathsf{V}}(t-1),\mathsf{Dist}_{\mathsf{V}}(t)]\right]\\
=&\mathbf{Pr}\left[\frac{\mathsf{Dec}(B_{1:\beta})-\inf\{I\}}{\sup\{I\}-\inf\{I\}} \le \mathsf{Dist}_{\mathsf{V}}(t)-r\right]\\
&+\mathbf{Pr}\left[\frac{\mathsf{Dec}(B_{1:\beta})-\inf\{I\}}{\sup\{I\}-\inf\{I\}} \ge 1-(r-\mathsf{Dist}_{\mathsf{V}}(t-1))\right]\\
\le&\frac{\lceil[(\sup\{I\}-\inf\{I\})\cdot(\mathsf{Dist}_{\mathsf{V}}(t)-r)+\inf\{I\}]\cdot 2^{\beta}\rceil}{2^{\beta}}\\
&+\frac{\lceil 1-[(\sup\{I\}-\inf\{I\})\cdot(1-r+\mathsf{Dist}_{\mathsf{V}}(t-1))-\inf\{I\}]\cdot 2^{\beta}\rceil}{2^{\beta}}\\
\le&1+(\sup\{I\}-\inf\{I\})\cdot(\mathsf{Dist}_{\mathsf{V}}(t)-\mathsf{Dist}_{\mathsf{V}}(t-1)-1)+\frac{2}{2^{\beta}}\\
\le&p(t)+\frac{2}{2^{\beta}} \qquad (18)
\end{aligned}$$

As $\frac{2}{2^{\beta}}$ is negligible in β and β is sufficiently large, this term can be ignored. Next we computes the expectation of the information that can be utilized by our steganography scheme.

Lemma 1. *$x^2\log_2(x) > \frac{1}{3}x\log_2(x)-\frac{1}{5}x$.*

Claim 3. *The expectation of utilizable information of our steganography scheme $SS=(\mathsf{KeyGen},\mathsf{Encode},\mathsf{Decode})$ is at least $\frac{2}{3}\mathbb{E}_{t\in\mathsf{V}}[\mathsf{Info}(t)]-\frac{1}{5}$.*

Proof. In a single encoding step, the expectation of utilizable information is

$$\begin{aligned}
\mathbb{E}[R] =&\sum_{t=1}^{|\mathsf{V}|} -p(t)\cdot(1-p(t))\cdot\log_2(p(t))\\
=&\sum_{t=1}^{|\mathsf{V}|} -p(t)\log_2(p(t))+[p(t)]^2\log_2(p(t))\\
>&\sum_{t=1}^{|\mathsf{V}|} -p(t)\log_2(p(t))+\left(\frac{1}{3}p(t)\log_2(p(t))-\frac{p}{5}\right)\\
=&\frac{2}{3}\mathbb{E}_{t\in\mathsf{V}}[\mathsf{Info}(t)]-\frac{1}{5}. \qquad (19)
\end{aligned}$$

□

The expectation $\mathbb{E}_{t\in\mathsf{V}}[\mathsf{Info}(t)]$ is often referred to as the entropy of the distribution in this step. This result indicates that the utilized information grows proportionally to the entropy, which is asymptotically optimal.

However, the utilized information is close to but does not equal to the capacity, which represents the expectation of extractable secret bits, because the information of the last iterative interval I is wasted. Then we analyze this information loss.

From the algorithm Extract we observe that if the input interval I contains $\frac{1}{2}$, the output secret bits B_{pre} will be an empty string. Because the $\inf\{I\}$ is less than $\frac{1}{2}$ and has a binary form starting with 0 while the $\sup\{I\}$ is greater than $\frac{1}{2}$ and has a binary form starting with 1, no shared prefix can be extracted from them. If the input interval I does not contain $\frac{1}{2}$, it will be expanded. The updated interval's upper and lower bound do not share any prefix. Therefore, the final iterative interval I must contains $\frac{1}{2}$. Theoretically, the minimum iterative interval I is $[\frac{1}{2}-\frac{1}{2^{\beta}},\frac{1}{2}]$, which contains β bits information. If the next chosen token interval is not $[0,1]$, the merged interval should have some shared prefix to extract. Therefore, the capacity of a token string T should be larger than $\frac{2}{3}\mathbb{E}[\mathsf{Info}(T)]-\frac{1}{5}|T|-\beta$.

In summary, we prove that if the interval splitting event does not happen, the whole process does not waste noticeable information. Taking the interval splitting event into account, the utilized information is $\frac{2}{3}\mathbb{E}_{t\in\mathsf{V}}[\mathsf{Info}(t)]-\frac{1}{5}$. Taking the final unextractable iterative interval into account, the capacity of our steganography scheme is at least $\frac{2}{3}\mathbb{E}[\mathsf{Info}(T)]-\frac{1}{5}|T|-\beta$.

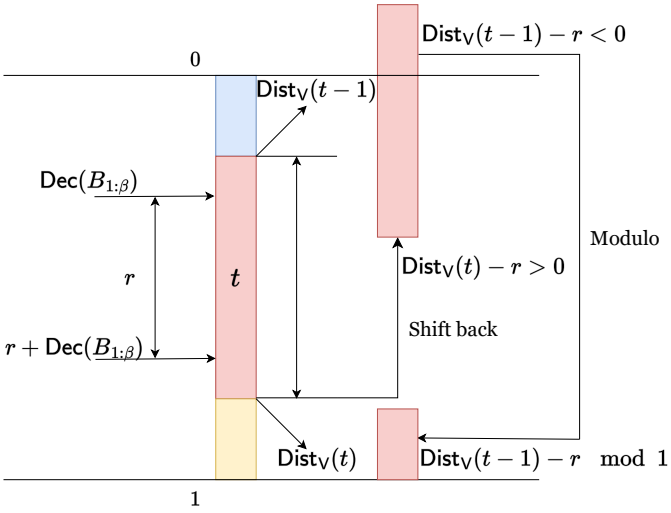


Figure 4: An example of interval splitting.

3.4.6 Mitigating the Interval Splitting Problem

As the Fig.4 shown, the interval splitting event happens when $r\in[\mathsf{Dist}_{\mathsf{V}}(t-1),\mathsf{Dist}_{\mathsf{V}}(t)]$ and

$$\frac{\mathsf{Dec}(B_{1:\beta})-\inf\{I\}}{\sup\{I\}-\inf\{I\}} \le \mathsf{Dist}_{\mathsf{V}}(t)-r, \qquad (20)$$

or

$$\frac{\mathsf{Dec}(B_{1:\beta})-\inf\{I\}}{\sup\{I\}-\inf\{I\}} \ge 1-(r-\mathsf{Dist}_{\mathsf{V}}(t-1)). \qquad (21)$$

Though the intervals that contains $\frac{\mathsf{Dec}(B_{1:\beta})-\inf\{I\}}{\sup\{I\}-\inf\{I\}}$ splits to 2 parts, the information they have is still equivalent to the information the token t has. Actually, the interval splitting

Algorithm 5 $\mathsf{MEncode}_{sk,\mathsf{Model},\beta}(H,B) \to T$

Input: key sk, model Model, bit precision β, prompt H, secret bit string B.
Output: token string T.

$I \longleftarrow [0,1]$
$i \longleftarrow 0$
$T \longleftarrow \emptyset$
$E \longleftarrow \emptyset$
while Model does not halt **do**
  $\mathsf{Dist_V}(\cdot) \longleftarrow \mathsf{Predictor}(H)$
  $r \longleftarrow \mathsf{PRG}_{sk}(i)$
  $r' \longleftarrow r + \frac{\mathrm{Dec}(B_{1:\beta}) - \inf\{I\}}{\sup\{I\} - \inf\{I\}} \mod 1$
  $t \longleftarrow \mathsf{Sample}(\mathsf{Dist_V}(\cdot), r')$
  $T, H \longleftarrow T||t, H||t$
  if $r \le \mathsf{Dist_V}(t-1)$ **or** $r \ge \mathsf{Dist_V}(t)$ **then**
    $I \longleftarrow I \circ [\mathsf{Dist_V}(t-1) - r \mod 1, \mathsf{Dist_V}(t) - r \mod 1]$
  else
    $E \longleftarrow [\mathsf{Dist_V}(t) - r, \mathsf{Dist_V}(t-1) - r + 1]$
    if $I \cap E \neq \emptyset$ **and** $E \not\subset I$ **then**
      $I \longleftarrow I \setminus I \cap E$
    end if
  end if
  $B_{pre}, I \longleftarrow \mathsf{Extract}_\beta(I)$
  $B \longleftarrow B_{|B_{pre}|:|B|}$
  $i \longleftarrow i + 1$
end while
return T

Algorithm 6 $\mathsf{MDecode}_{sk,\mathsf{Model},\beta,\alpha}(H,T) \to B$

Input: key sk, model Model, bit precision β prompt H, token string T.
Output: secret bit string B.

$I \longleftarrow [\frac{1}{2}, \frac{1}{2} + \frac{1}{2^\alpha}]$
$i \longleftarrow 0$
$B \longleftarrow \emptyset$
while Model does not halt **do**
  $\mathsf{Dist_V}(\cdot) \longleftarrow \mathsf{Predictor}(H||T_{1:i})$
  $r \longleftarrow \mathsf{PRG}_{sk}(i)$
  if $r \le \mathsf{Dist_V}(t-1)$ **or** $r \ge \mathsf{Dist_V}(t)$ **then**
    $I \longleftarrow I \circ [\mathsf{Dist_V}(t-1) - r \mod 1, \mathsf{Dist_V}(t) - r \mod 1]$
  else
    $E \longleftarrow [\mathsf{Dist_V}(t) - r, \mathsf{Dist_V}(t-1) - r + 1]$
    if $I \cap E \neq \emptyset$ **and** $E \not\subset I$ **then**
      $I \longleftarrow I \setminus I \cap E$
    end if
  end if
  $B', I \longleftarrow \mathsf{Extract}_\beta(I)$
  $B \longleftarrow B||B'$
  $i \longleftarrow i + 1$
end while
return B

event does not waste the information, but it is harmful to the correctness of our steganography scheme, as extracting the prefix of secret bits B from 2 intervals may lead to decoding error.

We take a different view on the encoding process. If the interval splitting event does not happen, we maintain an iterative interval I which contains the decimal $\mathrm{Dec}(B_{1:\beta})$. If the interval splitting event happens, we can store the exclusion interval $E = [\mathsf{Dist_V}(t) - r, 1 - (r - \mathsf{Dist_V}(t-1))]$ that do not contain $\mathrm{Dec}(B_{1:\beta})$. With the encoding process going forward, the iterative interval I becomes smaller while the exclusion interval becomes larger. Once I is small enough (or E is big enough), we subtract those excluded part from I without splitting I into multiple intervals.

We modify the original algorithms Encode and Decode. Details are shown in Algorithm 5 and 6.

4 Experiment & Result

4.1 Parameter Settings

We implement the proposed method Shimmer, with security parameter $\lambda = 256$ and $\beta = 64$. We use Shimmer to denote the original steganography scheme $SS = (\mathsf{KeyGen}, \mathsf{Encode}, \mathsf{Decode})$, and Shimmer (R.) to denote the improved steganography scheme $SS' = (\mathsf{KeyGen}, \mathsf{MEncode}, \mathsf{MDecode})$.

4.2 Experiment Settings

We compare our method with the current advanced steganography method DISCOP [8] and METEOR [14], because these method have similar settings to ours. We employ the random sampling to represent normal text generation, as a reference for estimating the impact of the steganography system.

We utilize LLAMA3 [9], LLAMA2 [11], and MISTRAL [10] as the generation models. The computation platform used for experiments consists of 4 × NVIDIA GeForce RTX 3090 GPUs. During testing, we maintain the following configurations: we sample the top-100 tokens and set the maximum number of generated tokens to 512. In a single test, the model generates 10,000 responses, each containing more than 100 tokens. To eliminate interference from the simultaneous execution of multiple tests, only one test is allowed to run on the server at any time.

4.3 Metrics

For the experiments we want to (1) show that our steganography schemes do not degrade the quality of stegotext and

Model	Method	PPL ↓	Entropy (bit/token)	Emb. Cap. (bit/token)	Time↓ (ms/token)	Util.↑ (%)	Chn. Cap.↑ (bit/s)
LLAMA3	Random Sampling	5.9678	1.7224	-	34.6645	-	-
	METEOR	6.0083	1.7442	1.0428	38.9126	0.5979	26.7985
	METEOR (Reorder)	6.0127	1.7429	1.1840	129.3332	0.6793	9.1546
	DISCOP	6.0571	1.6902	1.0519	37.1059	0.6224	28.3486
	DISCOP (Recursive)	6.1978	1.7455	1.5942	177.4882	0.9133	8.9820
	Shimmer	**5.9916**	1.6968	1.3230	**35.6269**	0.7797	**37.1349**
	Shimmer (R.)	**5.9827**	1.7441	1.3531	**35.6318**	0.7758	**37.9745**
LLAMA2	Random Sampling	6.7211	2.4203	-	44.9249	-	-
	METEOR	6.9504	2.4814	1.4301	49.0063	0.5763	29.1820
	METEOR (Reorder)	6.8482	2.4351	1.6217	134.2274	0.6660	12.0817
	DISCOP	6.7734	2.4041	1.3209	48.8803	0.5494	27.0232
	DISCOP (Recursive)	6.9973	2.4346	1.9603	128.8218	0.8052	15.2171
	Shimmer	**6.7998**	2.3670	1.8187	**46.4434**	0.7684	**39.1595**
	Shimmer (R.)	**6.7103**	2.4052	1.8717	**46.7664**	0.7782	**40.0223**
MISTRAL	Random Sampling	2.8639	1.5995	-	45.5452	-	-
	METEOR	2.9967	1.6344	0.7676	48.3961	0.4697	15.8608
	METEOR (Reorder)	2.8845	1.6124	0.8349	142.3430	0.5178	5.8654
	DISCOP	2.8587	1.5854	0.8045	48.4662	0.5074	16.5992
	DISCOP (Recursive)	2.8918	1.5404	1.2176	114.6680	0.7904	10.6185
	Shimmer	**2.8667**	1.6295	1.1418	**47.0969**	0.7007	**24.2436**
	Shimmer (R.)	**2.8786**	1.5978	1.1592	**47.7214**	0.7255	**24.2910**

Table 1: Main results. Our method Shimmer compared to METEOR [14], DISCOP [8] and Random Sampling. We tested these methods with 3 popular LLMs: LLAMA3 [9], LLAMA2 [11] and MISTRAL [10]. Emb. Cap. and Chn. Cap. denote the embedding capacity and channel capacity. Util. denotes the utilization metric. Numbers in **bold** indicate that our method outperforms METEOR (Reorder) and DISCOP (Recursive), which are enhanced versions of METEOR and DISCOP. Underlined numbers indicate that our method surpasses the standard versions of METEOR and DISCOP.

(2) explore the capacity of our steganography scheme in real-world scenario, because the theoretical bound is not tight at all. Therefore we set the following metrics:

Perplexity. This metric is commonly used to assess the fluency of generated texts. For a token string T, it is calculated as

$$\text{Perplexity} = 2^{\frac{1}{|T|}\sum_{n=1}^{|T|} p(T_i)}. \tag{22}$$

Entropy. It represents the expected upper limit of the embedding capacity, expressed in bits per token. For a token string T, the entropy is computed as

$$\text{Entropy} = \frac{1}{|T|}\sum_{n=1}^{|T|} -p(T_n)\log_2 p(T_n). \tag{23}$$

Embedding Capacity. This metric measures the average length of bits that can be embedded and extracted, expressed in bits per token.

Time. We evaluated the time required for each method to execute, measured in bits per token.

Utilization. This metric reflects the ratio of the actual embedding capacity to the theoretical capacity limit (Entropy), computed as

$$\text{Utilization} = \frac{\text{Embedding Capacity}}{\text{Entropy}} \tag{24}$$

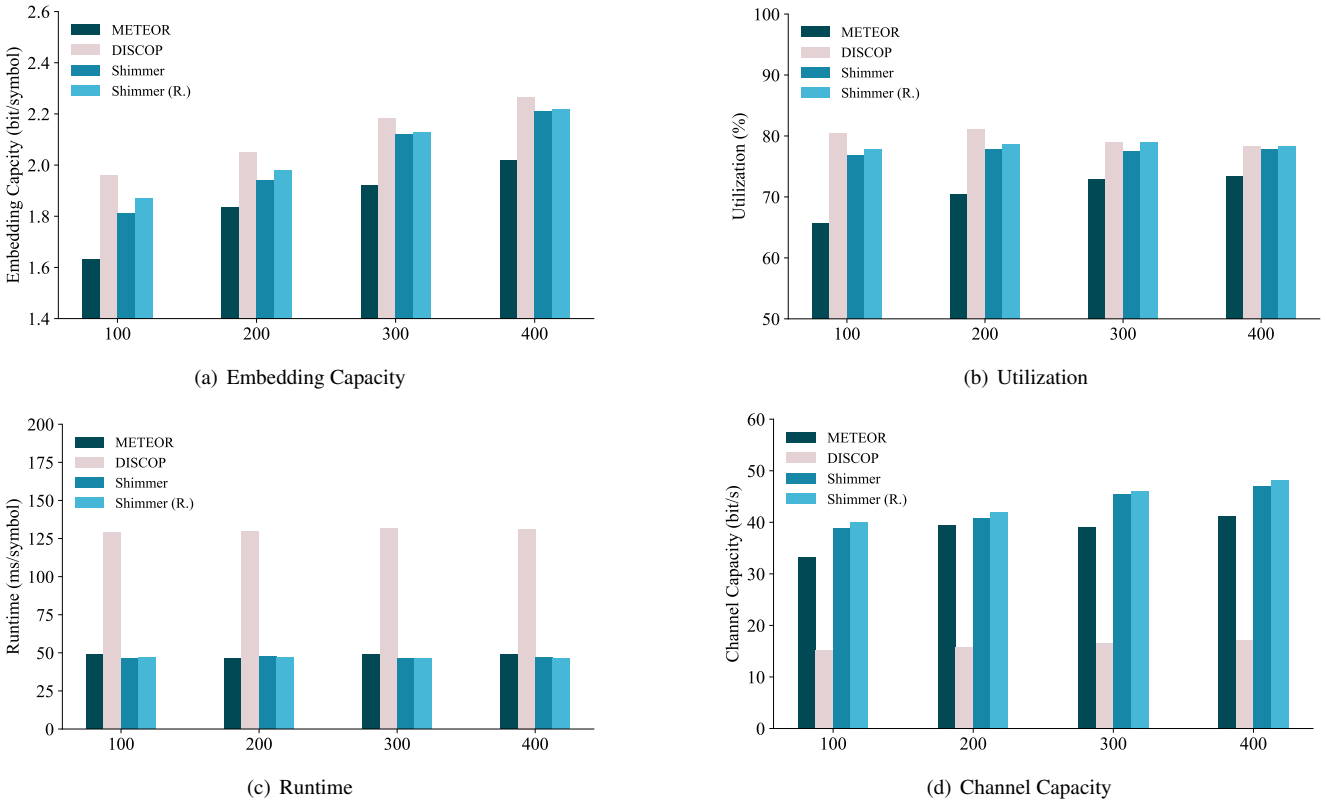
Channel Capacity. It represents the average length of bits that can be transmitted per second. This metric depends on the model and the complexity of the steganography system, computed as

$$\text{Channel Capacity} = \frac{\text{Embedding Capacity}}{\text{Time}} \tag{25}$$

4.4 Results

4.4.1 Overall Performance

As demonstrated in Table 1, Shimmer generally achieves perplexity that is similar to Random Sampling, which confirms that the quality of stegotext is similar to that of covertext. Our method exhibits an increase in utilization compared to METEOR and DISCOP, suggesting that it can leverage a greater portion of the available information for encoding. Shimmer consistently achieves an utilization of around 77%



(a) Embedding Capacity

(b) Utilization

(c) Runtime

(d) Channel Capacity

Figure 5: Metrics of METEOR, DISCOP and Shimmer when sampling from top-100 $\sim$ 400 tokens.

when tested with LLAMA3 and LLAMA2, indicating its effectiveness in fully utilizing the model's capacity for encoding information. Furthermore, the computational overhead introduced by Shimmer is not significant, and the runtime of the algorithm is comparable to that of random sampling. However, DISCOP is notably slower due to the construction of the Huffman tree.

In summary, our method Shimmer demonstrates better performance in terms of utilization, capacity, while maintaining linguistic quality and time efficiency.

4.4.2 The Impact of Various Entropy.

To investigate the performance of these steganography methods under different entropy levels, we tested them under varying top-k settings of LLAMA2. Increasing the value of k results in a slight improvement of the model's output entropy.

The details are illustrated in Fig. 5. We observed that the growth rate of the embedding capacity for these methods is faster than the growth rate of the entropy. From Fig. 5(a), we can see that methods with lower embedding capacity exhibit faster growth. The benefits of deploying the reorder algorithm are more pronounced when the model's output entropy is relatively low. However, as the model provides sufficient entropy, the advantage of the reorder algorithm diminishes. Fig. 5(b) reveals that the utilization of these methods increases with the entropy, leading to a faster increase in the embedding capacity. The performance of DISCOP is slightly anomalous, as its utilization decreases from 81% to 79%. This phenomenon might be attributed to the token-level embedding algorithm used. METEOR's utilization significantly improves due to the increased number of sampled tokens, which results in smaller probabilities for individual tokens and shorter intervals, thus reducing the occurrence of situations without shared prefixes. Shimmer without reordering performs similarly to METEOR for the same reason. We noticed that the runtime of these methods is almost unaffected by the number of sampled tokens. As shown in Figure 5(c), the runtime of these methods remains stable. Consequently, due to the almost unchanged runtime, the results for channel capacity are similar to the embedding capacity, as shown in Fig. 5(d).

5 Conclusion

In this paper, we propose a novel practical and provably secure steganography method, Shimmer. This steganography scheme utilizes an entropy collecting mechanism to improve its capacity while maintain time efficiency. Experimental results validate that Shimmer provides the highest channel

capacity among all of the practical and provably secure works. Moreover, our method can be extended to other types of auto-regressive models that generate audio or visual content. The future extension of our work may focus on solving the interval splitting problem to further increase the capacity.

6 Ethics Considerations

We notice that the steganography techniques may be leveraged by malicious parties to transmit harmful messages, which is similar to the cryptography techniques. Though steganography data is imperceptible, the communication behavior is not. So there still exists some behavioral analysis based approach for detecting steganography. This paper aims at proposing a provably secure steganography method with high bitrate. As nowadays many steganography-based tools are widely used, our work may not intensify this ethical concern.

7 Open Science

Our work complies with the open science policy, and our source code is open at https://doi.org/10.5281/zenodo.15582958.

This package includes:

- The core implementation of our steganography scheme, Shimmer and Shimmer(R.).
- An executable example notebook demonstrating how to use Shimmer with Hugging Face models.
- A list of Python dependencies for a reproducible environment.

References

[1] Michael Backes and Christian Cachin. Public-key steganography with active attacks. In *Theory of Cryptography Conference*, pages 210–226. Springer, 2005.

[2] R.J. Barron, Brian Chen, and G.W. Wornell. The duality between information embedding and source coding with side information and some applications. *IEEE Transactions on Information Theory*, 49(5):1159–1180, 2003.

[3] Luke A. Bauer, James K. Howes, Sam A. Markelon, Vincent Bindschaedler, and Thomas Shrimpton. Leveraging generative models for covert messaging: Challenges and tradeoffs for "dead-drop" deployments. In *Proceedings of the Fourteenth ACM Conference on Data and Application Security and Privacy*, CODASPY '24, page 67–78, New York, NY, USA, 2024. Association for Computing Machinery.

[4] Christian Cachin. An information-theoretic model for steganography. *Information and computation*, 192(1):41–56, 2004.

[5] Miranda Christ and Sam Gunn. Pseudorandom error-correcting codes. In *Annual International Cryptology Conference*, pages 325–347. Springer, 2024.

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

[7] Christian Schroeder de Witt, Samuel Sokota, J Zico Kolter, Jakob Nicolaus Foerster, and Martin Strohmeier. Perfectly secure steganography using minimum entropy coupling. In *The Eleventh International Conference on Learning Representations*, 2022.

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

[9] Aaron Grattafiori et al. The llama 3 herd of models, 2024.

[10] Albert Q. Jiang et al. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023.

[11] Touvron et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[12] Nicholas J Hopper, John Langford, and Luis Von Ahn. Provably secure steganography. In *Advances in Cryptology—CRYPTO 2002: 22nd Annual International Cryptology Conference Santa Barbara, California, USA, August 18–22, 2002 Proceedings 22*, pages 77–92. Springer, 2002.

[13] Tushar M. Jois, Gabrielle Beck, and Gabriel Kaptchuk. Pulsar: Secure steganography for diffusion models. In *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, CCS '24, page 4703–4717, New York, NY, USA, 2024. Association for Computing Machinery.

[14] Gabriel Kaptchuk, Tushar M Jois, Matthew Green, and Aviel D Rubin. Meteor: Cryptographically secure steganography for realistic distributions. In *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, pages 1529–1548, 2021.

[15] Guorui Liao, Jinshuai Yang, Weizhi Shao, and Yongfeng Huang. A framework for designing provably secure steganography. 2025.

[16] Yuang Qi, Kejiang Chen, Kai Zeng, Weiming Zhang, and Nenghai Yu. Provably secure disambiguating neural linguistic steganography. *IEEE Transactions on Dependable and Secure Computing*, pages 1–14, 2024.

[17] Esra Satir and Hakan Isik. A huffman compression based text steganography method. *Multimedia tools and applications*, 70:2085–2110, 2014.

[18] Jiaming Shen, Heng Ji, and Jiawei Han. 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, 2020.

[19] Gustavus J. Simmons. The prisoners' problem and the subliminal channel. In *Advances in Cryptology: Proceedings of CRYPTO '83*, pages 51–67. Plenum, 1983.

[20] Tri Van Le. Efficient provably secure public key steganography. *Cryptology ePrint Archive*, 2003.

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

[22] Xilong Wang, Yaofei Wang, Kejiang Chen, Jinyang Ding, Weiming Zhang, and Nenghai Yu. Icstega: Image captioning-based semantically controllable linguistic steganography. In *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5, 2023.

[23] Yaofei Wang, Gang Pei, Kejiang Chen, Jinyang Ding, Chao Pan, Weilong Pang, Donghui Hu, and Weiming Zhang. Sparsamp: Efficient provably secure steganography based on sparse sampling. *arXiv preprint arXiv:2503.19499*, 2025.

[24] Tianyu Yang, Hanzhou Wu, Biao Yi, Guorui Feng, and Xinpeng Zhang. Semantic-preserving linguistic steganography by pivot translation and semantic-aware bins coding. *IEEE Transactions on Dependable and Secure Computing*, 21(1):139–152, 2024.

[25] Zhen Yang, Zihan Xu, Ru Zhang, and Yongfeng Huang. T-gru: contextual gated recurrent unit model for high quality linguistic steganography. In *2022 IEEE International Workshop on Information Forensics and Security (WIFS)*, pages 1–6, 2022.

[26] 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.

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

[28] Or Zamir. Undetectable steganography for language models. *Transactions on Machine Learning Research*, 2024.

[29] Zachary Ziegler, Yuntian Deng, and Alexander M Rush. 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, 2019.

A Samples of METEOR, DISCOP and Shimmer

Prompt	Method	Stegotext
What is steganography?	Shimmer	Steganography is the practice of hiding one piece of data inside another in such a way that it's difficult to detect. This technique is often used to hide sensitive or confidential information, like passwords, Images, Audio, Video, etc. within files, texts, or other data carriers. The hidden data doesn't alter the appearance, size, or behavior of the original data, hence it's called "steganography" which comes from the Greek words "steganographos," meaning "hidden writer." This method is more secure compared to traditional encryption methods because it is much harder to detect the presence of hidden data.
	Shimmer (R.)	Steganography is the practice of hiding one piece of data inside another in a way that is difficult to detect. It is a method of disguising information by embedding it within another piece of data in such a way that it is not visible or easily distinguishable from the original data. This technique is used to transmit sensitive information securely, as it provides an additional layer of protection by making the data inconspicuous to those who do not have the necessary tools or knowledge to access it. The term "steganography" comes from the Greek words "steganos," meaning hidden or covered, and "graphein," meaning to write or engrave.
	METEOR	Steganography is the practice of hiding one piece of data inside another in such a way that it's difficult to detect. This technique is often used to hide sensitive or confidential information, like documents, images, or videos, within seemingly innocuous files like music, videos, or images. The hidden data is typically encoded in a way that makes it unreadable without the use of a specific decoding key or software. This method of data concealment is often used to bypass security measures, evade surveillance, or protect privacy.
	DISCOP	At first sight steganography might look a bit mysterious. The simple definition is that mini-steganography is the process, science or art of hiding one digital file inside another in such a way that the hidden data is unpredictible , and discoverable only to those people/systems who are specifically designed to detect or recover it.

Table 2: MISTRAL-7B [10] generated samples.