

A High-Capacity and Secure Disambiguation Algorithm for Neural Linguistic Steganography

Yapei Feng¹, Feng Jiang¹, Shanhao Wu², and Hua Zhong¹

Abstract—Neural linguistic steganography aims to embed information into natural text while preserving statistical undetectability. A fundamental challenge in this field stems from tokenization ambiguity in modern tokenizers, which can lead to catastrophic decoding failures. The recent method, SyncPool, addresses this ambiguity by employing a coarse-grained synchronization mechanism over groups of ambiguous candidates. However, SyncPool sacrifices embedding capacity, as it utilizes the entire Shannon entropy of an ambiguous group solely for synchronization rather than for payload embedding. We propose a method named look-ahead Sync, which overcomes the capacity limitation of SyncPool while retaining its provable security guarantees. Our approach performs minimal synchronized sampling only on truly indistinguishable token sequences, while strategically preserving all other discernible paths to maximize embedding capacity. We provide theoretical proofs for the security of our method and analyze the gap between its achievable embedding capacity and the theoretical upper bound. Experiments on English (using Llama 3) and Chinese (using Qwen 2.5) benchmarks show that our method consistently approaches the theoretical capacity upper bound and significantly outperforms SyncPool. The improvement in embedding rate exceeds 160% in English and 25% in Chinese, particularly in settings with larger candidate pools. This work represents a significant step toward practical high-capacity provably secure linguistic steganography.

Index Terms—linguistic steganography, provably secure steganography, zero KL divergence, tokenization ambiguity, embedding capacity

I. INTRODUCTION

LINGUISTIC steganography hides data in ordinary text to enable covert communication while concealing the existence of the message. The effectiveness of such systems is assessed along two often competing objectives, embedding capacity and statistical security [1]–[3]. Capacity, measured in bits per token (BPT), quantifies how much secret information a coartext can carry and is critical for practical utility [4], [5]. Security measures resistance to detection. [6]–[9] The strongest standard is zero Kullback–Leibler (KL) divergence, which requires the distribution of stegotext to equal that of coartext [10]. Under this standard, no statistical test of any power can distinguish the two [10]. The central challenge is therefore to maximize capacity without relaxing this criterion [11].

The advent of large language models (LLMs) provides fluent coartext [12]–[14], yet their subword tokenization schemes, such as Byte Pair Encoding (BPE) [15], [16], introduce the

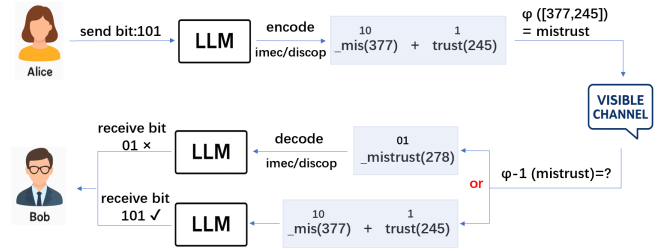


Fig. 1. A toy example of tokenization ambiguity. The detokenizer φ is not injective, so identical text (mistrust) can correspond to [278] or [377, 245]. If bits are embedded at the token level without resolving this ambiguity, the receiver may decode the wrong bits from the same visible string.

critical challenge of tokenization ambiguity, where a single visible string can correspond to multiple token sequences [17]. For example, the string mistrust may be tokenized as [_mistrust] or as [_mis, _trust] (see Fig. 1). In autoregressive generation, if the receiver reconstructs a different token path than the sender, the conditional distributions for subsequent steps become desynchronized, making the remaining payload unrecoverable and causing catastrophic decoding failure [18].

Addressing this ambiguity requires a dedicated disambiguation module. Early generative approaches by Nozaki and Murawaki [17] enforced a unique decoding path by pruning any token that is a prefix of another candidate. Although this guarantees decodability, pruning irreversibly alters the model’s distribution. Variants based on a maximum-weight independent set (MWIS) [19] can reduce but not eliminate this deviation and therefore do not satisfy the zero-KL requirement [10], [11], [18]. A recent breakthrough, SyncPool [18], achieves provable security by replacing pruning with synchronization. However, its coarse-grained intra-pool synchronization imposes a high cost. It systematically discards the Shannon entropy of non-selected candidates, which substantially reduces embedding capacity. The loss grows as candidate pools become larger, and this severely limits practicality for short-form covert communication [18], [20].

To address this trade-off between security and capacity, we introduce Look-ahead Sync, a recursive disambiguation algorithm that remains within the synchronization paradigm while recovering capacity. Instead of synchronizing an entire ambiguous group, Look-ahead Sync performs a minimal synchronized sample only over sequences that are indistinguishable to the receiver at the current visible step. The algorithm preserves all other discernible paths so that their entropy remains available for subsequent embedding. Our contributions are threefold:

¹The authors are with the School of Cyberspace, Hangzhou Dianzi University, Hangzhou 310018, P. R. China (email: fengyapei@hdu.edu.cn).

²Shanhao Wu is with the Bridge and Wind Engineering Laboratory, Department of Civil Engineering, School of Engineering, University of Tokyo, Tokyo 113-8656, Japan.

- (1) We design Look-ahead Sync, a recursive disambiguation algorithm, and prove its computational zero-KL security.
- (2) We derive the theoretical capacity upper bound for zero-KL disambiguation and analyze the sources of gap between this bound and Look-ahead Sync.
- (3) Using modern large language models, we have demonstrated that Look-ahead Sync consistently approaches the capacity upper bound and substantially outperforms state-of-the-art baselines in embedding capacity.

II. BACKGROUND

This section establishes the formal framework for our work by covering three key areas. First, we formalize the security standards that govern linguistic steganography. Second, we provide an in-depth account of the system architecture prevalent in modern ambiguity-aware methods. Finally, we review prior disambiguation architectures to situate our contribution within the current state of the art.

A. Security Definitions in Steganography

The foundational goal of linguistic steganography is to remain invisible to a watchful adversary. This adversarial setting is classically modeled by Simmons’ *Prisoners’ Problem*, where two parties must communicate covertly under the surveillance of a warden [21], [22]. In this scenario, every transmission attempted by the prisoners is inspected. For each observed message, the warden must decide whether it is an ordinary letter or a coded note carrying concealed information. From a formal perspective, this is equivalent to a hypothesis test between the coverttext distribution, P_T , and the stegotext distribution, P_s . A steganographic system is considered secure if it can render these two distributions statistically indistinguishable, and this requirement is formalized by two primary standards of security.

a) Information-Theoretic Security: This standard represents the most stringent security guarantee, mandating that the stegotext and coverttext distributions be mathematically identical. This is quantified by requiring the Kullback–Leibler (KL) divergence between them to be exactly zero [10]:

$$D_{\text{KL}}(P_s \parallel P_T) = \sum_{x \in \mathcal{X}} P_s(x) \log \frac{P_s(x)}{P_T(x)} = 0, \quad (1)$$

where $\mathcal{X}$ is the set of all possible (terminal) messages, P_T denotes the distribution of genuine coverttext, and P_s denotes the distribution of stegotext induced by the stegosystem under the same conditions. A system satisfying this condition achieves perfect security, as no adversary, regardless of their computational power, can gain an advantage in the decision problem.

b) Computational Security: This standard offers a practical and rigorous guarantee for systems that employ cryptographic primitives. It defines security in the context of an adversary restricted to probabilistic polynomial-time (PPT) computations [11]. A system is considered computationally secure if the advantage any such adversary has in distinguishing between P_s and P_T is negligible, denoted as $\text{negl}(\kappa)$ for a security parameter κ :

$$|\Pr[\mathcal{A}(s) = 1] - \Pr[\mathcal{A}(t) = 1]| < \text{negl}(\kappa), \quad (2)$$

where $s \sim P_s$, $t \sim P_T$, and $\mathcal{A}$ is any PPT adversary. Therefore, a primary objective for provably secure steganography in practice is to achieve a computational zero-KL guarantee, which ensures that the perfect security property of Eq. (1) holds against any computationally bounded adversary.

B. Secure Steganography in Autoregressive Models

The widespread adoption of large language models (LLMs) has revolutionized linguistic steganography. By generating fluent and contextually coherent text, these models provide an ideal source of coverttext [12]–[14], [23] that can mimic human writing. [24], [25] The core embedding principle is a process known as message-driven sampling. In this process, an entropy encoder uses the model’s output probability distribution and the secret bitstream at each step to select the next token, continuing autoregressively [3]–[5].

However, this steganographic paradigm is complicated by the practical necessity of tokenization. For transmission, the sender must *detokenize* the generated token sequence into a human-readable string. The receiver, in turn, must *retokenize* this string to recover the underlying message. Due to the nature of subword tokenizers, however, this retokenization is not guaranteed to reproduce the sender’s original sequence. This potential for discrepancy is the core of the tokenization ambiguity problem [17]. Any such desynchronization corrupts the conditional probabilities for all subsequent steps, leading to a catastrophic decoding failure [18].

To manage this challenge, modern ambiguity-aware systems adopt the modular pipeline depicted in Figure 2. A single generation step is decomposed into three distinct stages:

- (i) **LLM Generation.** The base language model takes the current context and produces a raw, and potentially ambiguous, probability distribution over its vocabulary.
- (ii) **Disambiguation Module.** A dedicated module transforms this raw distribution into a new, well-defined distribution over a set of unambiguous candidate choices, resolving all token-level conflicts before encoding.
- (iii) **Secure Encoding Module.** The entropy coder, guided by the secret message, samples a choice from the structured distribution provided by the disambiguation module to select the token(s) for the current step.

For the end-to-end system to achieve zero-KL security, both the Secure Encoding and Disambiguation modules must independently be distribution-preserving. The Secure Encoding module must function as a perfect sampler, drawing from its input distribution without statistical bias. This challenge has been effectively addressed by state-of-the-art entropy coders such as iMEC and Discop [2], [3]. The Disambiguation Module faces the more recent challenge of structuring the choices so that the resulting probability space remains mathematically equivalent to the original. Methods such as SyncPool have shown that this is achievable [18], setting the stage for further architectural improvements.

C. Prior Disambiguation Architectures

Existing designs for the disambiguation module can be categorized into two competing architectures.

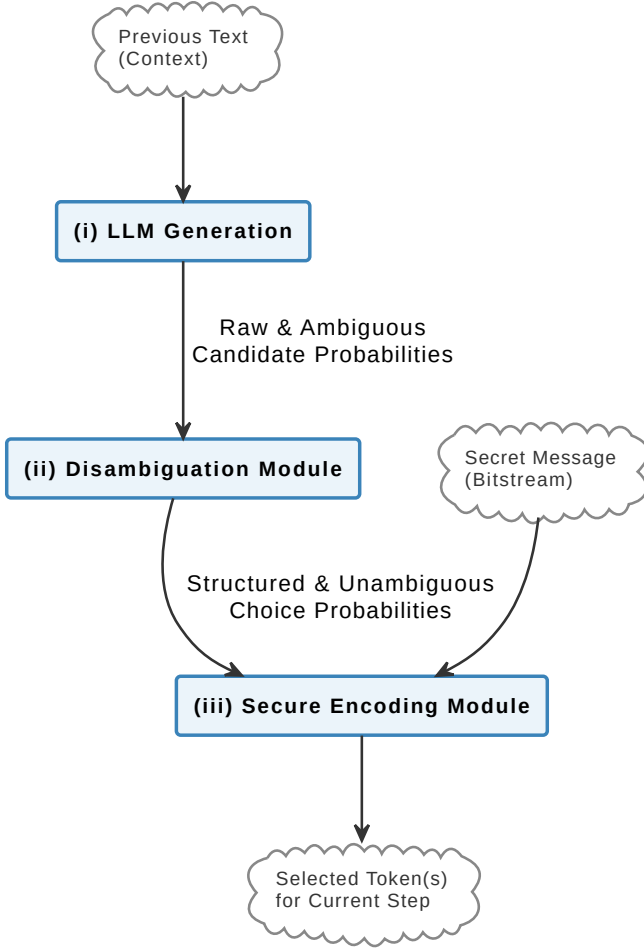


Fig. 2. A schematic of the modular pipeline for a single generation step in an ambiguity-aware steganographic system, where the Disambiguation Module transforms the LLM’s raw output into a structured set of choices for secure encoding.

a) Distribution-Altering Architectures: This approach, first proposed for generative models by Nozaki and Murawaki [17], resolves ambiguity by directly *pruning* the candidate set. The mechanism removes any token that serves as a prefix for another candidate. Although this ensures decodability, it fundamentally alters the probability distribution. More sophisticated variants that employ a maximum-weight independent set (MWIS) to minimize the probability mass of the pruned tokens [19] suffer from the same inherent flaw, since the act of deleting valid candidates creates a nonzero KL divergence and renders the entire architectural class insecure by definition. [1].

b) Distribution-Preserving Architectures: This architecture was introduced by SyncPool [18], the first module to achieve computational zero-KL security. The design involves a two-stage process. First, it groups all candidates with prefix relationships into ambiguity pools (e.g., $\{_B, _BB, _BBD\}$). The encoder module embeds the payload by selecting one of these pools, which is an unambiguous choice. Second, to resolve the selection *within* the chosen pool, the module employs a non-payload-bearing synchronized sampler. This

sampler uses a cryptographically secure pseudorandom number generator (CSPRNG), seeded with a shared secret key, to choose a representative token according to its original probability distribution. While provably secure, this design consumes the Shannon entropy of the intra-pool selection for synchronization rather than for payload. This architectural inefficiency establishes the central technical challenge our work aims to solve.

This trade-off between provable security and embedding capacity defines the critical frontier for modern linguistic steganography. Addressing this architectural inefficiency is the primary motivation for our work. In the subsequent sections, we introduce Look-ahead Sync, a disambiguation algorithm designed to retain the rigorous security guarantees of the synchronization-based paradigm while systematically recovering the capacity lost by existing methods.

III. THE LOOK-AHEAD SYNC ALGORITHM

The primary challenge for current provably secure steganography is resolving tokenization ambiguity. Existing secure methods, while avoiding statistical alterations, achieve this by aggressively discarding Shannon entropy, which leads to sub-optimal embedding capacity. To overcome this limitation, we present Look-ahead Sync, an algorithm designed to maximize capacity while upholding strict security guarantees. The core is a look-ahead resolution strategy that resolves only the necessary ambiguous cases, thereby preserving the information capacity of other distinguishable paths for subsequent embedding steps.

This strategy is implemented within an iterative, state-driven architecture, as depicted in Figure 3. The loop begins from the first set of candidates produced by the base model under a prompt shared between the sender and the receiver. In each iteration, the algorithm embeds part of the secret bit and prepares the next round by executing three phases:

- **Candidate Partitioning.** Group the model’s candidate continuations by shared visible prefixes to obtain pools that are unambiguously distinguishable from one another, and aggregate within-pool probability mass to form an inter-group distribution.
- **Inter-Group Entropy Coding.** Embed a portion of the secret by selecting exactly one pool according to the inter-group distribution at the sender, and recover the same bits by identifying the selected visible prefix at the receiver. This is the only payload-bearing step.
- **Look-ahead Resolution.** Resolve any remaining ambiguity inside the selected pool by synchronizing only among truly indistinguishable sequences and performing a minimal look-ahead expansion, while preserving all already discernible continuations so their information remains available for future embedding.

We next detail these three operations and then present an end-to-end description of how they compose at the sender and the receiver.

A. Candidate Partitioning

The first operation in iteration t partitions the global candidate set S_t to prepare it for secure payload embedding.

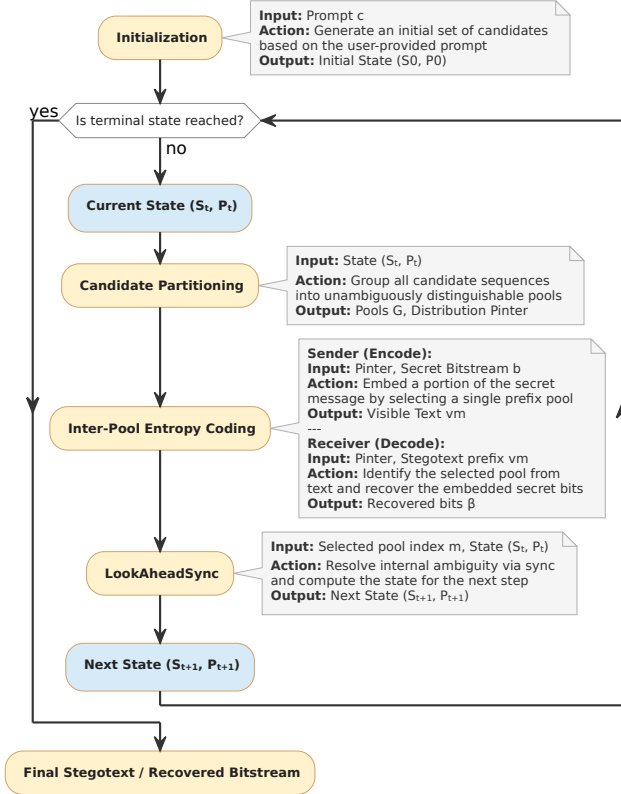


Fig. 3. The iterative architecture of Look-ahead Sync. The process begins with initialization and then enters a main loop that repeatedly executes three steps, namely partitioning candidates, embedding payload, and resolving ambiguity via a look-ahead mechanism to compute the next state. The loop continues until a terminal state is reached.

All candidate sequences are organized into disjoint groups according to their visible string prefixes. This partition ensures that each group is unambiguously distinguishable from the others, which is required by the subsequent encoding stage.

The mechanism proceeds in two steps. First, the algorithm applies the detokenization function $\varphi(\cdot)$ to each candidate sequence in S_t to obtain its visible string representation. Second, it groups the original candidate sequences based on these strings. We form groups keyed by a visible string v and assign to that group every candidate whose visible string starts with v (if no other item shares the prefix, the group has one element). To enable a single linear pass, we sort candidates by their visible strings so that a shorter prefix appears before any longer string that starts with it; this guarantees that all items sharing the same prefix are contiguous.

Formally, the outcome is a partition of S_t into M disjoint groups, denoted by

$$\mathcal{S}^{\text{inter}} := \{\mathcal{S}_0^{\text{intra}}, \mathcal{S}_1^{\text{intra}}, \dots, \mathcal{S}_{M-1}^{\text{intra}}\}. \quad (3)$$

With the candidates organized into these groups, we define the inter-group probability distribution P^{inter} , where each component p_m^{inter} is the total probability mass of group $\mathcal{S}_m^{\text{intra}}$:

$$p_m^{\text{inter}} := \sum_{s \in \mathcal{S}_m^{\text{intra}}} P_t(s), \quad (4)$$

and collect these components, in the same order as the groups, into the vector

$$P^{\text{inter}} := (p_0^{\text{inter}}, p_1^{\text{inter}}, \dots, p_{M-1}^{\text{inter}}). \quad (5)$$

Since $\{\mathcal{S}_m^{\text{intra}}\}_{m=0}^{M-1}$ is a partition of S_t and P_t is a probability distribution over S_t , we have $\sum_{m=0}^{M-1} p_m^{\text{inter}} = 1$. For decoding, we also record the ordered list of group keys (visible prefixes), denoted $V = [v_0, \dots, v_{M-1}]$.

Algorithm 1 PARTITIONBYPREFIX(S_t, P_t)

Require: Candidate sequence set S_t ; probability mapping P_t

Ensure: Collection of disjoint groups $\mathcal{S}^{\text{inter}}$; inter-group distribution P^{inter} ; list of group keys V

```

1: Sort  $S_t$  (and carry  $P_t$  along) by the visible strings  $\varphi(s)$  so that any shorter
   prefix appears before longer strings starting with it.
2:  $\mathcal{S}^{\text{inter}} \leftarrow \emptyset$ ;  $P^{\text{inter}} \leftarrow \emptyset$ ;  $V \leftarrow \emptyset$ 
3: if  $S_t$  is not empty then
4:    $v_{\text{prefix}} \leftarrow \varphi(S_t[0])$  ▷ current group key
5:    $\mathcal{S}_{\text{current}}^{\text{intra}} \leftarrow \{S_t[0]\}$ 
6:   for  $i \leftarrow 1$  to  $|S_t| - 1$  do
7:     if  $\varphi(S_t[i])$  starts with  $v_{\text{prefix}}$  then
8:       Add  $S_t[i]$  to  $\mathcal{S}_{\text{current}}^{\text{intra}}$ 
9:     else
10:      Append  $\mathcal{S}_{\text{current}}^{\text{intra}}$  to  $\mathcal{S}^{\text{inter}}$ 
11:      Append  $v_{\text{prefix}}$  to  $V$ 
12:       $\mathcal{S}_{\text{current}}^{\text{intra}} \leftarrow \{S_t[i]\}$  ▷ start new group
13:       $v_{\text{prefix}} \leftarrow \varphi(S_t[i])$ 
14:    end if
15:  end for
16:  Append  $\mathcal{S}_{\text{current}}^{\text{intra}}$  to  $\mathcal{S}^{\text{inter}}$ 
17:  Append  $v_{\text{prefix}}$  to  $V$ 
18: end if
19: for each group  $\mathcal{S}_m^{\text{intra}}$  in  $\mathcal{S}^{\text{inter}}$  do
20:    $p_m \leftarrow \sum_{s \in \mathcal{S}_m^{\text{intra}}} P_t(s)$ 
21:   Append  $p_m$  to  $P^{\text{inter}}$ 
22: end for
23: return ( $\mathcal{S}^{\text{inter}}, P^{\text{inter}}, V$ )

```

B. Inter-Group Entropy Coding

With the candidate space partitioned into $\mathcal{S}^{\text{inter}}$, the second operation embeds a segment of the secret payload by selecting one group. This is performed with a state-of-the-art entropy encoder, such as iMEC [3] or Discop [2], which we denote abstractly by $\mathcal{E}$. The sender applies $\mathcal{E}$ to the inter-group distribution P^{inter} and the secret bitstream b , yielding the selected group's index m and the consumed bits β . Symmetrically, the receiver identifies the index m from the visible string and applies the inverse function $\mathcal{E}^{-1}$ to recover β :

$$(m, \beta) \leftarrow \mathcal{E}(P^{\text{inter}}, b), \quad (6)$$

$$\beta := \mathcal{E}^{-1}(P^{\text{inter}}, m). \quad (7)$$

Selecting group m identifies the unique group key v_m that is a prefix of the final stegotext T . The decoder finds v_m as a prefix of the T , uses Eq. (7) to recover β .

While selecting $\mathcal{S}_m^{\text{intra}}$ completes the payload-bearing choice for this step, the ambiguity *within* the chosen group remains unresolved. To prepare this group for the final resolution stage, its sub-distribution must be normalized so that subsequent choices are probabilistically sound. The inputs to the next stage are the chosen intra-group candidate set $\mathcal{S}_m^{\text{intra}}$ and its normalized distribution P_m^{intra} :

$$P_m^{\text{intra}} := \text{normalize}(\{P_t(s) \mid s \in \mathcal{S}_m^{\text{intra}}\}), \quad (8)$$

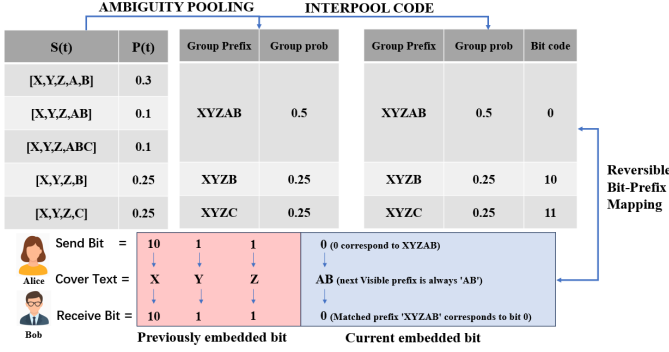


Fig. 4. Inter-group encoding. Candidate token sequences are grouped by a common visible prefix and their probabilities are aggregated. An entropy encoder then maps a segment of the secret bitstream to a unique group according to the aggregated probabilities.

which is equivalently given by

$$P_m^{\text{intra}}(s) = \frac{P_t(s)}{p_m^{\text{inter}}} \quad \text{for all } s \in \mathcal{S}_m^{\text{intra}}, \quad (9)$$

where p_m^{inter} is defined in Eq. (4).

Algorithm 2 LOOKAHEAD

Require: Intra-group set $\mathcal{S}_m^{\text{intra}}$; normalized distribution P_m^{intra} ; shared key K ; language model LLM

Ensure: Next-round state (S_{t+1}, P_{t+1}) or a terminal sequence s_{end}

```

1:  $v_m \leftarrow \text{VISIBLEPREFIX}(\mathcal{S}_m^{\text{intra}})$ 
2:  $\mathcal{S}_{\text{prefix}} \leftarrow \{s \in \mathcal{S}_m^{\text{intra}} \mid \varphi(s) = v_m\}$ 
3:  $\mathcal{S}_{\text{partial}} \leftarrow \mathcal{S}_m^{\text{intra}} \setminus \mathcal{S}_{\text{prefix}}$ 
4:  $P_{\text{prefix}} \leftarrow \text{normalize}(\{P_m^{\text{intra}}(s) : s \in \mathcal{S}_{\text{prefix}}\})$ 
5:  $s_{\text{sync}} \leftarrow \text{SyncSample}(\mathcal{S}_{\text{prefix}}, P_{\text{prefix}}, K)$ 
6: if  $\text{ISEOS}(s_{\text{sync}})$  and  $\mathcal{S}_{\text{partial}} = \emptyset$  then
7:   return  $\text{END}(s_{\text{sync}})$   $\triangleright$  terminate; wrapper for IsEnd/EndSeq
8: end if
9: if  $\text{ISEOS}(s_{\text{sync}})$  then
10:   $\mathcal{A}_{\text{next}} \leftarrow \emptyset$ ;  $P_{\text{next}} \leftarrow \emptyset$ 
11: else
12:   $(\mathcal{A}_{\text{next}}, P_{\text{next}}) \leftarrow \text{LLM}(\cdot \mid s_{\text{sync}})$ 
13: end if
14:  $p_{\text{sum}} \leftarrow \sum_{s \in \mathcal{S}_{\text{prefix}}} P_m^{\text{intra}}(s)$ 
15:  $S_{t+1} \leftarrow \mathcal{S}_{\text{partial}}$ 
16: for all  $x \in \mathcal{A}_{\text{next}}$  do
17:   $S_{t+1} \leftarrow S_{t+1} \cup \{s_{\text{sync}} \oplus x\}$ 
18: end for
19: Define  $P_{t+1}$  as a mapping as follows:
20: for all  $s \in \mathcal{S}_{\text{partial}}$  do
21:   $P_{t+1}(s) \leftarrow P_m^{\text{intra}}(s)$ 
22: end for
23: for all  $x \in \mathcal{A}_{\text{next}}$  do
24:   $P_{t+1}(s_{\text{sync}} \oplus x) \leftarrow p_{\text{sum}} \cdot P_{\text{next}}(x)$ 
25: end for
26: return  $(S_{t+1}, P_{t+1})$ 

```

C. Look-ahead Resolution

While the selection of a group in the previous step is an unambiguous, payload-bearing choice, the ambiguity *within* the chosen group $\mathcal{S}_m^{\text{intra}}$ remains unresolved. All candidate sequences in this set share the same visible prefix, which makes them indistinguishable to the receiver at the current step. This indistinguishability blocks further secure embedding and must be resolved before the next round. The objective in this operation is to perform a *distribution-preserving transformation* of the candidate space, converting the ambiguous input state

$(\mathcal{S}_m^{\text{intra}}, P_m^{\text{intra}})$ into a new, unambiguous state (S_{t+1}, P_{t+1}) on which the next payload-bearing decision can safely proceed.

We address the ambiguity at its source. Let v_m denote the common visible prefix of all sequences in $\mathcal{S}_m^{\text{intra}}$. The indistinguishability arises precisely from those sequences that decode *exactly* to v_m . The look-ahead mechanism resolves this by generating new, distinct continuations from that source, thereby separating paths that were previously identical at the visible level while preserving probability mass.

Formally, the algorithm first partitions $\mathcal{S}_m^{\text{intra}}$ into two functionally distinct subsets. The *Prefix Set* collects the exact-prefix items that decode to v_m and will serve as the input context for look-ahead; the *Partial Set* contains longer continuations that are already distinguishable and will be preserved to the next round:

$$\mathcal{S}_{\text{prefix}} := \{s \in \mathcal{S}_m^{\text{intra}} \mid \varphi(s) = v_m\}, \quad (10)$$

$$\mathcal{S}_{\text{partial}} := \mathcal{S}_m^{\text{intra}} \setminus \mathcal{S}_{\text{prefix}}. \quad (11)$$

To remain faithful to the original distribution while avoiding a combinatorial explosion, we select a *single* synchronized representative from the Prefix Set using a non-payload-bearing sampler. Concretely, we normalize the intra-set probabilities to obtain a valid sampling distribution P_{prefix} and then draw s_{sync} with a CSPRNG-seeded synchronized procedure:

$$P_{\text{prefix}} := \text{normalize}(\{P_m^{\text{intra}}(s) \mid s \in \mathcal{S}_{\text{prefix}}\}), \quad (12)$$

$$s_{\text{sync}} := \text{SyncSample}(\mathcal{S}_{\text{prefix}}, P_{\text{prefix}}, K). \quad (13)$$

where $\text{SyncSample}(\mathcal{S}, P, K)$ denotes a shared, distribution-preserving sampler driven by a CSPRNG initialized with the shared key K .

With s_{sync} fixed, the algorithm performs a single deterministic forward pass of the base model conditioned on this representative. This yields a set of next-token candidates and their conditional probabilities:

$$(\mathcal{A}_{\text{next}}, P_{\text{next}}) := \text{LLM}(\cdot \mid s_{\text{sync}}), \quad (14)$$

where P_{next} is a probability distribution over $x \in \mathcal{A}_{\text{next}}$ and satisfies $\sum_{x \in \mathcal{A}_{\text{next}}} P_{\text{next}}(x) = 1$.

We then merge the preserved partials with the freshly expanded continuations to construct the next state. Let

$$p_{\text{sum}} := \sum_{s \in \mathcal{S}_{\text{prefix}}} P_m^{\text{intra}}(s), \quad (15)$$

be the total probability mass of the Prefix Set under P_m^{intra} . The next candidate set concatenates each $x \in \mathcal{A}_{\text{next}}$ to the synchronized representative and unions the result with the Partial Set:

$$S_{t+1} := \mathcal{S}_{\text{partial}} \cup \{s_{\text{sync}} \oplus x \mid x \in \mathcal{A}_{\text{next}}\}, \quad (16)$$

where $\oplus$ denotes sequence concatenation. The corresponding probabilities follow the law of total probability. We carry over masses for preserved partials and reallocate the entire Prefix-Set mass onto the new continuations by scaling P_{next} :

$$P_{t+1}(s) := \begin{cases} P_m^{\text{intra}}(s), & \text{if } s \in \mathcal{S}_{\text{partial}}, \\ p_{\text{sum}} \cdot P_{\text{next}}(x), & \text{if } s = s_{\text{sync}} \oplus x, x \in \mathcal{A}_{\text{next}}. \end{cases} \quad (17)$$

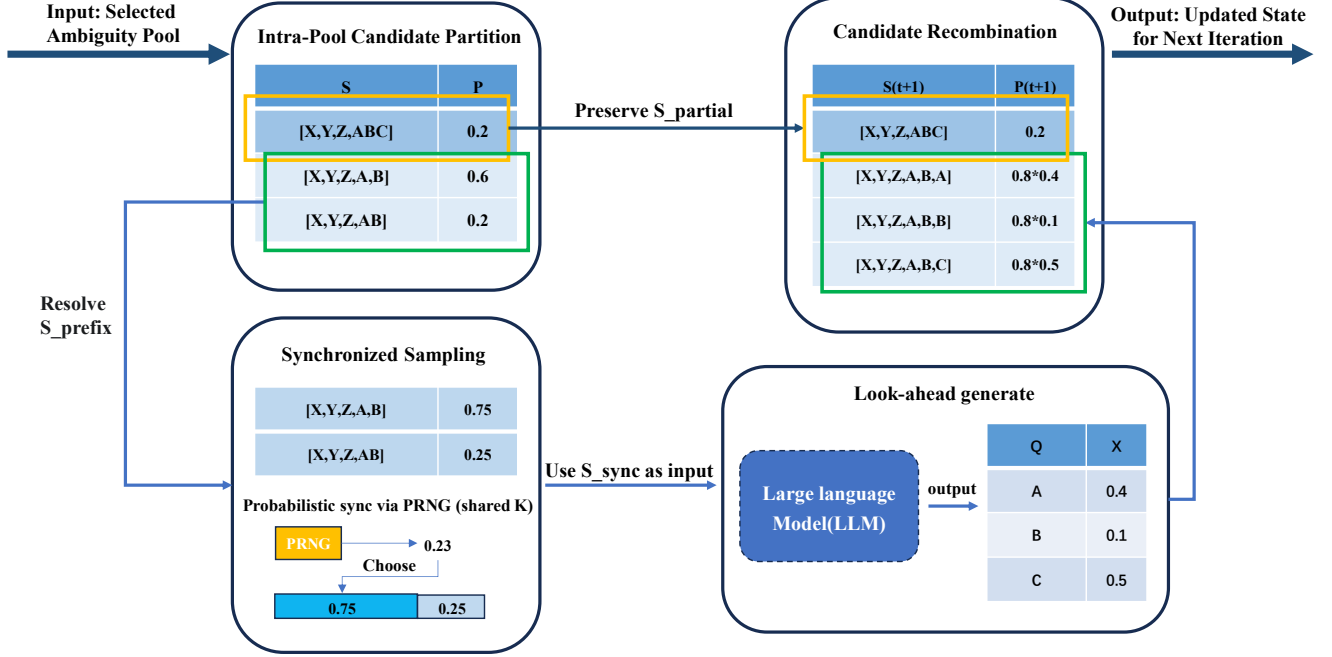


Fig. 5. Look-ahead disambiguation. The selected intra-group candidates are partitioned into a prefix set and a partial set. A synchronized sampler selects a representative s_{sync} from the prefix set, which is then expanded via an LLM call. The new candidate set for the next round is obtained by combining the preserved partial set with the new expansions, thereby reallocating probability mass to new continuations.

Since $\sum_{s \in \mathcal{S}_{\text{partial}}} P_m^{\text{intra}}(s) = 1 - p_{\text{sum}}$ and $\sum_{x \in \mathcal{A}_{\text{next}}} P_{\text{next}}(x) = 1$, it follows that $\sum_{s \in \mathcal{S}_{t+1}} P_{t+1}(s) = (1 - p_{\text{sum}}) + p_{\text{sum}} = 1$.

The resulting state (S_{t+1}, P_{t+1}) is thus a new, probabilistically sound candidate space in which the immediate ambiguity tied to v_m has been eliminated without altering the overall distribution. We treat EOS as a visible string that cannot be extended; termination occurs if and only if the synchronized choice is EOS and the partial set is empty. All forward passes are deterministic; the only randomness arises from the synchronized sampler and the entropy coder, which are shared between sender and receiver.

D. End-to-End Steganographic Pipeline

Algorithm 3 EMBEDLOOP

Require: Prompt c , secret bitstream b
Ensure: Stegotext

- 1: $(S_t, P_t) \leftarrow \text{LLM}(c)$ ▷ initialization
- 2: **while** true **do**
- 3: $(\mathcal{S}^{\text{inter}}, P^{\text{inter}}, V) \leftarrow \text{PARTITIONBYPREFIX}(S_t, P_t)$
- 4: $(m, \beta) \leftarrow \text{INTERGROUPENCODE}(P^{\text{inter}}, b)$
- 5: $b \leftarrow \text{CONSUME}(b, \beta)$
- 6: $\mathcal{S}_m^{\text{intra}} \leftarrow \mathcal{S}^{\text{inter}}[m]$
- 7: $P_m^{\text{intra}} \leftarrow \text{normalize}(\{P_t(s) \mid s \in \mathcal{S}_m^{\text{intra}}\})$
- 8: $\rho \leftarrow \text{LOOKAHEAD}(\mathcal{S}_m^{\text{intra}}, P_m^{\text{intra}}, K, \text{LLM})$
- 9: **if** $\text{ISEND}(\rho)$ **then**
- 10: **return** $\varphi(\text{ENDSEQ}(\rho))$
- 11: **else**
- 12: $(S_{t+1}, P_{t+1}) \leftarrow \rho$
- 13: $S_t \leftarrow S_{t+1}; P_t \leftarrow P_{t+1}$
- 14: **end if**
- 15: **end while**

Having detailed the constituent modules of a single generation step, we now present the complete, end-to-end steganographic pipeline. These modules are integrated into the iterative loop described above and are specified for the sender in Algorithm 3 and for the receiver in Algorithm 4. Let T denote the final stegotext observed by the decoder; write ϵ for the empty bitstring and use $\parallel$ for bit concatenation. We also use ρ to denote either a terminal marker or the next-state pair (S_{t+1}, P_{t+1}) .

Algorithm 4 DECODELOOP

Require: Prompt c , stegotext T ▷ T is immutable
Ensure: Recovered bitstream $\hat{b}$

- 1: $(S_t, P_t) \leftarrow \text{LLM}(c)$ ▷ initialization
- 2: $\hat{b} \leftarrow \epsilon$ ▷ empty bitstream
- 3: **while** true **do**
- 4: $(\mathcal{S}^{\text{inter}}, P^{\text{inter}}, V) \leftarrow \text{PARTITIONBYPREFIX}(S_t, P_t)$
- 5: $m \leftarrow \text{MATCHPREFIXINDEX}(V, T)$ ▷ find the unique $v_m \in V$ that prefixes the same T
- 6: $\beta \leftarrow \text{INTERGROUPDECODE}(P^{\text{inter}}, m)$
- 7: $\hat{b} \leftarrow \hat{b} \parallel \beta$
- 8: $\mathcal{S}_m^{\text{intra}} \leftarrow \mathcal{S}^{\text{inter}}[m]$
- 9: $P_m^{\text{intra}} \leftarrow \text{normalize}(\{P_t(s) \mid s \in \mathcal{S}_m^{\text{intra}}\})$
- 10: $\rho \leftarrow \text{LOOKAHEAD}(\mathcal{S}_m^{\text{intra}}, P_m^{\text{intra}}, K, \text{LLM})$
- 11: **if** $\text{ISEND}(\rho)$ **then**
- 12: **return** $\hat{b}$
- 13: **else**
- 14: $(S_{t+1}, P_{t+1}) \leftarrow \rho$
- 15: $S_t \leftarrow S_{t+1}; P_t \leftarrow P_{t+1}$
- 16: **end if**
- 17: **end while**

The structural identity of the embedding and decoding

loops is the foundation of the system’s reliability. The only functional difference lies in their interaction with the entropy coder. The `EMBEDLOOP` calls an encoder to consume bits, whereas the `DECODELOOP` calls a decoder to recover bits from the visible string. All other state transitions, including candidate partitioning and the synchronized sampling inside `LOOKAHEAD`, are deterministic functions of the public model state and the shared secret key K . This ensures that the receiver follows the same execution path as the sender and that the recovered bitstream $\hat{b}$ is identical to the original bitstream b .

In summary, Look-ahead Sync operates as a recursive, state-driven process that separates payload embedding from ambiguity resolution. Payload is embedded only during the unambiguous inter-group selection, whereas the subsequent intra-group resolution is handled by the look-ahead mechanism. This mechanism preserves distinguishable continuations and uses a single, non-payload-bearing synchronized sample to resolve only the core source of ambiguity. The architecture is designed to minimize the loss of Shannon entropy compared with prior work and thereby substantially improves embedding capacity.

IV. SECURITY AND CAPACITY ANALYSIS

Having detailed the mechanics of the Look-ahead Sync algorithm, we now present the theoretical analysis that supports its design and guarantees.

- **Security Analysis.** We prove that the algorithm upholds the standard of computational zero-KL security, ensuring its output is statistically indistinguishable from genuine text.
- **Capacity Analysis.** We derive the embedding-capacity upper bound for zero-KL disambiguation and quantify the gap between this bound and Look-ahead Sync.

To ensure consistency and rigor in subsequent analyses, we first formalize the notation employed in this section:

Let $\mathcal{V}$ denote the set of all terminal visible strings.

Let $\varphi(\cdot)$ denote the detokenization function that maps a token sequence to a visible string.

All text generation is conditioned on an initial prompt c ; equivalently, all probabilities (e.g., $P_\theta(\cdot \mid c)$) are taken conditional on c throughout.

A. Computational Zero-KL Security

The central security guarantee of Look-ahead Sync is its computational security, which we formally prove in this section. The proof analyzes the algorithm’s end-to-end generative process. We demonstrate that the probability of any given terminal sequence s_{end} being generated by the algorithm is computationally indistinguishable from its true probability under the base language model, $P_\theta(s_{\text{end}} \mid c)$. The validity of this proof is contingent on the standard cryptographic assumption that the output of a cryptographically secure pseudorandom number generator (CSPRNG) is computationally indistinguishable from a truly random sequence to any probabilistic polynomial-time (PPT) adversary. [26]

The proof proceeds by induction on an invariant that holds throughout the algorithm’s state-driven execution. At each step t , the algorithm maintains a state (S_t, P_t) , where S_t is the set

of candidate sequences and P_t is a probability mapping over S_t . As this state is determined by the history of probabilistic choices, both S_t and P_t are random variables. For clarity in this proof, we use the functional notation $P_t(s)$ to denote the weight corresponding to a sequence $s \in S_t$. A single instance of the state will therefore not align with the base model’s distribution. To prove the security of the overall generative process, our analysis instead focuses on its behavior in expectation.

The invariant is defined over the expected values of these weights. It asserts that for any candidate sequence $s \in S_t$, the expectation of its corresponding weight $P_t(s)$ is computationally indistinguishable from the sequence’s true probability under the base model P_θ :

$$\mathbb{E}[P_t(s)] \approx_c P_\theta(s \mid c). \quad (18)$$

Throughout this section, $\approx_c$ is interpreted in the sense of computational indistinguishability defined in Eq. (2). When applied to real-valued quantities such as expectations, it indicates that the absolute deviation is bounded by a negligible function $\text{negl}(\kappa)$. The expectation $\mathbb{E}$ is taken over the space of all possible random histories up to step t , which are determined by the probabilistic choices made by the entropy encoder and the synchronized sampler in all preceding steps.

The invariant holds for the base case ($t = 0$), as the initial state (S_0, P_0) is generated directly by the language model via $\text{LLM}(c)$. In this case, there is no random history, and for any $s \in S_0$, the equality $\mathbb{E}[P_0(s)] = P_0(s) = P_\theta(s \mid c)$ holds exactly.

For the inductive step, we assume the invariant holds for step t and demonstrate that it also holds for step $t + 1$. Any sequence $s' \in S_{t+1}$ is formed in one of two ways:

- (i) **Preservation:** If s' is preserved from a partial set, its weight $P_{t+1}(s')$ is non-zero only if its containing group m is selected. By applying the law of total expectation over all possible group selections and substituting the definitions for p_m^{inter} and $P_m^{\text{intra}}(s')$, the expression for the expected weight simplifies directly to its value from the previous step:

$$\mathbb{E}[P_{t+1}(s')] = \mathbb{E}[P_t(s')]. \quad (19)$$

By the inductive hypothesis, $\mathbb{E}[P_t(s')] \approx_c P_\theta(s' \mid c)$, thus preserving the invariant.

- (ii) **Generation:** If s' is newly generated as an expansion $s' := s_{\text{sync}} \oplus x$, its weight is defined as $P_{t+1}(s') := p_{\text{sum}} \cdot P_\theta(x \mid s_{\text{sync}})$. The expectation is taken over the choice of group m and the choice of s_{sync} . The selection of s_{sync} via `SyncSample` is driven by the CSPRNG, making its expected behavior computationally indistinguishable from a true random sample over the prefix set $\mathcal{S}_{\text{prefix}, m}$. This allows us to express the expectation as a sum over the members of that prefix set. By the linearity of expectation, we can move the summation outside:

$$\mathbb{E}[P_{t+1}(s')] \approx_c \sum_{s_p \in \mathcal{S}_{\text{prefix}, m}} \mathbb{E}[P_t(s_p) \cdot P_\theta(x \mid s_p)]. \quad (20)$$

Since $P_\theta(x \mid s_p)$ is a constant with respect to the outer expectation, it can be factored out:

$$\mathbb{E}[P_{t+1}(s')] \approx_c \sum_{s_p \in \mathcal{S}_{\text{prefix}, m}} \mathbb{E}[P_t(s_p)] \cdot P_\theta(x \mid s_p). \quad (21)$$

By the inductive hypothesis, $\mathbb{E}[P_t(s_p)] \approx_c P_\theta(s_p | \mathbf{c})$. Substituting this in and applying the chain rule of probability, we get:

$$\mathbb{E}[P_{t+1}(s')] \approx_c \sum_{s_p \in \mathcal{S}_{\text{prefix}, m}} P_\theta(s_p | \mathbf{c}) \cdot P_\theta(x | s_p) \quad (22)$$

$$= P_\theta(s' | \mathbf{c}). \quad (23)$$

The invariant is therefore maintained for newly generated sequences.

Since the invariant holds for both cases, it holds for all sequences in $\mathcal{S}_{t+1}$. By induction, the probability of generating any terminal sequence s_{end} is computationally indistinguishable from its true model probability. Therefore, the overall output distribution of Look-ahead Sync is computationally secure.

B. Theoretical Capacity Upper Bound

A rigorous analysis of the embedding capacity for provably secure linguistic steganography necessitates an understanding of its theoretical limits. This section derives the capacity upper bound for any steganographic system that resolves tokenization ambiguity while adhering to the zero-KL security standard. The bound provides a benchmark against which such methods, including our own, can be measured.

Perfect security dictates that the distribution of generated terminal sequences, P_{embed} , must be identical (or computationally indistinguishable) to the distribution induced by the original language model. A critical implication is that the distribution over the final, visible strings must also be preserved. Recall that $\mathcal{V}$ denotes the set of all possible terminal visible strings, and define the probability of generating a specific visible string $v \in \mathcal{V}$ as:

$$P_{V_{\text{end}}}(v) := \sum_{s_{\text{end}}: \varphi(s_{\text{end}})=v} P_\theta(s_{\text{end}} | \mathbf{c}). \quad (24)$$

The zero-KL security constraint requires that the distribution over visible string produced by the steganographic system, $P_{\text{embed}}(v)$, equals $P_{V_{\text{end}}}(v)$:

$$\begin{aligned} P_{\text{embed}}(v) &= \sum_{s_{\text{end}}: \varphi(s_{\text{end}})=v} P_{\text{embed}}(s_{\text{end}}) \\ &= \sum_{s_{\text{end}}: \varphi(s_{\text{end}})=v} P_\theta(s_{\text{end}} | \mathbf{c}) = P_{V_{\text{end}}}(v). \end{aligned} \quad (25)$$

Any secure system is thus a perfect sampler from the model's distribution over observable, visible string. To quantify the embedding performance of such a system, we employ two standard metrics. Let $B(s_{\text{end}})$ be the total number of bits embedded during the generation of a terminal sequence s_{end} . The expected number of bits per sequence (BPS) and bits per token (BPT) are:

$$\text{BPS} := \mathbb{E}_{s_{\text{end}} \sim P_\theta} [B(s_{\text{end}})], \quad (26)$$

$$\text{BPT} := \frac{\text{BPS}}{\mathbb{E}_{s_{\text{end}} \sim P_\theta} [|s_{\text{end}}|]}. \quad (27)$$

where $|s_{\text{end}}|$ denotes the number of tokens in the terminal sequence s_{end} .

To analyze the capacity bounds of the disambiguation strategy itself, independent of a specific coder's implementation,

we employ the theoretical construct of an ideal entropy encoder. [27] Such a coder is perfectly efficient, meaning the expected number of bits it embeds is exactly equal to the Shannon entropy of the target distribution. [28] Since a secure system must sample from the distribution of visible string $P_{V_{\text{end}}}$, Shannon's source coding theorem dictates that the maximum average number of bits that can be encoded is bounded by the entropy of this distribution,

$$H(V_{\text{end}}) = - \sum_{v \in \mathcal{V}} P_{V_{\text{end}}}(v) \log_2 P_{V_{\text{end}}}(v),$$

see, e.g., [29]. This yields the upper bounds:

$$\text{BPS} \leq H(V_{\text{end}}), \quad (28)$$

$$\text{BPT} \leq \frac{H(V_{\text{end}})}{\mathbb{E}_{s_{\text{end}} \sim P_\theta} [|s_{\text{end}}|]}. \quad (29)$$

These bounds represent the maximum embedding rates achievable by any disambiguation algorithm that satisfies the zero-KL security criterion.

C. Analysis of Look-ahead Sync's Capacity

Having established the theoretical capacity upper bound, we now analyze the performance of the Look-ahead Sync algorithm against this benchmark. The payload capacity of our method is derived exclusively from the entropy of the inter-group selection process at each step t . However, a gap exists between the achieved capacity and this theoretical optimum. This discrepancy is an inherent consequence of the non-payload-bearing synchronization mechanism required to resolve intra-group ambiguity.

This capacity gap materializes during the SyncSample step. Although the member sequences of the prefix set $\mathcal{S}_{\text{prefix}}$ are mapped to an identical visible string at the current step, they function as distinct contexts for subsequent generation. The statistical divergence in their future outcomes could, in principle, be leveraged to encode information. To exploit this, a globally optimal algorithm would need to recursively expand every sequence within $\mathcal{S}_{\text{prefix}}$ to explore all possible futures, leading to a combinatorial explosion of candidate paths. In contrast, our algorithm deliberately forgoes this exhaustive expansion, instead using SyncSample to select a single representative path for feasibility.

To analyze this forgone capacity, we define the synchronization loss as the difference between the maximum theoretical capacity available at a given step and the actual capacity achieved by our algorithm. At a step where a group with visible prefix v_m is chosen, the maximum future capacity is the entropy of the subsequent visible outcomes V_{end} conditioned only on this public information, namely $H(V_{\text{end}} | v_m)$. Our algorithm's actual capacity, however, is further conditioned on the specific choice made by SyncSample. We model this choice as a random variable S_{prefix} over the prefix set $\mathcal{S}_{\text{prefix}}$. Thus, the actual capacity is the expected conditional entropy, averaged over all possible synchronized choices:

$$H_{\text{loss}}^{\text{sync}} := H(V_{\text{end}} | v_m) - H(V_{\text{end}} | S_{\text{prefix}}, v_m). \quad (30)$$

This definition has a direct interpretation in information theory. Based on the chain rule for conditional entropy, this

difference is the conditional mutual information between the synchronized choice S_{prefix} and the future visible outcomes V_{end} , given the common prefix v_m [30]:

$$H(V_{\text{end}} | S_{\text{prefix}}, v_m) := \sum_{s_p \in \mathcal{S}_{\text{prefix}}} P(s_p | v_m) \times H(V_{\text{end}} | S_{\text{prefix}} = s_p, v_m). \quad (31)$$

The synchronization loss at this step is therefore

$$H_{\text{loss}}^{\text{sync}} \equiv I(S_{\text{prefix}}; V_{\text{end}} | v_m). \quad (32)$$

This equivalence provides a formal tool to reason about the capacity gap by revealing that the loss is the amount of information about the final outcome revealed by the non-payload-bearing synchronization process itself. The total gap between our algorithm’s performance and the theoretical bound is the sum of these expected losses over the entire generation process.

An analysis of the synchronization loss in Eq. (32) shows that its magnitude is determined by the statistical divergence among the future text distributions conditioned on different sequences within the ambiguous prefix set $\mathcal{S}_{\text{prefix}}$. A significant loss arises only when the distributions that follow semantically equivalent but tokenization distinct sequences, such as $[_{\text{mis}}, _{\text{trust}}]$ versus $[_{\text{mistrust}}]$, differ substantially. This observation motivates the Linguistic Smoothness Hypothesis, which states that a model’s semantic understanding yields statistically similar conditional distributions for such semantically equivalent sequences, so the divergence is small and the resulting conditional mutual information in Eq. (32) is negligible. We provide empirical evidence in Section V.

V. EXPERIMENTS

Following the theoretical analysis of Look-ahead Sync’s security and capacity limits in Section IV, we now turn to its empirical evaluation. This section presents a set of experiments to validate our central theoretical claims. The evaluation is designed to confirm that Look-ahead Sync maintains zero-KL security while achieving a high embedding capacity. We assess our method against key baselines on both English and Chinese benchmarks. The experiments also provide direct empirical support for the Linguistic Smoothness Hypothesis, which, as we have argued, underpins our algorithm’s high efficiency.

A. Experimental Setup

To empirically evaluate our proposed method, we conduct steganography experiments on both English and Chinese. These languages were chosen to ensure a robust assessment across different linguistic characteristics.

For English tasks, we employ the **Llama3-8b** model, and for Chinese, we utilize the **Qwen2.5-7b** model. [31], [32] Both models are large language models based on Transformer architectures [12], and their tokenizers are implemented based on subwords [15], [16], which is critical for the tokenization ambiguity problem addressed in this work. To intuitively assess how performance is affected by the number of elements in the candidate pool, we exclusively use top- k sampling [33] to constrain the size of the initial candidate pool and its

probability distribution. The temperature is consistently fixed at 1.0, as it does not directly affect the number of tokens. In our experiments, we set the truncation parameter $k \in \{16, 32, 128\}$.

For evaluation data, we sample 50 unique reviews from the IMDB movie review dataset for English tasks and 50 unique short-comments from the Douban dataset for Chinese tasks. Each sampled review or comment is truncated to serve as an initial context. This context is then provided as a prompt to the corresponding language model, which is tasked with generating a continuation. For each of these 50 unique prompts and for every top- k setting, we generate 100 steganographic messages. Each message is embedded using a new, independently generated random bitstream as the secret payload, ensuring that our results are averaged over a diverse set of contexts and messages.

We benchmark Look-ahead Sync against a carefully selected set of baselines. These include **MWIS Pruning** [19], a state-of-the-art distribution-altering method, and **SyncPool** [18], which represents the current state of the art in provably secure disambiguation. Additionally, we include results from applying the entropy encoder directly to the raw LLM output without any disambiguation, serving as a practical upper bound for capacity. To demonstrate broad compatibility, all methods are implemented on top of two distinct back-end entropy encoders: iMEC [3] and Discop [2]. All experiments are carried out on NVIDIA RTX 4090 GPUs.

In a separate experiment designed to empirically test our Linguistic Smoothness Hypothesis, we analyze the statistical divergence within real-world ambiguity scenarios. For each model, we collect 1,000 instances where an ambiguity pool, or prefix set $\mathcal{S}_{\text{prefix}}$, contains at least two distinct token sequences that decode to the same visible string. For each instance, we then compute the generalized Jensen-Shannon Divergence (JSD) [34] among the next-token probability distributions conditioned on each of these ambiguous sequences. This process yields an empirical distribution of JSD scores, which allows us to quantify the practical impact of the Synchronization Loss.

B. Evaluation Metrics

We evaluate all methods based on four key metrics, each designed to assess a critical aspect of the steganographic system’s performance.

- **Embedding Capacity (BPT $\uparrow$):** The primary metric for steganographic efficiency is bits Per Token (BPT). It is calculated by dividing the total number of embedded secret bits by the total number of tokens in the generated stegotext. A higher BPT value signifies a more efficient utilization of the language channel for data hiding.
- **Statistical Security (KL $\downarrow$):** We quantify the statistical security by measuring the Kullback-Leibler (KL) Divergence between the probability distribution used for steganographic sampling and the original distribution from the base model. A KL divergence of zero indicates that the stego and cover distributions are identical, satisfying the requirement for perfect (provably secure) steganography. Any non-zero value implies a statistical deviation that could be exploited by an adversary.

TABLE I
QUANTITATIVE COMPARISON OF LOOK-AHEAD SYNC WITH SYNCPOOL AND MWIS ON THE ENGLISH BENCHMARK
(LLAMA3-8B/IMDB)

Method	top- k	BPT ($\uparrow$)	Tok/Call ($\uparrow$)	PPL ($\downarrow$)	KL ($\downarrow$)
Baseline	16	1.51	1.00	7.13	0.00
	32	1.86	1.00	9.18	0.00
	128	2.33	1.00	12.90	0.00
MWIS [19]	16	1.59	1.00	8.80	1.80
	32	1.75	1.00	12.93	2.70
	128	2.24	1.00	27.44	5.91
SyncPool [18]	16	1.30	1.00	6.81	0.00
	32	1.39	1.00	8.11	0.00
	128	0.87	1.00	12.15	0.00
Look-ahead Sync	16	1.49	0.63	6.89	0.00
	32	1.83	0.59	9.00	0.00
	128	2.32	0.42	12.68	0.00

(a) Results with iMEC encoder.

Method	top- k	BPT ($\uparrow$)	Tok/Call ($\uparrow$)	PPL ($\downarrow$)	KL ($\downarrow$)
Baseline	16	1.72	1.00	6.84	0.00
	32	2.14	1.00	8.64	0.00
	128	2.82	1.00	12.12	0.00
MWIS [19]	16	1.75	1.00	8.77	1.79
	32	2.25	1.00	12.35	2.75
	128	2.92	1.00	28.94	6.25
SyncPool [18]	16	1.36	1.00	6.42	0.00
	32	1.63	1.00	8.90	0.00
	128	0.95	1.00	12.16	0.00
Look-ahead Sync	16	1.67	0.63	6.95	0.00
	32	2.07	0.58	8.71	0.00
	128	2.66	0.42	12.82	0.00

(b) Results with Discop encoder.

- **Text Quality (PPL $\downarrow$):** The fluency and naturalness of the generated text are assessed using perplexity (PPL). Perplexity measures how well the language model’s probability distribution predicts the generated text. A lower PPL suggests that the stegotext is closer to what the model would naturally produce, indicating less degradation in text quality.
- **Computational Efficiency (Tok/Call $\uparrow$):** To specifically measure the computational overhead introduced by our look-ahead mechanism, we define the metric Tokens per LLM Call (Tok/Call). This is the ratio of the total number of tokens in the final sequence to the total number of forward passes through the language model. A value of 1.0 corresponds to traditional single-pass generation methods, while a value less than 1.0 quantifies the additional computational cost of the recursive calls in our algorithm. We adopt this metric instead of wall-clock time to provide a more robust and hardware-independent measure of efficiency. Since factors such as the specific model, prompt, and GPU hardware can significantly affect execution time, and because all disambiguation methods share a common baseline cost (one LLM call plus ambiguity detection), Tok/Call directly isolates the core computational trade-off of our look-ahead strategy, which is directly proportional to its real-world runtime.

C. Results and Analysis

To empirically evaluate the performance of Look-ahead Sync, we benchmark it against a carefully selected set of baselines across both English and Chinese tasks. The comprehensive results are presented in Tables I and II. In these tables, **Baseline** represents the direct application of the entropy encoder to the raw model output, serving as a theoretical capacity upper bound. The **MWIS** method denotes the state-of-the-art distribution-altering approach from Yan et al. [19], while **SyncPool** is the current state-of-the-art provably secure disambiguation algorithm from Qi et al. [18]. Finally, **Look-ahead Sync** refers to our proposed algorithm. This setup enables a direct and comprehensive comparison of capacity, security, and efficiency across the spectrum of existing approaches.

1) *Security:* A steganographic method’s claim to be provably secure rests on its ability to maintain a probability distribution identical to the original cover source, which is quantified by a zero Kullback-Leibler (KL) divergence. Existing ambiguity resolution methods that rely on pruning candidates, such as MWIS, fundamentally disrupt this property. By altering the candidate pool, they inevitably introduce a non-zero KL divergence (up to 6.25 in our experiments), thereby forfeiting the guarantee of provable security. In contrast, as shown in Tables I and II, Look-ahead Sync consistently maintains a KL divergence of 0.00. This confirms that its look-ahead mechanism is a distribution-preserving transformation, making it fully compatible with the strict requirements of a provably secure framework.

In terms of text quality, perplexity (PPL) measures the fluency of the generated stegotext. However, the ideal PPL is not necessarily the lowest possible value, but rather one that closely matches the PPL of the Baseline method. The Baseline reflects the statistical properties of text generated via standard random sampling from the model’s true distribution. Any significant deviation from this PPL, whether higher or lower, suggests an unnatural generation process. The tables show that while MWIS can sometimes yield a lower PPL, it does so by altering the underlying probabilities. Look-ahead Sync, on the other hand, produces PPL values that are consistently and remarkably close to the Baseline across all settings. This indicates that the text generated by Look-ahead Sync is statistically indistinguishable in quality and naturalness from that of an unmodified language model, making it perceptually secure.

2) *Embedding Capacity:* A key contribution of this work is the high embedding capacity of Look-ahead Sync, measured in bits Per Token (BPT). The method is designed to overcome the structural limitations of prior secure disambiguation techniques. As shown in Tables I and II, the BPT of Look-ahead Sync scales effectively with the candidate pool size (k), closely following the trend of the theoretical maximum defined by the Baseline.

This scaling behavior differs from that of SyncPool. The ca-

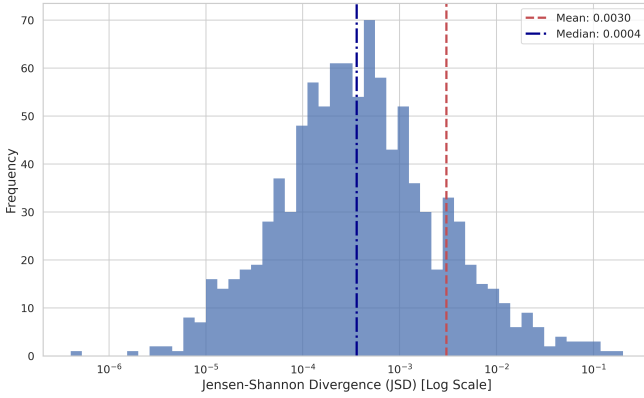
TABLE II
QUANTITATIVE COMPARISON OF LOOK-AHEAD SYNC WITH SYNCPOOL AND MWIS ON THE CHINESE BENCHMARK (QWEN2.5-7B/DOUBAN)

Method	top- k	BPT ($\uparrow$)	Tok/Call ($\uparrow$)	PPL ($\downarrow$)	KL ($\downarrow$)
Baseline	16	1.56	1.00	4.39	0.00
	32	1.74	1.00	5.44	0.00
	128	2.09	1.00	7.85	0.00
MWIS [19]	16	1.19	1.00	4.02	2.21
	32	1.47	1.00	4.90	2.75
	128	1.75	1.00	7.00	2.97
SyncPool [18]	16	1.21	1.00	4.44	0.00
	32	1.43	1.00	5.53	0.00
	128	1.61	1.00	7.42	0.00
Look-ahead Sync	16	1.53	0.75	4.47	0.00
	32	1.74	0.71	5.49	0.00
	128	2.07	0.66	7.72	0.00

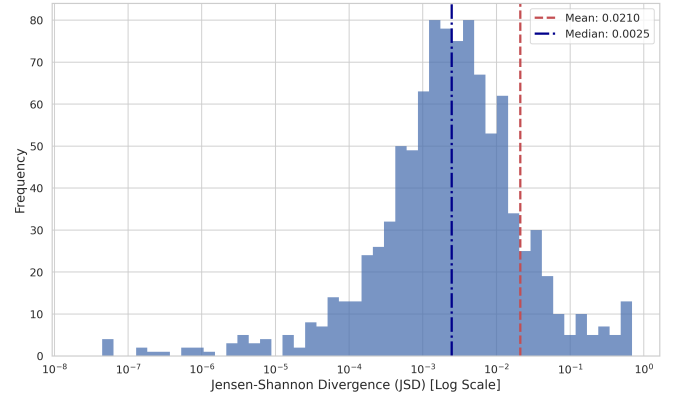
(a) Results with iMEC encoder.

Method	top- k	BPT ($\uparrow$)	Tok/Call ($\uparrow$)	PPL ($\downarrow$)	KL ($\downarrow$)
Baseline	16	1.66	1.00	4.31	0.00
	32	1.97	1.00	5.40	0.00
	128	2.53	1.00	7.67	0.00
MWIS [19]	16	1.30	1.00	3.92	2.25
	32	1.61	1.00	5.02	2.57
	128	2.17	1.00	7.02	3.03
SyncPool [18]	16	1.27	1.00	4.29	0.00
	32	1.44	1.00	5.47	0.00
	128	1.88	1.00	7.29	0.00
Look-ahead Sync	16	1.57	0.75	4.37	0.00
	32	1.85	0.70	5.24	0.00
	128	2.38	0.65	7.72	0.00

(b) Results with Discop encoder.



(a) Llama3 on IMDB (English)



(b) Qwen on Douban (Chinese)

Fig. 6. Empirical distribution of Jensen-Shannon Divergence (JSD) scores for ambiguity scenarios on (a) Llama3 and (b) Qwen.

capacity of SyncPool diminishes as k increases, a consequence of its coarse-grained synchronization mechanism which discards the entire entropy of each selected ambiguity pool. A larger candidate pool size leads to more frequent and larger ambiguity pools, thus amplifying this entropy loss. For example, in the English/Discop experiment at $k = 128$ (Table IIb), Look-ahead Sync achieves a BPT of 2.66, which is a 180% increase over SyncPool’s 0.95 BPT.

The underlying mechanism for this performance is empirically illustrated in Figure 7, which plots the ratio of Shannon entropy available for encoding. The available entropy for Look-ahead Sync remains above 97% and is stable across different values of k . In contrast, the ratio for SyncPool drops below 76% as ambiguity becomes more prevalent. This confirms that Look-ahead Sync’s look-ahead mechanism successfully preserves and utilizes the entropy of non-terminal ambiguous paths, converting what would otherwise be lost information into usable embedding capacity.

The ability of Look-ahead Sync to achieve a capacity close to the theoretical limit is further explained by the JSD distributions shown in Figure 6. The low Jensen-Shannon

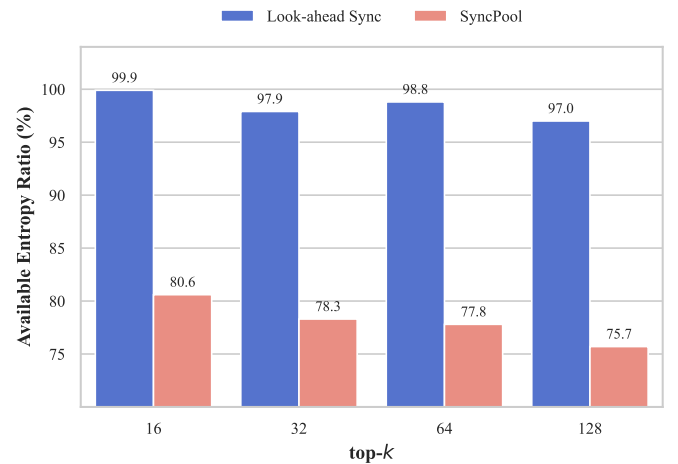


Fig. 7. Comparison of the Available Entropy Ratio for different methods on the Qwen model as top- k varies.

Divergence (JSD) scores (median $\approx$ 0.003) between the future

conditional distributions of ambiguous sequences indicate that the information loss from the synchronized sampling step is minimal in practice. This allows the computationally tractable, local resolution strategy to achieve a high capacity that approaches the theoretical capacity limit.

3) *Computational Efficiency*: The look-ahead mechanism that enables the high capacity of Look-ahead Sync introduces additional computational overhead. This is quantified by the Tokens per LLM Call (Tok/Call) metric, which measures the average number of final tokens generated per forward pass of the language model. As shown in Tables I and II, the Tok/Call for Look-ahead Sync is less than 1.0. This is a direct result of the additional LLM calls that are triggered when ambiguity is encountered in the candidate pool.

This computational overhead is a key characteristic of the method. In steganographic applications where maximizing embedding capacity is the primary objective, Look-ahead Sync makes it possible to achieve substantial BPT gains, and this is associated with an increase in the number of LLM calls. The extent of this overhead is influenced by the sampling parameter k ; larger values tend to increase the frequency of ambiguity, leading to a lower Tok/Call.

VI. DISCUSSION

While the Look-ahead Sync algorithm successfully enhances embedding capacity under a provably secure framework, two primary limitations warrant discussion. The first is the computational overhead introduced by the look-ahead mechanism, and the second is that the embedding capacity, while significantly improved, does not yet reach the theoretical upper bound established in our analysis.

The computational overhead stems from the additional language model forward passes required to execute the look-ahead strategy. This increase in computation is a direct trade-off for the algorithm's capacity gains, as each additional call serves to preserve and transform the Shannon entropy that is otherwise discarded by prior secure methods. For steganographic applications where achieving high embedding capacity is the principal objective, such as embedding substantial payloads within concise coverttexts, this increased demand on computational resources may be an acceptable compromise.

The second limitation, the residual gap to the theoretical capacity limit, originates from the information loss inherent in the non-payload-bearing SyncSample step. A clear avenue for future research to mitigate this loss lies in the development of adaptive look-ahead strategies. A more sophisticated implementation could employ a heuristic, such as the entropy of the prefix set, $H(\mathcal{S}_{\text{prefix}})$, to dynamically guide its operations. For instance, a system could trigger a look-ahead call only when the potential capacity gain is significant, or even perform a more resource-intensive multi-path expansion for particularly high-entropy prefix sets. Such an approach could lead to a more optimal allocation of computational resources, further closing the gap to the theoretical capacity limit and enhancing the practical utility of the look-ahead paradigm.

VII. CONCLUSION

A critical barrier to the practical use of provably secure linguistic steganography has been the substantial embedding capacity sacrificed by existing secure algorithms. In this paper, we identify the root of this limitation in the state-of-the-art method, SyncPool, whose coarse-grained synchronization mechanism systematically discards valuable Shannon entropy. To overcome this critical bottleneck, we introduce Look-ahead Sync, a novel, recursive disambiguation algorithm. The core of our method is a look-ahead resolution strategy that performs minimal synchronized sampling only on truly indistinguishable token sequences.

By strategically preserving all other discernible candidate paths and reallocating their probability mass, Look-ahead Sync maintains the rigorous zero-KL divergence guarantee of the synchronization paradigm while systematically eliminating the capacity constraints of prior work. Our approach ensures a uniquely determined decoding result for the receiver without forfeiting the informational content of distinguishable future paths.

Extensive experiments conducted on both English and Chinese language models validate our approach. The results demonstrate that Look-ahead Sync's embedding rate consistently approaches the theoretical capacity upper bound and substantially outperforms existing secure methods, with capacity gains exceeding 160% in some settings. These gains are achieved without compromising the strict zero-KL security standard or the perceptual quality of the generated text. This work represents a significant step toward making high-capacity, provably secure linguistic steganography a practical and viable technology for covert communication.

REFERENCES

- [1] S. Zhang, Z. Yang, J. Yang, and Y. Huang, "Provably secure generative linguistic steganography," *arXiv preprint arXiv:2106.02011*, 2021.
- [2] J. Ding, K. Chen, Y. Wang, N. Zhao, W. Zhang, and N. Yu, "Discop: Provably secure steganography in practice based on "distribution copies"," in *2023 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2023, pp. 2238–2255.
- [3] C. S. de Witt, S. Sokota, J. Z. Kolter, J. Foerster, and M. Strohmeier, "Perfectly secure steganography using minimum entropy coupling," *arXiv preprint arXiv:2210.14889*, 2022.
- [4] Z. M. Ziegler, Y. Deng, and A. M. Rush, "Neural linguistic steganography," *arXiv preprint arXiv:1909.01496*, 2019.
- [5] Z.-L. Yang, X.-Q. Guo, Z.-M. Chen, Y.-F. Huang, and Y.-J. Zhang, "Rnn-stega: Linguistic steganography based on recurrent neural networks," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 5, pp. 1280–1295, 2018.
- [6] H. Yang, Y. Bao, Z. Yang, S. Liu, Y. Huang, and S. Jiao, "Linguistic steganalysis via densely connected lstm with feature pyramid," in *Proceedings of the 2020 ACM Workshop on Information Hiding and Multimedia Security*, 2020, pp. 5–10.
- [7] Z.-L. Yang, S.-Y. Zhang, Y.-T. Hu, Z.-W. Hu, and Y.-F. Huang, "Vae-stega: linguistic steganography based on variational auto-encoder," *IEEE Transactions on Information Forensics and Security*, vol. 16, pp. 880–895, 2020.
- [8] H. Wang, Z. Yang, J. Yang, C. Chen, and Y. Huang, "Linguistic steganalysis in few-shot scenario," *IEEE Transactions on Information Forensics and Security*, vol. 18, pp. 4870–4882, 2023.
- [9] J. Yang, Z. Yang, J. Zou, H. Tu, and Y. Huang, "Linguistic steganalysis toward social network," *IEEE Transactions on Information Forensics and Security*, vol. 18, pp. 859–871, 2022.
- [10] C. Cachin, "An information-theoretic model for steganography," in *International Workshop on Information Hiding*. Springer, 1998, pp. 306–318.

- [11] N. J. Hopper, J. Langford, and L. Von Ahn, "Provably secure steganography," in *Annual International Cryptology Conference*. Springer, 2002, pp. 77–92.
- [12] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.
- [13] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.
- [14] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, and G. Lample, "LLaMA: Open and efficient foundation language models," *arXiv preprint arXiv:2302.13971*, Feb. 2023.
- [15] R. Sennrich, B. Haddow, and A. Birch, "Neural machine translation of rare words with subword units," *arXiv preprint arXiv:1508.07909*, 2015.
- [16] T. Kudo and J. Richardson, "Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing," *arXiv preprint arXiv:1808.06226*, 2018.
- [17] J. Nozaki and Y. Murawaki, "Addressing segmentation ambiguity in neural linguistic steganography," *arXiv preprint arXiv:2211.06662*, 2022.
- [18] Y. Qi, K. Chen, K. Zeng, W. Zhang, and N. Yu, "Provably secure disambiguating neural linguistic steganography," *IEEE Transactions on Dependable and Secure Computing*, 2024.
- [19] R. Yan, Y. Yang, and T. Song, "A secure and disambiguating approach for generative linguistic steganography," *IEEE Signal Processing Letters*, vol. 30, pp. 1047–1051, 2023.
- [20] F. Li, P. Wei, T. Fu, Y. Lin, and W. Zhou, "Imperceptible text steganography based on group chat," in *2024 IEEE International Conference on Multimedia and Expo (ICME)*. IEEE, 2024, pp. 1–6.
- [21] G. J. Simmons, "The prisoners' problem and the subliminal channel," in *Advances in Cryptology: Proceedings of Crypto 83*. Springer, 1984, pp. 51–67.
- [22] F. A. Petitcolas and S. Katzenbeisser, *Information hiding techniques for steganography and digital watermarking (Artech House Computer Security Series)*. Artech House, 2000.
- [23] Y. Li, K. Chen, Y. Wang, X. Zhang, G. Wang, W. Zhang, and N. Yu, "Coas: Composite audio steganography based on text and speech synthesis," *IEEE Transactions on Information Forensics and Security*, 2025.
- [24] K. Lin, Y. Luo, Z. Zhang, and P. Luo, "Zero-shot generative linguistic steganography," *arXiv preprint arXiv:2403.10856*, 2024.
- [25] Y.-S. Huang, P. Just, K. Narayanan, and C. Tian, "Od-stega: Llm-based near-imperceptible steganography via optimized distributions," *arXiv preprint arXiv:2410.04328*, 2024.
- [26] J. Håstad, R. Impagliazzo, L. A. Levin, and M. Luby, "A pseudorandom generator from any one-way function," *SIAM Journal on Computing*, vol. 28, no. 4, pp. 1364–1396, 1999.
- [27] J. J. Rissanen, "Generalized kraft inequality and arithmetic coding," *IBM Journal of research and development*, vol. 20, no. 3, pp. 198–203, 1976.
- [28] C. E. Shannon, "A mathematical theory of communication," *The Bell system technical journal*, vol. 27, no. 3, pp. 379–423, 1948.
- [29] T. M. Cover, *Elements of information theory*. John Wiley & Sons, 1999.
- [30] W. McGill, "Multivariate information transmission," *Transactions of the IRE Professional Group on Information Theory*, vol. 4, no. 4, pp. 93–111, 1954.
- [31] A. Grattafiori, A. Dubey, A. Jauhri, A. Pandey, A. Kadian, A. Al-Dahle, A. Letman, A. Mathur, A. Schelten, A. Vaughan *et al.*, "The llama 3 herd of models," *arXiv preprint arXiv:2407.21783*, 2024.
- [32] B. Hui, J. Yang, Z. Cui, J. Yang, D. Liu, L. Zhang, T. Liu, J. Zhang, B. Yu, K. Lu *et al.*, "Qwen2. 5-coder technical report," *arXiv preprint arXiv:2409.12186*, 2024.
- [33] A. Holtzman, J. Buys, L. Du, M. Forbes, and Y. Choi, "The curious case of neural text degeneration," *arXiv preprint arXiv:1904.09751*, 2019.
- [34] J. Lin, "Divergence measures based on the shannon entropy," *IEEE Transactions on Information theory*, vol. 37, no. 1, pp. 145–151, 2002.