Look Who's Talking Now: Covert Channels From Biased LLMs

Daniel Silva[1], Frederic Sala[2], Ryan Gabrys[1],
[1]Naval Information Warfare Center Pacific, [2]University of Wisconsin-Madison,
{daniel.silva61, ryan.c.gabrys}.civ@us.navy.mil, fsala@wisc.edu

Abstract

Large language model-based steganography encodes hidden messages into model-generated tokens. The key tradeoff is between how much hidden information can be introduced and how much the model can be perturbed. To address this tradeoff, we show how to adapt strategies previously used for LLM watermarking to encode large amounts of information. We tackle the practical (but difficult) setting where we do not have access to the full model when trying to recover the hidden information. Theoretically, we study the fundamental limits in how much steganographic information can be inserted into LLM-created outputs. We provide practical encoding schemes and present experimental results showing that our proposed strategies are nearly optimal.

1 Introduction

Steganography is the art of hiding information within other non-secret text or data, commonly referred to as *covertext*. This technique ensures that only someone with the appropriate decoding key can access the hidden information, thus providing a layer of security. Initially used in historical contexts for espionage and secret communications, steganography has evolved with technological advancements. When coupled with large language models (LLMs), steganography embeds hidden information into text generated by these models.

Our approach is to ***bias the output distribution of the LLM*** in order to allow for the encoding of the hidden message (the *stegotext*). However, in order to maintain secrecy, we bound the maximum bias allowed for each sampled token. We study the central challenge for such techniques: the tradeoff between the secrecy of the scheme, which is usually measured in terms of how statistically similar the stegotext and the covertext are, and how much information can be communicated by the stegotext.

To recover the stegotext, we must use a *decoding function*. One key aspect of our work is that we do not necessarily assume that this function has access to the underlying LLM. Instead, we consider the following two scenarios: (i) The LLM is only accessible during encoding, (ii) The LLM is accessible during encoding and a quantized version of the model is accessible for the decoding. We also consider a third scenario where the encoder and decoder both have access to the same model and show that under this (much simpler) setting, nearly optimal steganogographic schemes exist. We assume in all three cases that the encoder always initially samples from the unquantized LLM before potentially biasing its output distribution.

Our contributions are the following. First, we introduce a simple encoding model whose structure is an extension of previous techniques used in watermarking, but with the goal of encoding larger quantities of information. We propose encoding schemes for each of the three scenarios and evaluate these in light of theoretical bounds we develop.

Experimentally, using data extracted from several large language models, we show that when the maximum bias is at least two, positive information rate schemes are achievable, and that in many cases, our schemes empirically appear to approach the optimum information rate.

2 Related Works

We briefly present some related work, predominantly focusing on steganography and watermarking with large language models.

LLM Steganography and Watermarking. There are a large number of works tackling watermarking for large language models; such watermarking techniques share aspects with steganography as they seek to encode hidden messages in language model aspects. We highlight two such works: Kirchenbauer et al. (2024)

Findings of the Association for Computational Linguistics: EMNLP 2024, pages 16648–16658
November 12-16, 2024 ©2024 Association for Computational Linguistics

tackles digital watermarking in the , proposing methods that ensure watermarks persist despite modifications to the generated text. The goal is to enhance traceability of content and ensure authenticity. Zhao et al. (2023) introduces a framework for robust watermarking. Their method supports the embedding and efficient verification of watermarks with provable guarantees.

On the steganography side, Ziegler et al. (2019) introduced a method for neural network-based steganography. The underlying technique uses arithmetic encoding. The basic setting is similar to ours: the goal is to embed a hidden message in such a way that model's generated output does not change in a detectable way. Similarly, de Witt et al. (2023) focused on achieving near-perfect steganographic security through a technique called minimum entropy coupling.

On a more theoretical note, Zhang et al. (2021) proposes a generative linguistic steganography method that is provably secure. They ensure that modifications made to embed information are completely imperceptible.

Prompt Engineering, Black-Box Approaches. Recent studies have explored the use of prompt engineering and black-box approaches to steganography in LLMs. Wu et al. (2024) shows how soft prompts can be used to guide LLMs to generate data that includes embedded steganographic content without needing direct access to model internals. This method proves particularly useful in scenarios where users must rely on black-box API interactions.

Cryptographic Watermarking. Christ et al. (2023) introduces cryptographically-inspired undetectable watermarks for language models, where the presence of the watermark can only be confirmed through the possession of a secret key

Our Approach. Our approach shares a common goal with Kirchenbauer et al. (2024), as we aim to design schemes that perturb or bias the outputs of a large language model in a controlled manner to embed information. The key difference is that *we focus on encoding the maximum possible amount of information within a given bound on the perturbation magnitude*.

Similar to the challenges addressed in de Witt et al. (2023) and Ziegler et al. (2019), one of our primary obstacles is designing codes that minimize the statistical distance (or divergence) between the generated stegotexts and the covertexts. Our problem is more challenging than these previous works, as we do not assume the decoder has access to token distributions during decoding, aligning with black-box approaches. In contrast to these recent methods, we aim to maximize the information rate by allowing the encoder to retain access to the internal state of the LLM during encoding, even if this assumption does not hold during decoding.

3 Problem Formulation

A language model is a function f that accepts as input a sequence of tokens $x^{(-N_p)}, \dots, x^{(t-1)}$. This sequence can be partitioned into two disjoint subsequences consisting of: (i) the prompt $\boldsymbol{x}_p = (x^{(-N_p)}, \dots, x^{(-1)})$, and (ii) the sequence $\boldsymbol{x}_{t-1} = (x^{(1)}, \dots, x^{(t-1)})$, which is the first $t-1$ tokens previously produced by the model. Given the input $(\boldsymbol{x}_p, \boldsymbol{x}_{t-1})$, the model outputs a vector of V logits, one for each word of the vocabulary $\mathcal{V} = [V]$. These logits are passed into a softmax function producing a distribution over $\mathcal{V}$. The next token at time t is then sampled according to the resulting distribution. We refer to the probability that the t-th token is equal to the k-th element from the vocabulary $\mathcal{V}$ as $p_k^{(t)}$ and represent the distribution at time t as the vector $\boldsymbol{p}^{(t)} = \left(p_1^{(t)}, p_2^{(t)}, \dots, p_V^{(t)}\right)$.

For inputs $x^{(-N_p)}, \dots, x^{(t-1)}$, f outputs

$$f\left(\boldsymbol{x}_p, \boldsymbol{x}_{t-1}\right) = \boldsymbol{p}^{(t)}. \tag{1}$$

For a distribution $\boldsymbol{p} = (p_1, p_2, \dots, p_V)$ over the set of tokens, we use the notation $Z \sim \boldsymbol{p}$ to denote the fact that Z is sampled from the distribution $\boldsymbol{p}$ so that the probability that Z is equal to the k-th token is p_k if $Z \sim \boldsymbol{p}$.

Recall $\boldsymbol{x}_p = \left(x^{(-N_p)}, x^{(-N_p+1)}, \dots, x^{(-1)}\right)$ is our input prompt. Our goal is to develop an *embedding function* denoted $\widehat{f}$ that can ***hide a message in the tokens*** produced by the LLM. To obtain the hidden message, we also need an *efficient decoder* $\mathcal{D}$. For a given a binary message to be hidden $\boldsymbol{u} \in \{0, 1\}^m$ and a starting prompt $\boldsymbol{x}_p$, the embedding function and decoder must satisfy a pair of properties detailed below. Let $Y_1 \sim \widehat{f}\left(\boldsymbol{x}_p, \boldsymbol{u}\right)$ and recursively define Y_t as

$$Y_t \sim \widehat{f}\left(\boldsymbol{x}_p, Y_1, \dots, Y_{t-1}, \boldsymbol{u}\right).$$

Requirements. We have two requirements:

1. **Reliability Property**: $\mathcal{D}\left(\boldsymbol{x}_p, Y_1, \dots, Y_N\right) = \boldsymbol{u}$, with probability at least $1 - \Delta$ for small

$\Delta > 0$, where Δ becomes increasingly small as m, N increases.

2. **Security Property**: For any $t \in [N]$ and sequence of tokens $\boldsymbol{x}_{t-1}$, the following holds: for all $k \in [V]$ we have $\hat{p}_k^{(t)}/p_k^{(t)} \leq \beta$. That is, each token $\hat{p}_k^{(t)}$ is the result of biasing the original distribution by a factor of at most β. Here, $\hat{\boldsymbol{p}}^{(t)} = \widehat{f}\left(\boldsymbol{x}_p, \boldsymbol{x}_{t-1}, \boldsymbol{u}\right)$ and $\boldsymbol{p}^{(t)} = f\left(\boldsymbol{x}_p, \boldsymbol{x}_{t-1}\right)$.

The first condition ensures that we can recover our information bits with vanishing levels of error. The second condition seeks to ensure $\widehat{f}$ makes the generated stegotext similar to the covertext. The level of similarity is controller by the parameter β. We will study the properties of β in the following.

Approach. We use a simple strategy to encode the binary information from $\boldsymbol{u}$ into the tokens. At each generation step, we partition the token vocabulary into two parts, so that the tokens in one part represent 1, and the others represent 0. Since we do not directly control the choice of next token, to make it more likely a token from the correct part (the one representing the next bit of $\boldsymbol{u}$ we wish to hide), we bias the probabilities of tokens in this part. This makes it easier to achieve ***reliability***. However, to meet the ***security*** property, we can only bias up to a level β. In some scenarios, this will lead to a decoding error probability that is too high (violating reliability). To account for this case, we first encode $\boldsymbol{u}$ with an error-correcting code, which protects against such errors. However, by coding, we may sacrifice efficiency (so that we may need many more tokens to hide a message compared to its length). We address this tradeoff next.

Goals. Our primary aim is to develop techniques that, for fixed levels of β, achieve the largest possible *information rate*. We explain this below.

We assume that our hidden information $\boldsymbol{u} \in \{0,1\}^m$ is comprised of m i.i.d. bits (each bit is equal to one with probability 1/2). It is easy to adapt these results for other distributions, but we do not do so here for simplicity. As detailed earlier, at the beginning of the embedding process, we first perform an intermediate encoding step where the information $\boldsymbol{u} \in \{0,1\}^m$ is encoded into a binary codeword $E(\boldsymbol{u}) = \boldsymbol{c} \in \mathcal{C} \subseteq \{0,1\}^N$.

The motivation for the encoding process is standard in the error-correcting codes literature. The idea is that each hidden message sequence $\boldsymbol{u}$ is encoded (via the encoder E) into a longer *codeword* from a set of codewords (a code) $\mathcal{C}$. These codewords have longer length, so that $N > m$, and therefore take more tokens to embed. However, the code $\mathcal{C}$ has additional structure that makes it possible to reliably decode into the original message despite errors affecting codeword bits. Our goal therefore is to obtain this desirable property while maximizing the information rate $\frac{m}{N}$. More formally, using coding-theoretic terminology, we refer to any coding scheme that satisfies the two properties as an (N, m, β)-stego code with rate $\frac{m}{N}$.

Our aim is to develop codes that achieve the maximum possible rate under two different scenarios. (i) ***Decoder Uninformed (DU)***: At the time of decoding, the distribution $\boldsymbol{p}^{(t)}$ is unknown. (ii) ***Decoder Informed (DI)***: At decoding, the distribution of $\widetilde{\boldsymbol{p}}^{(t)}$ is known but not $\boldsymbol{p}^{(t)}$, where $\widetilde{\boldsymbol{p}}^{(t)}$ represents an approximation of $\boldsymbol{p}^{(t)}$, which we assume is obtained using a quantized model.

Embedding Process. The embedding procedure for the hidden text relies on three inputs: (i) The value of each component of $\boldsymbol{c}$, the codeword representing a hidden message, (ii) A ***partition function***, which at each time step t assigns each token in $\mathcal{V}$ either the logical value of 0 or 1, and (iii) the bias parameter $\beta > 1$, which allows us to perturb our distribution in order to represent the bits from $\boldsymbol{c}$.

Recall that $\boldsymbol{p}^{(t)} = f\left(\boldsymbol{x}_p, \boldsymbol{x}_{t-1}\right)$ and suppose that $\mathcal{P}_0^{(t)}$ is the set of all tokens that, at time t, are assigned to zero by the partition function. Similarly, let $\mathcal{P}_1^{(t)} = \mathcal{V} \setminus \mathcal{P}_0^{(t)}$. We also denote the sum of the probabilities of all the tokens in $\mathcal{P}_0^{(t)}$ as $P_0^{(t)}$ and $P_1^{(t)} = 1 - P_0^{(t)}$. For shorthand, we denote the maximum of $P_0^{(t)}$ and $P_1^{(t)}$ as $P_{\max}^{(t)}$ and with a slight abuse of notation we denote $\min\left(\beta P_{c_t}, 1\right)$ as $\lfloor \beta P_{c_t} \rceil$. Under this setup, for $k \in [N]$ our biased distribution is the following:

$$\widehat{f}\left(\boldsymbol{x}_p, \boldsymbol{x}_{t-1}, \boldsymbol{u}_k\right) = \qquad (2)$$
$$\begin{cases} \frac{\lfloor \beta P_{c_t} \rceil p_k^{(t)}}{P_{c_t}^{(t)}} & \text{If } k \in \mathcal{P}_{c_t}^{(t)}, \\ \frac{\left(1 - \lfloor \beta P_{c_t} \rceil\right) p_k^{(t)}}{1 - P_{c_t}^{(t)}} & \text{otherwise.} \end{cases}$$

Note that (2) biases tokens in such a manner that we are more likely to sample from the set of tokens representing c_t. We assume $\beta > 1$ and that the information $\boldsymbol{u}$ is determined before the start of encoding.

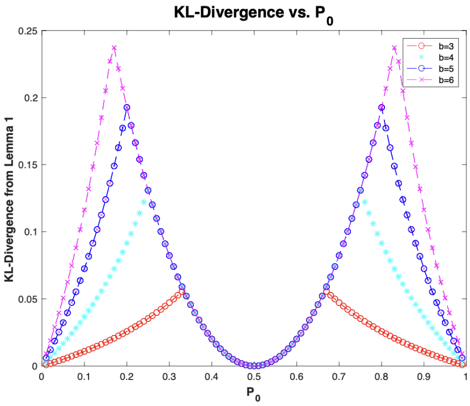

Figure 1: KL divergence between original and biased distributions as a function of P_0 for several values of β (colored curves). The crucial insight: near $P_0 = 1/2$, *the divergence is not a function of β* .

4 Insights and Methods

In order to achieve our goals, we must (i) select a partition function and (ii) choose an error-correcting code that help us meet the two properties and maximize information rate. In this section, we first obtain some insights into the properties of our problem to help us with these tasks. Equipped with these, we propose a family of partition function techniques and make our choice of error-correcting code. We defer proofs and a full set of theoretical results (including a coding-theoretic analysis on the maximum rate of an (N, m, β)-stego code and an achievability argument) to the Appendix.

We begin by considering the effect that β (the maximum bias in the security property) has on the the relationship between our biased and unbiased language model distributions. A simple result (Lemma 1), stated in the Appendix, helps us understand the behavior of $D\left(\boldsymbol{p}^{(t)}||\hat{\boldsymbol{p}}^{(t)}\right)$, the KL divergence between the original and biased distributions. We illustrate this quantity as a function of $P^{(0)}$ in Figure 1.

We observe some trends. First, for a fixed β, $D(\boldsymbol{p}^{(t)}||\hat{\boldsymbol{p}}^{(t)})$ is symmetric about the point $P_0 = \frac{1}{2}$. This is intuitive, as $P_0 = P_1 = 1 - P_0$ results in the same partition (with a reversal of the labels of the parts). As expected, the KL divergence grows as a function of β *except in the interval near* $\frac{1}{2}$. In particular, when $P_0 \in (\frac{1}{3}, \frac{2}{3})$, the KL-divergence does not depend on β, which implies that a key ingredient are partition functions that produce P_{max} close to $\frac{1}{2}$. More concretely, notice if we can develop partition functions that guarantee with high probability P_0 lies within a small range around $\frac{1}{2}$, then there is no need to use larger values of β.

Our second insight comes from our development of upper bounds on the achievable rate of an (N, m, β)-stego code (Theorem 1 in the Appendix). The maximum rate is reached whenever $\beta P_{\text{min}}^{(t)} \geq 1$. This is intuitive, as when $\beta P_{\text{min}}^{(t)} \geq 1$, we can deterministically bias the tokens in such a manner that we are guaranteed to extract the correct part for each output token. One of the challenges which we will encounter, however, is that the behavior of $P_{\text{max}}^{(t)}$ is highly variable, and it can fluctuate significantly between $\frac{1}{2}$ and 1. This motivates our error-correcting code approach: we allow a certain number of tokens to be incorrectly sampled and then rely on the properties of $\mathcal{C}$ to correct the incorrectly sampled tokens.

4.1 Partitioning And Coding

Now are ready to introduce our partition function and error-correcting code approaches.

Partition Functions. Recall the purpose of the partition function, denoted F_t, is to assign to each token in $\mathcal{V}$ to either $\mathcal{P}_0^{(t)}$ or $\mathcal{P}_1^{(t)}$ as described in (2). Assume that at each step t, $\Pr(c_t = 0) = \Pr(c_t = 1) = 1/2$. Using our insights, we can minimize the KL distance and maximize the potential information rate by designing partitioning schemes *that are approximately balanced.*

Unfortunately, this goal is related to the NP-complete *subset sum* difference problem where the aim is to identify two disjoint sets $\mathcal{P}_0, \mathcal{P}_1 \subseteq \mathcal{V}$ where $\left|\sum_{k \in \mathcal{P}_0} p_k - \sum_{k \in \mathcal{P}_1} p_k\right|$ is minimal[1]. For shorthand, for any partition $\mathcal{P}_0, \mathcal{P}_1$ we will refer to the difference in magnitude between the two partitions as the ***partition bias***. Our goal is to develop partitioning schemes that minimize the partition bias under the DU and the DI settings. Recall that for the DI setting the decoder has access to $\widetilde{\boldsymbol{p}}^{(t)}$, which is an approximation of $\boldsymbol{p}^{(t)}$.

In both the DU and DI settings, the decoder does ***not*** have access to the true distribution $\boldsymbol{p}^{(t)}$. For DU, we use a pseudo-random (and invertible) hash function. The remainder of this section will be dedicated to investigating potential partition functions that can address the DI scenario. In addition to considering the partition bias, we also measure the ***partition error***, defined as $\sum_{k \in [V]} \left| p_k^{(t)} - \widetilde{p}_k^{(t)} \right|$.

[1] Guaranteeing the difference is below a certain threshold ensures such a partitioning scheme is highly desirable both in terms of security and potential information rate

For DI, we investigate the following three types of partitioning schemes:

1. ***Approximate subset-sum (ASU)***: We use an ϵ-approximation algorithm where $\epsilon = .01$. The algorithm takes as input a target probability T and identifies (to within a multiplicative factor of $1 + \epsilon$) the largest partition whose probabilities sum to less than T. We set $T = \frac{1}{2}$. This approach gives the lowest partition bias but the largest partition error.
2. ***Hash***: The partition function in this case consists of an invertible hash function, which is seeded with the previously generated token. Although the partition error is zero, the partition bias can be quite large.
3. ***Greedy***. We consider three variations:
   (a) ***Greedy ascending (GA)***: The tokens are first ordered in a list in ascending order according to their probabilities. $\mathcal{P}_0$ is initialized to be the emptyset. Starting from the beginning of the list, each token is added to $\mathcal{P}_0$ if and only if after the addition of the new token the sum of all the probabilities in $\mathcal{P}_0$ is at most $\frac{1}{2}$.
   (b) ***Greedy descending (GD)***: GD uses the same logic as GA except that the tokens are first ordered in descending order.
   (c) ***Greedy random (GR)***: GR also uses the same logic as GA except that the tokens are initially placed in a random order.

In general, ASU yields partitions with the lowest partition bias and the largest partition error whereas Hash has no partition error but has a large partition bias. The performance of the greedy approaches tend to be in between the two extremes offered by Hash and ASU. In order to find a better compromise between the tradeoff between the partition bias and the partition error, we explored the following two-round "hybrid" approaches:

1. ***Hash approximate (HA)***: This approach consists of a labeling procedure comprised of two rounds. In the first round, we generate two partitions $\mathcal{P}_{\max}$ and $\mathcal{P}_{\min}$ using the Hash approach discussed previously. Then, in the second round, we employ ASU to relabel a subset of tokens from $\mathcal{P}_{\max}$. In particular we instantiate ASU to identify a set of elements $\mathcal{P}' \subseteq \mathcal{P}_{\max}$ given a target sum of $T = P_{\max} - P_{\min}$. Any tokens which were initially assigned to $\mathcal{P}_{\max}$ but were subsequently assigned to $\mathcal{P}'$ in the second round, have their labels updated so that after the completion of the two rounds we update $\mathcal{P}_{\min} = \mathcal{P}_{\min} \cup \mathcal{P}'$ and $\mathcal{P}_{\max} = \mathcal{V} \setminus \mathcal{P}_{\min}$.
2. ***Hash Greedy***: Similar to before, there are three variations of Hash Greedy all of which follow a similar two-round format. The first round of all three variations is exactly the same where in the first round (and similar to HA), two partitions $\mathcal{P}_{\min}$ and $\mathcal{P}_{\max}$ are created using the Hash approach. The goal in the second stage will be to identify a subset of tokens $\mathcal{P}' \subsetneq \mathcal{P}_{\max}$ whose target sum is $T = P_{\max} - P_{\min}$.
   (a) ***Hash Greedy Ascending (HGA)*** uses the GA approach to determine $\mathcal{P}'$.
   (b) ***Hash Greedy Descending (HGD)*** employs GD to determine $\mathcal{P}'$.
   (c) ***Hash Greedy Random (HGR)*** leverages GR to identify $\mathcal{P}'$.

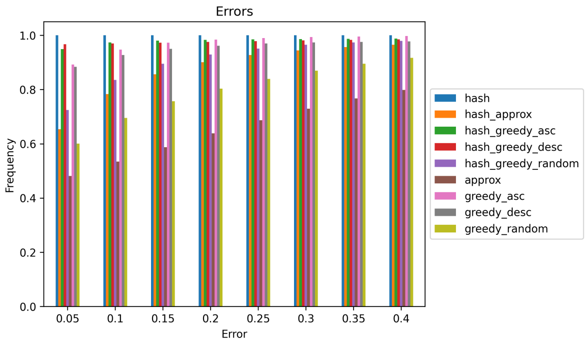

Figure 2: Partition error across various choices of partition functions.

Partition Function Comparisons. To gain a better understanding of these tradeoffs with our proposed schemes, we generated 64,000 tokens using OPT-1.3B (Zhang et al., 2022) with prompts from the C4 RealNewsLike (Raffel et al., 2020) dataset where 200 output tokens are generated per prompt.

Figures 2 and 3 show the distribution of the partition error and the partition bias across the 9 partition functions discussed thus far. We can interpret Figure 2, for example, by observing that the lines shown under the 0.1 label on the x-axis indicate that for the Hash scheme the partition error is at most 0.1 100% of the time (in fact, as discussed earlier the partition error is always zero for the Hash scheme). Furthermore, for the GD partition approach roughly 90% of the time the partition error is at most 0.1.

Figure 3 displays the frequency with which each

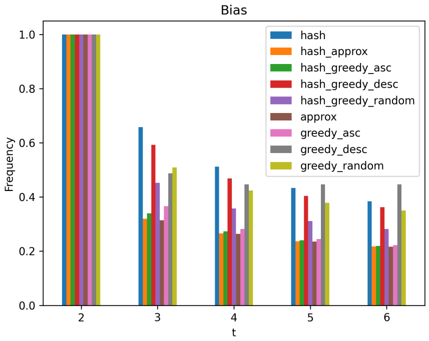

Figure 3: Partition bias across various choices of partition functions

partition function yielded $P_{\min} < \frac{1}{t}$ as a function of t. As an example, for the point $t = 2$, $P_{\min} < \frac{1}{2}$ nearly 100% of the time. However, when $t = 4$, we see more variation with respect to the partition bias and in particular, we observe ASU resulted in $P_{\min} < \frac{1}{4}$ with a frequency of around .25 whereas hash exhibited much larger partition bias with $P_{\min} < \frac{1}{4}$ occurring with a frequency of close to .5.

An interesting insight gained from this is that increasing the bias parameter β in (2) *does not yield much benefit* beyond $\beta = 3$ for the better partitioning schemes. For example, looking at HGA, the fact that $\mathcal{P}_{\min} < \frac{1}{3}$ with a frequency of roughly .33 implies that if we set $\beta = 3$ in (2) then .33 of the time we have $\beta P_{\min} < 1$ implying that it is possible to still sample an incorrect token at most .33 of the time if the value of the bit we wish to encode maps to the smaller partition. Increasing the bias parameter to $\beta = 5$, we see that $\beta P_{\min} < 1$, which using the same logic implies that it is still possible to sample an incorrect token at most .25 of the time.

For the experimental results in the following section, we used the HGA partition function for the DI scenario, since HGA seems to offer a good tradeoff between exhibiting low levels of partition errors while maintaining smaller partition bias. For example, from Figure 2, it can be observed that HGA has a partition error of at most 0.05 with a frequency of roughly 95% and where $P_{\min} < \frac{1}{3}$ roughly $\frac{1}{3}$ of the time.

Coding Schemes. Recall from Section 3 that we encode the information $\boldsymbol{u} \in \{0,1\}^m$ into a codeword $\boldsymbol{c} \in \mathcal{C} \subseteq \{0,1\}^N$. Then, the token distribution, which is sampled to produce the output token of the LLM, is biased according to each bit of the codeword $\boldsymbol{c}$ as discussed in (2). The codes $\mathcal{C}$ we use are LDPC codes (Gallager, 1962) from the DVB-S2 standard (Yadav and Parhi, 2005). This family offers a flexible range of compatible rates and it has been shown to achieve near Shannon limit performance. For our results, we ran simulations using such codes of rates $1/4, 1/3, 2/5, 1/2, 3/5, 2/3, 3/4, 4/5, 5/6, 8/9$ and $9/10$. Each had a block length of 64800 bits.

For both the encoding and decoding functions, we use the DVB-S2 repository (ldp). The decoding proceeds follows. Suppose $\boldsymbol{y} = (y_1, \ldots, y_N)$ denote the sequence of tokens encoded according to (2) and let Suppose $\boldsymbol{z} = (G_1(y_1), \ldots, G_t(y_N)) = (z_1, \ldots, z_N) \in \{0,1\}^N$ represent the logical values extracted using the labeling G_t. For the DU setting, we initialized the decoder log likelihood ratios (LLRs) to be

$$\mathrm{LLR}_j = \begin{cases} \log\left(\frac{1-\Delta}{\Delta}\right), & \text{if } z_j = 0 \\ \log\left(\frac{\Delta}{1-\Delta}\right), & \text{otherwise,} \end{cases} \tag{3}$$

where the parameter Δ was determined beforehand via experimentation.

The LLRs for the DI setting were less straightforward. For $t \in [N]$, we begin by making use of the quantized LLM to determine the probability vector $\widetilde{\boldsymbol{p}}^{(t)}$. Next, we perform the HGA approach discussed in Section 4.1 provided the input $\widetilde{\boldsymbol{p}}^{(t)}$, which produces the partitions $\widetilde{\mathcal{P}}_0^{(t)}, \widetilde{\mathcal{P}}_1^{(t)}$ as output. Recall we refer to the resulting labeling at G_t where given a token k as input $G_t(k) = 0$ if $k \in \widetilde{\mathcal{P}}_0^{(t)}$ and otherwise $G_t(k) = 1$. For shorthand, let $\widetilde{P}_0^{(t)} = \sum_{k \in \widetilde{\mathcal{P}}_0^{(t)}} \widetilde{p}_k^{(t)}$ and $\widetilde{P}_1^{(t)} = 1 - \widetilde{P}_0^{(t)}$.

For the DI setting, given $j \in [N]$ we set the j-th LLR to be:

$$\mathrm{LLR}_j = \begin{cases} \log\left(\frac{\lfloor \beta \widetilde{P}_0^{(t)} \rfloor (1-\widetilde{\mu}) + \left(1 - \lfloor \beta \widetilde{P}_0^{(t)} \rfloor\right)\widetilde{\mu}}{\left(1 - \lfloor \beta (1-\widetilde{P}_0^{(t)}) \rfloor\right)(1-\widetilde{\mu}) + \lfloor \beta (1-\widetilde{P}_0^{(t)}) \rfloor \widetilde{\mu}}\right), & z_j = 0 \\ \log\left(\frac{\left(1 - \lfloor \beta \widetilde{P}_0^{(t)} \rfloor\right)(1-\widetilde{\mu}) + \lfloor \beta \widetilde{P}_0^{(t)} \rfloor \widetilde{\mu}}{\lfloor \beta (1-\widetilde{P}_0^{(t)}) \rfloor (1-\widetilde{\mu}) + \left(1 - \lfloor \beta (1-\widetilde{P}_0^{(t)}) \rfloor\right)\widetilde{\mu}}\right), & z_j = 1, \end{cases}$$

where $\widetilde{\mu}$ is an approximation for μ. The details behind the computations are verified in the appendices.

5 Experimental Results

We evaluated the performance of our coding schemes, focusing on determining the highest achievable information rate given a fixed level of bias.

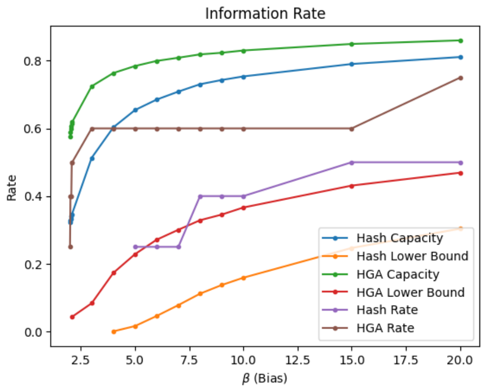

Figure 4: Information Rates Across Different Schemes

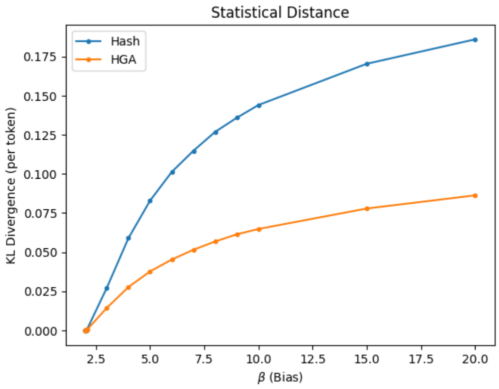

Figure 5: KL divergence of HGA and Hash.

Setup. We use OPT-1.3B with prompts from the C4 dataset's RealNewsLike subset where 200 output tokens are generated per prompt (Raffel et al., 2020; Zhang et al., 2022). To address both scenarios, we simulated the performance of two code designs:

1. **Hash**: For this setting, we used a pseudorandom hash function for the function F_t which at each time instance assigns each output token a logical value 0 or a logical value 1. The hash function at time t is seeded with the output token from time $t-1$ or the last token of the input prompt when $t = 1$. (DU)
2. **Hash Greedy Ascending (HGA)**: The function F_t is constructed using a two step labeling process that relies on a combination of a hash function along with a simple greedy algorithm as described in Section 4.1. (DI)

In order to accommodate the length of our LDPC code, we coded across a sequence of outputs taken from various prompts. Since under our setting there are 200 tokens per output prompt, our codewords were embedded in the concatenation of $\frac{64800}{200} = 324$ prompts.

For both the Hash and HGA settings, the decoding is straightforward. For Hash, we simply use the previously generated token to recover the hash. Once we know F_t, the recovery process amounts to decoding an LDPC code over a channel whose LLRs are given by (3). Similarly, for HGA, we use the quantized version of the model to determine F_t and then perform decoding.

To determine the highest information rate, we let β vary from 1 to 7. We encoded up to 100 codewords and if each of the 100 codewords was recovered error-free, we consider the rate of the resulting code achievable.

Results. Figure 4 shows the rates achieved as a function of the bias which was introduced. As expected, the rate is increasing as a function of the bias. For the HGA method, in particular, we see the rate increasing from .25 at $\beta = 2$ to .75 at $\beta = 20$. For the Hash method, we were unable to achieve positive rate schemes for $\beta < 5$. However, the achievable rate for Hash increased from .25 when $\beta = 5$ to .40 when $\beta = 15$. The HGA and Hash lower bounds were computed using the expression from (5). In general, both the upper and lower bounds were tighter for the HGA method than the Hash and we suspect these distances would converge faster than the results for the Hash method.

Figure 5 displays the KL divergence as a function of β. Here we observe that the bias increases from 0 at $\beta = 2$ for both methods to .175 for Hash and .075 for HGA. In general, HGA exhibits smaller KL divergence than the Hash, probably as a result of the better quality partitions it produces.

Figures 6 and 7 focus on the result for HGA in the regime where β is between 2 and 2.1. Note that at $\beta = 2$, the KL divergence between our biased and unbiased outputs is approximately zero implying that we were able to achieve perfect security while encoding error-free at a rate of $\frac{1}{4}$. Note also that the achievable rate improves dramatically in this regime increasing from $\frac{1}{4}$ at $\beta = 2$ to $\frac{1}{2}$ using $\beta = 2.1$. This means that by increasing the KL divergence from 0 to only .0006, we can double our achievable rate.

In order to investigate whether we could further improve the trade-off between information rate and security, we considered a third scenario where the encoder has access to both the quantized and unquantized models. As with the other two methods, we assume that the encoder will always sample from a potentially biased version of the unquan-

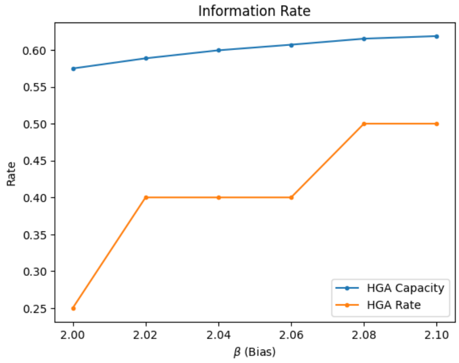

Figure 6: HGA Rate vs. Bias

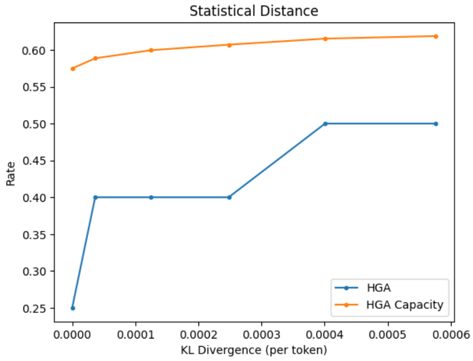

Figure 7: HGA Rate vs. KL divergence

tized outputs, but here the idea will be to attempt to reduce the number of tokens we bias by leveraging the quantized model which is now shared between the encoder and decoder. Our approach, which will be discussed in the next paragraph, will be to only bias the distribution when we know for certain that the correct partition will be sampled at the output (after biasing).

Under this setting, we considered the following encoding procedure. Suppose $\widetilde{P}_{\min}^{(t)}$ represents the total probability of the smaller of the two partitions under the quantized model. We assume that our labeling function F_t is the same pseudo-random function used by Hash in the previous simulations. We will bias the token distribution at time t according to (2) and sample from the resulting distribution if $\widetilde{P}_{\min}^{(t)}\beta \geq 1$. Otherwise, we will simply sample from the unbiased token distribution. The decoding proceeds as follows. For every time $t \in [N]$, we check whether or not $\widetilde{P}_{\min}^{(t)}\beta \geq 1$, which is possible since by assumption the decoder has access to the quantized model. If $\widetilde{P}_{\min}^{(t)}\beta \geq 1$, then we assume that the value of c_t (the t-th bit of our codeword $\boldsymbol{c} \in C$) is equal to $F_t(k)$ where k represents the token which was sampled at time k. Otherwise, we know that the t-th component of $\boldsymbol{c}$ was not encoded using the current token and proceed to the next one.

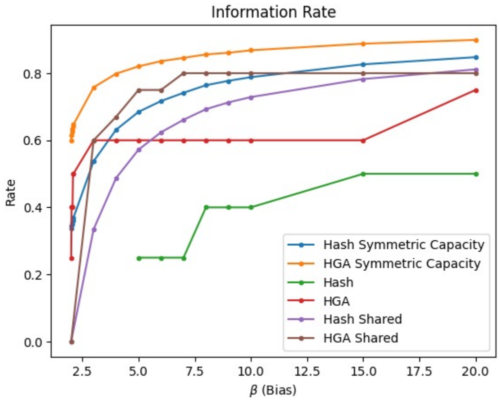

Figure 8: Shared Quantized LLM

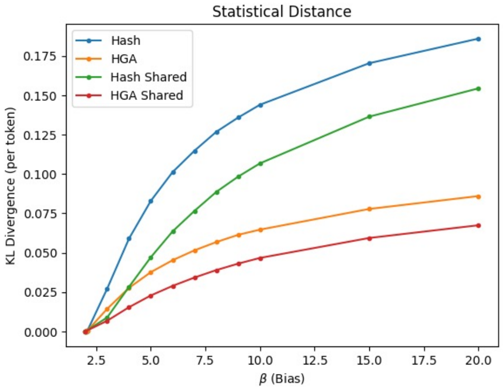

Figure 9: KL divergence, Shared Quantized LLM

Figures 8 and 9 show the results of the procedure described in the previous paragraph that leverages knowledge of the quantized LLM at both the encoder and decoder. Consistent with our previous results, HGA exhibits the highest performance both in terms of rate and KL divergence. For all levels of β tested, the use of the quantized LLM at encoding also reduced the KL divergence. Both methods converge to within 0.05 of the symmetric capacity of Hash (see Appendix for details of this notion).

6 Conclusion

We studied large language-model based steganography with the goal of maximizing the amount of information that can be hidden in LLM generated text. We studied the fundamental limits of this problem and proposed simple techniques that are close to optimal, including in challenging settings where, when decoding, we do not have access to the decoder, or only to a quantized version.

7 Limitations

One limitation we face is that our technique depends on the availability of good error-correcting codes with appropriate lengths.

References

Ldpc. https://github.com/xdsopl/LDPC.

Miranda Christ, Sam Gunn, and Or Zamir. 2023. Undetectable watermarks for language models. Cryptology ePrint Archive, Paper 2023/763. https://eprint.iacr.org/2023/763.

Thomas M. Cover and Joy A. Thomas. 2006. *Elements of Information Theory 2nd Edition (Wiley Series in Telecommunications and Signal Processing)*. Wiley-Interscience.

Christian Schroeder de Witt, Samuel Sokota, J. Zico Kolter, Jakob Foerster, and Martin Strohmeier. 2023. Perfectly secure steganography using minimum entropy coupling. *Preprint*, arXiv:2210.14889.

R. Gallager. 1962. Low-density parity-check codes. *IRE Transactions on Information Theory*, 8(1).

John Kirchenbauer, Jonas Geiping, Yuxin Wen, Jonathan Katz, Ian Miers, and Tom Goldstein. 2024. A watermark for large language models. *Preprint*, arXiv:2301.10226.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Jiaxuan Wu, Zhengxian Wu, Yiming Xue, Juan Wen, and Wanli Peng. 2024. Generative text steganography with large language model. *Preprint*, arXiv:2404.10229.

M.K. Yadav and K.K. Parhi. 2005. Design and implementation of ldpc codes for dvb-s2. In *Conference Record of the Thirty-Ninth Asilomar Conference onSignals, Systems and Computers, 2005.*

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 T. 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. *ArXiv*, abs/2205.01068.

Xuandong Zhao, Prabhanjan Vijendra Ananth, Lei Li, and Yu-Xiang Wang. 2023. Provable robust watermarking for ai-generated text. *ArXiv*, abs/2306.17439.

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 Analysis

For shorthand, let $\hat{\boldsymbol{p}}^{(t)} = \widehat{f}\left(\boldsymbol{x}_p, \boldsymbol{x}_{t-1}, \boldsymbol{u}\right)$ and $\boldsymbol{p}^{(t)} = f\left(\boldsymbol{x}_p, \boldsymbol{x}_{t-1}\right)$. Let $\mathcal{C}$ be the code from (2) from the encoding process and let C_t be a random variable that represents the value of the t-th bit of a codeword $\boldsymbol{c} \in \mathcal{C}$. Let M_t be a random variable that represents the largest partition at time t. In particular, $M_t = 0$ if $P_0^{(t)} = P_{\max}^{(t)}$. We assume throughout this section that $\Pr\left(C_t = 0\right) = \frac{1}{2}$, $\Pr\left(M_t = 0\right) = \frac{1}{2}$ and that M_t and C_t are independent.

Lemma 1. *The KL-divergence between the outputs of a biased LLM whose tokens are sampled according to (2) and an unbiased LLM at time t is*

$$\begin{aligned} &D\left(\boldsymbol{p}^{(t)} || \hat{\boldsymbol{p}}^{(t)}\right) = \\ &- P_0^{(t)} \log\left(\frac{\lfloor \beta P_0^{(t)} \rfloor + 1 - \lfloor \beta(1 - P_0^{(t)}) \rfloor}{2P_0^{(t)}}\right) \\ &- (1 - P_0^{(t)}) \times \\ &\quad \log\left(\frac{1 - \lfloor \beta P_0^{(t)} \rfloor + \lfloor \beta(1 - P_0^{(t)}) \rfloor}{2(1 - P_0^{(t)})}\right). \end{aligned}$$

Furthermore, if $\beta = 2$, then $D\left(\boldsymbol{p}^{(t)} || \hat{\boldsymbol{p}}^{(t)}\right) = 0$.

Next, we consider the maximum achievable rate of (N, m, β)-stego codes. Our bound, which appears in Theorem 1, depends on the underlying code $\mathcal{C}$ employed by (2) during the encoding, the partition functions used, along with the bias β. In the expressions and discussions that follow we assume H is the binary entropy function and it equal to zero outside the range of $(0, 1)$. Let $F_t : [V] \to \{0, 1\}$ be a function which at time t provides the logical partition for the token provided as input belongs to. Similarly, let $G_t : [V] \to \{0, 1\}$ be the partition function for the decoder. For any $k_0 \in \mathcal{P}_0^{(t)}$, $k_1 \in \mathcal{P}_1^{(t)}$, $t \in [N]$, we assume $\mu = \Pr\left(F_t(k_0) \neq G_t(k_0)\right) = \Pr\left(F_t(k_1) \neq G_t(k_1)\right)$.

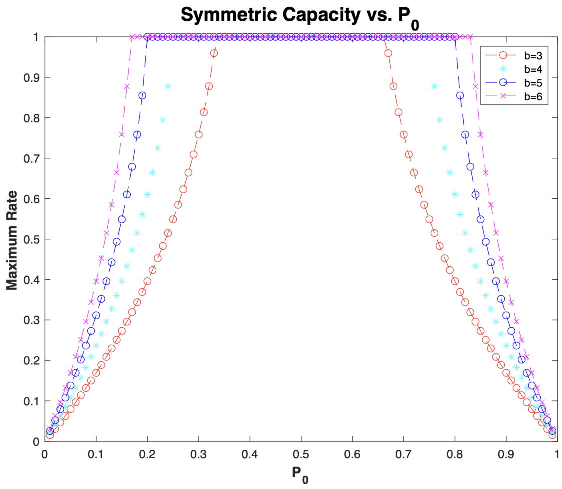

Figure 10: Symmetric capacity from 1.

Theorem 1. *As $N \to \infty$, the maximum rate of an (N, m, β)-stego code is:*

$$\begin{aligned} R_S(P^{(t)}_{\max}, \beta, \mu) &= \frac{1}{N}\sum_{t=1}^{N} H\Bigg(\frac{1}{2} + \Big(\lfloor \beta P^{(t)}_{\max} \rfloor \\ &\quad - \lfloor \beta(1 - P^{(t)}_{\max} \rfloor \left(\frac{1-\mu}{2}\right) + \big(\lfloor \beta(1 - P^{(t)}_{\max} \rfloor \\ &\quad - \lfloor \beta P^{(t)}_{\max} \rfloor\big)\frac{\mu}{2}\Bigg) \\ &\quad - \frac{1}{2} H\left(\lfloor \beta P^{(t)}_{\max} \rfloor (1-\mu) + \left(1 - \lfloor \beta P^{(t)}_{\max} \rfloor\right)\mu\right) \\ &\quad - \frac{1}{2} H\big(\lfloor \beta(1 - P^{(t)}_{\max} \rfloor (1-\mu) \\ &\quad + \left(1 - \lfloor \beta(1 - P^{(t)}_{\max} \rfloor\right)\mu\big). \end{aligned}$$

For shorthand, we will refer to the quantity in Theorem 1 as the ***symmetric capacity*** of the channel. In order to better understand this quantity, we have plotted the function $NR_S\left(P_{\max}, \beta, \mu\right)$ as a function of $P_{\max}$ Figure 2 for $\mu = 0$.

Similar to the result regarding the KL-divergence, the symmetric capacity is symmetric about the point $P_0 = \frac{1}{2}$, and we therefore plot the maximum rate as a function of $P_{\max}$.

Next we turn our attention to the task of obtaining upper bounds on the achievable rate of an (N, m, β)-stego code. Here, we will focus on the Decoder Uninformed (DU) setting since a lower bound for the DU setting will trivially also hold for the DI case as well. One of the challenges to developing such bounds (and subsequent efficient coding schemes) on the set of achievable rates stems from the fact that the partitioning itself is a difficult problem and the assignment of tokens to logical values itself can depend on the probabilities of each of the tokens at each instant in time.

In order to circumvent some of these difficulties, we will use two simple ideas: First, we employ a pseudo-random function to assign each token with equal probability to be in one of the two partitions. The goal here is to reduce the need for higher bias parameters in certain settings. Second, we make use of error-correcting codes to address the setup where there are some tokens that require a bias value that is beyond the acceptable range due to our security property. In such cases, it may be highly likely (despite the fact that we bias) that the incorrect token is still sampled.

Our next result shows that when the variance of the generated token distributions is low enough, the approach from the previous paragraph is still capable of producing high rate codes. For shorthand, we say that a categorical distribution $\boldsymbol{p}$ has square magnitude s if the cross-product of $\boldsymbol{p}$ with itself is equal to s.

Lemma 2. *Suppose that for a fraction of $1 - \epsilon$ tokens, the output distribution of f from (1) has square magnitude at most V_p where $V_p < \alpha^2\left(1 - \frac{2(1-\alpha)}{\beta}\right)^2$. Then, given access to a pseudo-random function $g^{(t)} : \mathcal{V} \to \{0, 1\}$, there exists an efficient scheme that achieves a rate $1 - H\left(\zeta\right)$, for N large enough, where*

$$\zeta = \epsilon + (1-\epsilon)\left(\alpha + \alpha^2\right),$$

provided $\zeta < \frac{1}{2}$.

One of the attractive properties of the previous lemma is that we do not require knowledge of $P^{(t)}_{\max}$, and instead only require a bound on the square magnitude of the underlying categorical distributions. Unfortunately, the fact that the bound is not written in terms of $P^{(t)}_{\max}$ also makes difficult to compare the previous lemma with Theorem 1 since Theorem 1 requires knowledge of $P^{(t)}_{\max}$ for each time instance. In order to develop a lower bound that is more comparable to Lemma 1, for a sequence of length N, suppose that $Pr(\Pi = p)$ is $\frac{\left|t : P^{(t)}_{\min} = p\right|}{N}$. Then, from Theorem 1, we have

$$\int_0^1 \Pr\left(\Pi = x\right) NR_S(1-x, \beta, \mu)\, dx. \tag{4}$$

For the lower bound, we note that if the partitions $\mathcal{P}_{\min}, \mathcal{P}_{\max}$ are the result of using an invertible hash, then using similar logic as in the proof of Lemma 2, errors can occur during decoding only if $P_{\min}\beta < 1$. Then, we can achieve a rate of at least

$$1 - H\left(\text{Pr}\left(\Pi < \frac{1}{\beta}\right)\right). \tag{5}$$

by treating each token where $P_{\min}\beta < 1$ as a random error under the binary symmetric channel (Cover and Thomas, 2006). In the next section, we investigate one of the key details of our proposed scheme, which is the design of the partitioning function.