Breaking the Generative Steganography Trilemma: ANStega for Optimal Capacity, Efficiency, and Security

Yaofei Wang* Weilong Pang* Kejiang Chen† Jinyang Ding†
Donghui Hu* Weiming Zhang† Nenghai Yu†
*Hefei University of Technology †University of Science and Technology of China
Email: *{wyf@,pangwl@mail.,hudh@}hfut.edu.cn †{chenkj@,source@mail.,zhangwm@,ynh@}ustc.edu.cn

Abstract*—Generative steganography shows immense promise for covert communication, yet existing methods are often constrained by a trilemma of capacity, efficiency, and security. Methods based on Huffman Coding (HC) suffer from poor efficiency and security, while those based on Arithmetic Coding (AC), despite achieving optimal capacity, also pose security risks. Although recent provably secure methods have addressed the security issue, they often do so at the cost of elevated embedding complexity or diminished capacity—failing to match the high capacity exhibited by AC-based methods. To address this trilemma, we adapt Asymmetric Numeral Systems (ANS) for steganography. Our core insight is to repurpose the ANS state machine, using its decoding function for embedding and its encoding function for extraction. To translate this concept into a practical system, we introduce several key innovations. First, we incorporate a streaming architecture with state renormalization to enable the stable embedding of arbitrarily long messages. Second, we employ direct floating-point arithmetic, avoiding costly probability-to-frequency conversions to reduce complexity and precision loss. More critically, we introduce an innovative cryptographic mask mechanism that ensures the sampling process is driven by a cryptographically secure pseudo-random number generator, thereby achieving provable security. Finally, by optimizing core computations into highly efficient bitwise shift operations, ANStega achieves exceptional embedding and extraction speeds. Experimental results validate that ANStega simultaneously achieves optimal embedding capacity, optimal efficiency (embedding complexity with $O(1)$) and optimal security, successfully resolving the long-standing trilemma in generative steganography.

I. Introduction

In today's environment of increasing online surveillance and widespread traffic analysis [1], [2], the ability to share information without revealing that a conversation is taking place has become as important as protecting the content of the messages themselves. While encryption protects content, it exposes metadata [3]. This vulnerability can be exploited by adversaries to infer relationships between individuals or to enforce censorship. Steganography [4] addresses this by embedding secret data within innocuous media. This method ensures that the act of sending messages remains undetectable and appears indistinguishable from normal data exchange. Furthermore, these covert channels facilitate digital "dead drop" workflows [5], allowing a hidden message to be posted in a public forum and discreetly retrieved by the intended recipient without any traceable interaction. This approach extends privacy protections beyond what encryption alone can offer.

Recently, the emergence of powerful generative models,

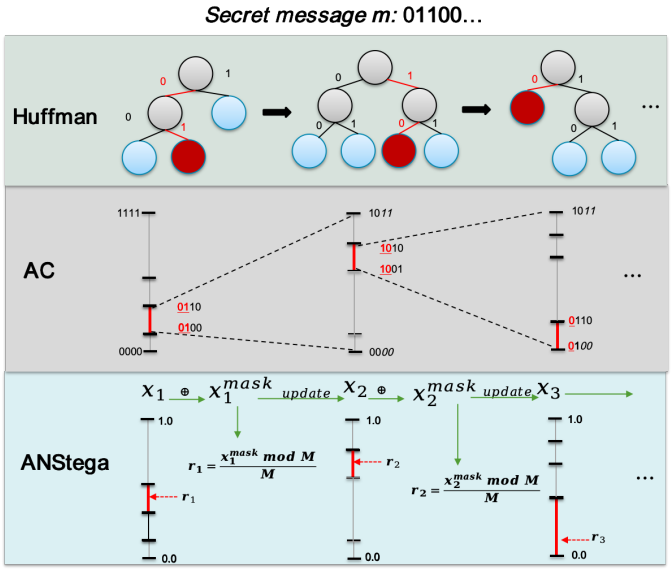

Fig. 1: **Comparison of entropy-coding schemes for generative steganography.** Top: Huffman coding maps bits to tokens via dynamic trees, causing distributional deviation. Middle: Arithmetic Coding (AC) operates on cumulative intervals but suffers from precision limitations and rescaling overhead. Bottom: **ANStega (ours)** utilizes constant-time state updates based on Asymmetric Numeral Systems, achieving near-entropy capacity with distributional indistinguishability.

Corresponding authors: Donghui Hu and Weiming Zhang

Network and Distributed System Security (NDSS) Symposium 2026
23-27 February 2026, San Diego, CA, USA
ISBN 979-8-9919276-8-0
https://dx.doi.org/10.14722/ndss.2026.240605
www.ndss-symposium.org

TABLE I: The Capacity-Efficiency-Security trilemma in generative steganography. ● denotes that the method achieves the optimal goal for the corresponding property, whereas ○ indicates a sub-optimal solution. Our work, ANStega, is the first to resolve this trilemma.

Method	**Optimal Capacity** (Near Entropy Limit)	**Optimal Efficiency** ($O(1)$ Complexity)	**Optimal Security** ($D_{\mathrm{KL}} \leq \epsilon$)	**Modality**
Huffman-based [6], [7]	●*	○	○	Text †
AC-based [8]–[11]	●*	○	○	Text / Audio †
ADG [12]	○	○	○	Text †
Meteor [13]	○	○	●	Text †
Discop [14]	○	○	●	Text / Image / Audio †
iMEC [15]	○	○	●	Text / Image / Audio †
SA-ANS [16]	○	○	○	Text †
FDPSS [17]	○	○	●	Text †
Shimmer [18]	○	●	●	Text †
SparSamp [19]	○	●	●	Text / Image / Audio †
Image-based Methods:				
Pulsar [20]	○	○	●	Image
StegaDDPM [21]	○	●	●	Image
ANStega (Ours)	●	●	●	Text / Image †

*These methods achieve near-optimal capacity by altering the model's original probability distribution.
†Note that the coding algorithms are theoretically modality-agnostic; the listed modalities reflect the specific implementations in the cited works.

particularly Large Language Models (LLMs), has catalyzed a significant shift in steganographic methodologies. Unlike traditional approaches that modify existing covers and risk detectable statistical anomalies [22], [23], generative steganography leverages powerful generative models to synthesize high-entropy, natural-looking media directly. Consequently, generative steganography has flourished across various modalities, including text [11], [12], images [9], and audio [10]. Researchers leverage these models' rich, high-dimensional probability distributions to construct covert channels with high capacity and minimal statistical detectability

Historically, generative steganography is encoding-driven, adapting long-standing entropy coding techniques to steganographic use. Anderson and Petitcolas [24] first highlighted a critical insight—the duality between steganography and inverse compression. Early generative approaches adapted classical entropy coders, predominantly Huffman Coding (HC) and Arithmetic Coding (AC), to navigate generative model distributions. HC-based methods [6], [7], despite their conceptual simplicity, suffer from frequent tree reconstruction overhead and unavoidable distribution modifications, resulting in suboptimal efficiency and compromised security. Conversely, AC-based methods [9]–[11] achieve near-optimal capacity but incur significant computational overhead due to probability scaling, and also introduce precision-induced distortions, weakening security. As illustrated in Fig. 1, entropy-coding-based steganography navigates the model distribution to "decompress" a secret bitstream into tokens, thereby aligning embedding with generative sampling.

Recognizing these limitations, subsequent research emphasized provable security, as seen in schemes like Meteor [13], Discop [14], and iMEC [15]. Although these schemes guarantee robust security, they often impose substantial penalties on embedding capacity and computational performance, limiting their practicality. Even recent methods such as SparSamp [19], which achieve optimal security and efficiency, exhibit only a slight shortfall in embedding capacity—despite effectively reaching the entropy limit—due to constraints inherent to sparse sampling. Beyond the modality-agnostic coding algorithms, generative steganography has also been explored in image generation using diffusion models, yet similar trade-offs persist. For example, StegaDDPM [21] achieves high operational efficiency by integrating embedding into the diffusion sampling steps, but its capacity is low. On the other hand, Pulsar [20] prioritizes cryptographic security and extraction, however, this comes at the cost of computational overhead and reduced effective embedding rates.

Despite this promising direction, generative steganography remains fundamentally constrained by the Capacity-Efficiency-Security Trilemma, illustrated in Table I. An optimal steganographic method should simultaneously achieve three goals: (1) **Optimal Capacity**, approaching the Shannon entropy limit with a vanishing gap (near-entropy embedding); (2) **Optimal Efficiency**, characterized by constant-time complexity $O(1)$; and (3) **Optimal Security**, ensuring statistical indistinguishability between the stego-content and the cover-content. However, existing methods have invariably compromised at least one of these crucial attributes.

This fragmented landscape highlights the need for a fundamentally novel coding paradigm to comprehensively resolve the trilemma. In this context, we identify Asymmetric Numeral Systems (ANS), introduced by Duda [25], as an overlooked yet promising entropy coder. ANS uniquely combines the computational simplicity reminiscent of Huffman Coding with the high compression efficiency characteristic of Arithmetic Coding. Its robust industrial adoption by major technology

firms, including Meta [26], Apple [27], Google [28], and Microsoft [29], underscores its practical strengths. While recent studies have begun to explore ANS for steganography [16], fully harnessing its potential to simultaneously achieve optimal capacity, constant-time efficiency, and provable security remains an open challenge.

We present ANStega, a strictly provably secure steganographic encoder built on the ANS framework. Unlike concurrent approaches that focus primarily on sampling strategies [16], we do not merely transplant ANS: a naïve port fails to provide long-message support, constant-time embedding, or rigorous distributional security. We therefore introduce a set of steganography-specific modifications: streaming rANS with bounded state to embed arbitrarily long messages without numerical blow-up; direct probability updates without global rescaling (no convert/renormalize over the full vocabulary), preserving the model distribution while maintaining $O(1)$ per-step work; a per-step MASK (CSPRNG-based) that enforces sampling randomness and independence, yielding distributional indistinguishability; and a reimplementation of core arithmetic in bitwise primitives, further improving embedding throughput. Together, these changes make ANS fit for steganography: ANStega attains near-entropy capacity with constant-time embedding and provable security, while remaining practical for high-throughput LLM generation.

ANStega uniquely solves the generative steganography trilemma, achieving simultaneous optimal performance across embedding capacity, security, and efficiency. We experimentally validate that ANStega approaches the theoretical entropy limit, matches AC-based methods in capacity, and significantly exceeds them in computational efficiency and statistical security. Our theoretical proofs and extensive empirical evaluations confirm that ANStega preserves model distributions and achieves $O(1)$ complexity by eliminating the scaling overhead typical in AC-based schemes.

Our main contributions are as follows:

- We propose ANStega, a generative steganographic scheme based on Asymmetric Numeral Systems. Rather than directly adopting ANS, we systematically redesign it for steganographic encoding by reversing its encoding-decoding roles, aligning seamlessly with LLM-based token generation.
- We introduce a tailored streaming rANS approach, allowing the stable embedding of arbitrarily long messages by maintaining bounded state intervals. This strategy effectively eliminates numerical overflow issues inherent in traditional ANS implementations.
- To enhance efficiency, we devise a direct floating-point update strategy that avoids global probability rescaling, and further accelerate embedding through bitwise arithmetic transformations, resulting in a constant-time embedding complexity ($O(1)$).
- We ensure provable steganographic security by integrating a cryptographic masking mechanism (MASK), which guarantees uniform randomness during token sampling. We prove that ANStega preserves the original model distribution, achieving optimal security.
- Extensive empirical evaluations validate that ANStega achieves optimal embedding performance across capacity, efficiency, and security metrics, successfully resolving the long-standing generative steganography trilemma.

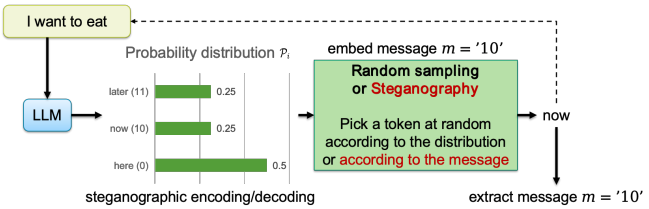

Fig. 2: A concrete example of LLM-based Steganography.

II. Background and Related Work

A. Steganographic Model

Steganography enables two communicating parties to exchange secret messages over a public channel without an adversary detecting the existence of communication. A classic formalization of this scenario is Simmons' "Prisoners' Problem" [30], in which two prisoners, Alice and Bob, wish to plan an escape by exchanging seemingly innocuous messages. A prison warden inspects all messages carefully, rejecting any communication that appears suspicious. This scenario captures two fundamental requirements of steganography:

1) Correctness: Bob must reliably recover the hidden message M from the received stego medium S without errors.
2) Undetectability: The warden's ability to distinguish the stego medium S from a legitimate cover medium C should be no better than random guessing.

Steganographic models generally fall into two principal paradigms: *modification-based* and *generation-based*.

In *modification-based steganography*, given a cover medium $C \in \mathcal{C}$, a secret message $M \in \mathcal{M}$, and a secret key K, the embedding process produces a stego medium $S \in \mathcal{S}$ by minimally altering the cover medium:

$$S = \mathcal{E}(C, M, K), \tag{1}$$

where K is shared secretly between Alice and Bob. The embedding process ensures S is visually or statistically indistinguishable from C to unauthorized observers.

In contrast, *generation-based steganography* directly synthesizes a stego medium from the secret message without relying on a pre-existing cover. Formally, this generation process is represented as:

$$S = \mathcal{G}(M, K), \tag{2}$$

where $\mathcal{G}$ denotes a generative function parameterized by the secret key K. The principal advantage of generative steganography lies in eliminating explicit modification artifacts, potentially complicating detection.

To facilitate a better understanding of the generation-based steganographic model, we introduce a concrete running example based on LLM steganography, as illustrated in Figure 2.

Consider a scenario where a sender wishes to conceal a secret message, m = '10', within a generated text stream. The process operates as follows:

1) **Context and Distribution:** The LLM receives a prompt (e.g., "I want to eat") and computes a probability distribution $\mathcal{P}_i$ for the next token. In our example, the tokens "later" (11), "now" (10), and "here" (0) are assigned probabilities of 0.25, 0.25, and 0.5, respectively.
2) **Embedding (Encoding):** Rather than sampling the next token purely at random—which is standard in normal generation—the sender utilizes a steganographic algorithm to map the message bits m to a specific token. Here, the message '10' maps directly to the token "now".
3) **Output:** The token "now" is selected and appended to the text, resulting in "I want to eat now". To an outside observer, this selection appears to be a natural continuation of the sentence, statistically consistent with the distribution $\mathcal{P}_i$.

B. Generative Steganography

Early theoretical work on steganography established provably secure techniques for "cover-free" channels, yet these methods remained largely impractical due to the inability to sample realistic media distributions [31]. The advent of high-quality generative models—LLMs for text, diffusion or GAN/flow models for images and audio—has fundamentally changed the landscape by enabling realistic, high-entropy content synthesis that supports covert message embedding with minimal detectability.

In text, early LSTM-based schemes such as Fang et al. [32] encoded bits by selecting tokens from fixed vocabulary bins, but suffered from detectability issues. RNN-Stega [33] improved capacity by using Huffman coding over conditional token probabilities, though the required tree reconstruction and distribution distortion limited security and efficiency. The introduction of Arithmetic Coding (AC) with LLMs (e.g. Ziegler et al. [11]) allowed near-entropy embedding, but AC's rescaling operations introduce precision loss and runtime overhead.

Subsequent work aimed to establish provably secure schemes. Meteor [34] mitigated randomness-reuse risks in AC-based embedding via recoverable-range sampling, improving cryptographic security at the cost of added complexity. Adaptive Dynamic Grouping (ADG) [35] dynamically partitions the token space to align with the LLM distribution, enhancing imperceptibility. Distribution-copy methods such as Discop [14] and minimum-entropy coupling like iMEC [15] further preserve statistical indistinguishability by constructing exact distributional copies or couplings—but both incur significant runtime cost. More recently, SparSamp [19] achieved $O(1)$ embedding and perfect security in practice, but still exhibits a minor capacity gap due to its sparse-sampling design. Very recently, concurrent work SA-ANS [16] has begun to explore ANS for linguistic steganography. However, SA-ANS relies on direct state-dependent sampling without cryptographic masking, leaving statistical artifacts detectable by analysis. Furthermore, it lacks optimal complexity and employs fixed-rate consumption that limits capacity on low-entropy distributions.

Generative steganography has also developed beyond text: PixelCNN-based approaches for images [9], audio steganography via TTS models [10], and diffusion-based frameworks [20], [21] have all explored embedding messages during generation. Collectively, these advances demonstrate modality-wide applicability of generative steganography in AIGC, covering encoding-based designs and alternative mechanisms alike.

Despite this rich body of work, no existing approach simultaneously achieves (i) near-entropy embedding rate, (ii) constant-time efficiency ($O(1)$), and (iii) provable distributional indistinguishability. In particular, while ANS—a coding paradigm with Huffman-like efficiency and AC-level compression effectiveness—has attracted recent attention [16], its potential to fundamentally resolve this trilemma remains unrealized. Our work fills that gap by systematically adapting ANS to the steganographic setting, enabling efficient, provably secure, and high-capacity generative steganography.

C. Asymmetric Numeral Systems

Asymmetric Numeral Systems (ANS) is a family of entropy coders introduced by Duda that combines AC–like compression effectiveness with Huffman-like efficiency [36]. Two widely used ANS variants are table-based ANS (tANS) and range-based ANS (rANS). tANS precomputes state transitions in lookup tables so that runtime encoding/decoding primarily performs bit-shifts, additions, and table reads; rANS maintains a single integer state and updates it with integer multiply–add and div/mod operations analogous to range (arithmetic) coding. ANS has seen broad industrial adoption: Zstandard [26], [37] and Apple's LZFSE [27] employ Finite State Entropy, and the JPEG XL standard [38] supports ANS as one of its entropy coders. For steganographic use with generative models, we favor rANS because the target token distribution changes at every step, making tANS lookup tables impractical to precompute or maintain.

The core idea of ANS is to represent a sequence of symbols as a single natural number, effectively capturing the entire message's information content. This is achieved by maintaining a state variable x, which is updated as each symbol is processed. Let f_s denote the frequency of symbol s, $C_s = \sum_{a<s} f_a$ its cumulative frequency, and $M = \sum_{a\in\Sigma} f_a$ the total frequency. Each symbol s is encoded via:

$$x \leftarrow \left\lfloor \frac{x}{f_s} \right\rfloor \cdot M + C_s + (x \bmod f_s), \tag{3}$$

after all symbols have been processed, the final state x and any residual bits are written out as the compressed output. During decoding the process is reversed, successively recovering

symbols while restoring earlier states—thereby reproducing the original message exactly.

The decoding in rANS proceeds by inverting the encoding transformation to recover the original message from the final state x. At each step, the current symbol s is identified by the residue:

$$s \leftarrow \mathsf{symbol}\left(x \bmod M\right), \tag{4}$$

the state is then updated as

$$x \leftarrow f_s \cdot \left\lfloor \frac{x}{M} \right\rfloor + (x \bmod M) - C_s, \tag{5}$$

this process iterates until all symbols are decoded in order.

III. Motivation

The relationship between entropy coding and steganographic encoding provides a useful duality for modern generative steganography. In classical information theory, entropy coders such as Huffman Coding and Arithmetic Coding compress data by removing statistical redundancy, transforming a symbol sequence into a bitstream that is close to uniform [39]. Anderson and Petitcolas [24] noted a consequence for steganography: if one has a (near-)perfect generative model of the cover distribution and an invertible coder, then the *inverse* mapping can "decompress" a random bitstream (e.g., encrypted data) into plausible samples from that distribution. Informally, steganographic embedding can be viewed as the inverse of entropy coding: instead of compressing symbols into bits, we map a secret bitstream into symbols that follow the model's target distribution.

This perspective naturally motivated adapting classical coders for steganography. However, direct use of HC/AC exposes a persistent capacity–efficiency–security tension:

- **AC-based** methods can approach the entropy limit and thus offer high embedding capacity [8]–[11]. In practice, each token step requires range updates and renormalization under finite precision, plus converting model probabilities to scaled integers. Although amortized time per step is constant, these operations introduce nontrivial overhead and sensitivity to precision/rounding, which complicates accurate sampling of low-probability tokens and can affect security if randomness management is not careful.
- **HC-based** methods, while fast in static compression, lose that advantage here: the model distribution changes at every step, implying frequent tree rebuilds or reweighting. Moreover, HC requires dyadic probabilities; quantization to dyadic weights induces a systematic mismatch to the target distribution, yielding suboptimal capacity and measurable KL divergence [6], [7].

In this paper, we hypothesize that the unique properties of ANS provide the ideal foundation for resolving the steganographic trilemma. Its near-optimal compression ratio suggests the potential to achieve maximum capacity; its low-complexity points toward $O(1)$ efficiency; and its precise, scale-free arithmetic offers a way to avoid the distributional distortions that plague other methods. Therefore, this paper is dedicated to investigating this hypothesis. We present the adaptation of ANS for generative steganography, describing the necessary innovations to harness its theoretical strengths.

IV. Threat Model

We analyze ANStega against a *passive* adversary (the warden, Eve) whose goal is *steganographic detection*. Eve observes the public channel between the sender (Alice) and the receiver (Bob) and seeks to distinguish stego text S (carries a hidden message) from cover text C (does not). This is a binary hypothesis test; Eve's success is measured by any distinguishing advantage over random guessing.

a) Kerckhoffs-aligned capabilities: We adopt a strong, standard model in which security relies only on secret keys, not on algorithm secrecy.

- **System knowledge.** Eve knows the generative model $\mathcal{G}$ (architecture, parameters, and sampling hyperparameters such as temperature) and the complete ANStega algorithm and its implementation details.
- **Channel access.** Eve intercepts the full token sequence and the prompt/context Φ for every message sent over the public channel. She may perform arbitrary statistical tests or train detectors using in-distribution corpora, and she may query $\mathcal{G}$ (black-box sampling) to draw matched reference samples.
- **No access to secrets.** Eve does *not* know the shared key k_{seed} used to derive the per-step MASK via a CSPRNG. Internal ANS states and MASK values are never revealed on the public channel.

b) Passive Adversary: We focus on a passive adversary model, which is the standard setting for linguistic steganography aiming for high capacity [6]–[19]. In this model, Eve monitors the public channel to detect anomalies but does not modify traffic. We defer the discussion of active adversaries—who may inject or alter tokens to disrupt synchronization—to Section VIII, noting that resistance to such attacks typically requires sacrificing the near-entropy capacity that ANStega aims to achieve.

V. Proposed Method

A. Overview of Steganography System Based on ANS

The foundational principle of our system lies in mapping the ANS framework to the generative steganography task. Instead of merely using ANS as a black-box compressor, we repurpose its core state machine for message embedding and extraction. The key correspondences are established as follows (and illustrated in Figure 3):

- **ANS Symbol Sequence** $\leftrightarrow$ **StegoText** S: The sequence of symbols produced by the state machine constitutes the final steganographic text.
- **ANS Alphabet** $\leftrightarrow$ **Model Vocabulary** $\mathcal{V}$: The set of all possible symbols corresponds to the vocabulary of the generative model $\mathcal{G}$.
- **ANS Frequency Table** $\leftrightarrow$ **Model Probability Distribution** $P(s \mid \cdot)$: The symbol frequencies are determined

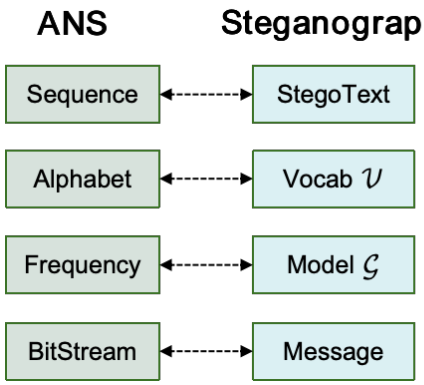

Fig. 3: Mapping the core components of ANS to the generative steganography task.

at each step by the next-token probability distribution predicted by the generative model.

- **ANS Bitstream $\leftrightarrow$ Secret Message** m: The secret message provides the bits that drive the state machine's evolution.

The central innovation of our approach lies in **inverting the conventional roles** of the ANS state transition functions, $\mathcal{C}(s, x)$ (encode) and $\mathcal{D}(x)$ (decode). This inverse design is intuitive for generative tasks:

- Standard ANS **decoding** takes a state x_{t+1} and produces a symbol s_t. This is precisely the functionality required for generative steganography, where we need to generate a token (symbol) based on the current state, which is derived from the secret message.
- Conversely, standard ANS **encoding** consumes a sequence of symbols to produce a final state. This mirrors our message extraction process, which consumes the sequence of received tokens to reconstruct the state and thereby reveal the original secret message.

Therefore, we formally define our embedding function $\mathsf{Encode}_{\mathcal{G}}$ as the ANS decoding function $\mathcal{D}$, and our extraction function $\mathsf{Decode}_{\mathcal{G}}$ as the ANS encoding function $\mathcal{C}$:

$$\begin{aligned} \text{Embedding}: &\quad (x_t, s_t) = \mathcal{D}(x_{t+1}) \\ \text{Extraction}: &\quad x_{t+1} = \mathcal{C}(s_t, x_t) \end{aligned} \tag{6}$$

A critical consequence of this "decode-first, encode-later" architecture is the need to synchronize the extractor's initial state with the embedder's final state. To correctly reconstruct the message, the extractor must begin its computation from the exact state value, x_{final}, where the embedder finished.

B. Construction of the ANStega System

The previous section established the theoretical framework for using an ANS state machine for generative steganography. However, a direct implementation faces a significant practical challenge: the state value x can grow indefinitely when processing a long message, leading to numerical overflow. This section details how we engineer ANStega, an efficient system that overcomes this limitation through a streaming architecture and targeted optimizations.

*1) **Addressing State Overflow with Streaming rANS**:* To solve the problem of unbounded state growth, we adapt our system to use Streaming rANS. This approach constrains the state value x to a fixed operational range $[L, H]$, ensuring numerical stability. Whenever an operation causes x to fall outside this range, a renormalization step is performed by streaming bits in or out. We adopt the standard configuration where $L = \alpha \cdot M$ and $H = 2L - 1$, for a hyperparameter α and $M = 2^{\text{precision}}$.

This is accomplished by integrating two core operations, Expand and Shrink, into our steganographic functions:

- **Expand (during Embedding)**: During the embedding process ($\mathsf{Encode}_{\mathcal{G}}$), a state update can cause x to become too small (i.e., $x < L$). To renormalize it, the Expand operation repeatedly shifts in bits from the secret message m into the low-order bits of the state until x is back within the $[L, H]$ range. This is the primary mechanism by which the secret message is consumed.
- **Shrink (during Extraction)**: In the extraction process ($\mathsf{Decode}_{\mathcal{G}}$), to ensure that the state variable x consistently remains within the predefined valid interval $I = [L, H]$ after each update, we introduce a preprocessing operation named Shrink. This operation is performed before each state update with ExtractStep (Equation 3). Its purpose is to pre-adjust x to guarantee that the new state value after the update will fall within I. A direct implementation would be to use a loop that speculatively calls ExtractStep to check if the next state is valid, but this is inefficient due to the repeated, costly function calls. As we prove in Appendix A-A, this complex check is equivalent to a much more efficient condition that only involves the current state x: while ($x \geq 2\alpha f_s$). Therefore, the optimized Shrink loop simply uses this fast condition. In each iteration, it outputs the least significant bit (LSB) of x and then performs a right-shift, continuing until x is small enough.

While Streaming rANS ensures stability, we introduce two further optimizations to maximize the performance and precision of ANStega.

*2) **Avoiding Rescale for Higher Fidelity**:* Traditionally, using generative models with entropy coders requires converting the model's floating-point probabilities into integer frequencies for a range $[0, M]$. This rescaling process is computationally expensive and, more importantly, introduces quantization errors that distort the original probability distribution. To eliminate this, we perform all arithmetic directly in the floating-point domain. This is achieved by normalizing the state value on the fly (i.e., using $\frac{x \pmod M}{M}$) and adapting the state update function 5 accordingly:

$$\begin{aligned} &f_s^{\text{scaled}} \leftarrow \lceil P_s \cdot M \rceil, \quad C_s^{\text{scaled}} \leftarrow \lceil C_s \cdot M \rceil \\ &x \leftarrow f_s^{\text{scaled}} \cdot \left\lfloor \frac{x}{M} \right\rfloor + (x \bmod M) - C_s^{\text{scaled}} \end{aligned} \tag{7}$$

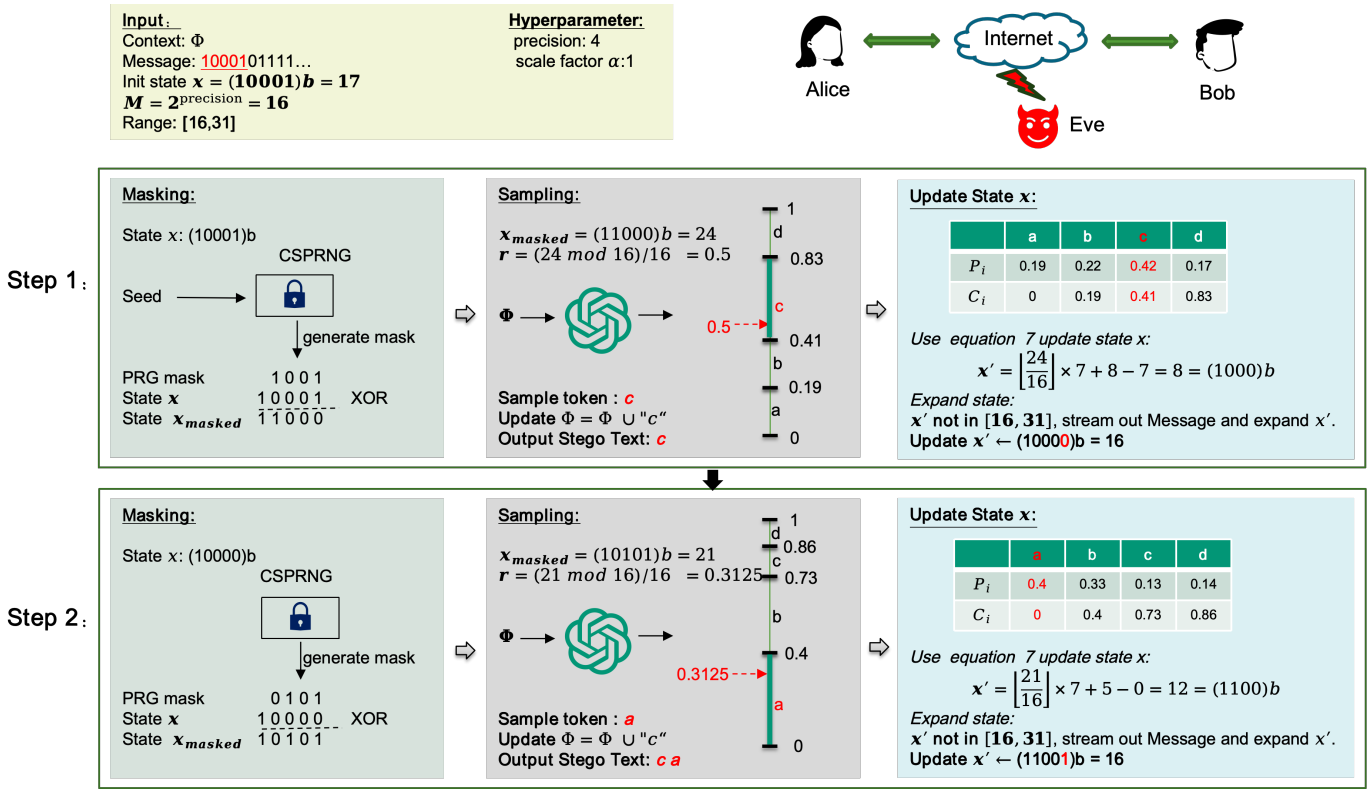

Fig. 4: An example of ANStega encoding process.

This approach preserves the model's distribution while also reducing computational overhead.

*3) **Securing Randomness with a Cryptographic Mask**:* Although the Avoiding Rescale optimisation preserves the language-model distribution exactly, the residue $r = (x \bmod M)/M$ is uniform only if the least-significant bits of the ANS state x are themselves unbiased. Increasing the normalisation factor α mitigates—but cannot eliminate—bias, and it provides no formal guarantee.

To obtain unconditional uniformity we transform ANStega into a symmetric-key scheme that injects cryptographic randomness at every generation step:

1) **Shared seed.** Sender and receiver agree on a secret key k_{seed} and seed an identical cryptographically secure pseudo-random number generator (CSPRNG).
2) **Mask Generation:** At each step t, the CSPRNG generates a precision-bit pseudo-random value, which serves as a cryptographic mask, R_t.
3) **State Scrambling:** The sampling variable r is then computed by first applying this mask to the low-order bits of the state via a bitwise XOR operation:

$$r \leftarrow \frac{(x \bmod M) \oplus R_t}{M} \tag{8}$$

where $\oplus$ denotes the bitwise XOR operation.

Because R_t is computationally indistinguishable from uniform, the resulting r is provably uniform regardless of any latent structure in x or the choice of α. Steganographic security therefore reduces to the secrecy of k_{seed} and the strength of the underlying CSPRNG, elevating ANStega's indistinguishability guarantee from heuristic to cryptographic.

*4) **Computational Simplification with Bitwise Operations**:* The rANS state update formulas rely heavily on integer division and modulo operations. These can be computationally intensive. By setting our range parameter M to be a power of two (i.e., $M = 2^{\text{precision}}$), all division and modulo operations with respect to M can be replaced with highly efficient bitwise shift and AND operations. This optimization dramatically reduces the computational load, making ANStega exceptionally fast and well-suited for deployment on resource-constrained devices.

*5) **Algorithmic Implementation**:* The mechanisms described above are implemented as a set of core algorithms. The Expand and Shrink algorithms handle the state renormalization for the streaming protocol. As shown in Algorithm 1, Expand normalizes a state $x < L$ by consuming bits from the message queue m and appends them to the state's least significant bits. Algorithm 2 preemptively adjusts the state x to ensure that the subsequent state, after performing one extraction step, remains within the interval I.

The core inverse operations are realized in Algorithm 3 and Algorithm 4. Figure 4 provides a detailed, step-by-step walkthrough of this embedding process, illustrating the state transitions and token generation. The Sample function, detailed in Algorithm 3, performs the embedding step: it uses the state x to generate a uniform random value r, samples a

token s according to the model's distribution P, and computes the new, smaller state using the rANS decoding formula. The EncodeStep function performs the inverse extraction step: it takes the received token s and current state x to compute the new, larger state using the rANS encoding formula, ensuring perfect reversibility.

Finally, Algorithms 5 and 6 orchestrate the entire process. The main embedding loop, $\mathsf{Encode}_{\mathcal{G}}$, iteratively calls Sample to generate a token and Expand to renormalize the state. The main extraction loop, $\mathsf{Decode}_{\mathcal{G}}$, processes tokens in reverse, calling EncodeStep to update the state and Shrink to renormalize and extract the hidden message bits.

Algorithm 1: $\mathsf{Expand}(x, m)$: Expand state x by consuming message bits

Input: State x, Message Queue m
Output: Expanded State x
while $x < L$ **do**
  $b \leftarrow m.\mathsf{pop\_front}()$ `// Get next bit from message`
  $x \leftarrow (x \ll 1) \mid b$ `// Append bit to LSB, increasing state value`
end
return x

Algorithm 2: $\mathsf{Shrink}(x, s, P)$: Shrink state x and extract message bits

Input: State x, Symbol s, Probability Distribution P
Output: Shrunk State x, Extracted Bits bits
Initialization: bits $\leftarrow \varnothing$
$f_s^{\text{scaled}} \leftarrow \lceil P_s \cdot M \rceil$
$x_{\max} \leftarrow 2 \cdot t \cdot f_s^{\text{scaled}}$
while $x \geq x_{\max}$ **do**
  $b \leftarrow x \bmod 2$
  bits.$\mathsf{prepend}(b)$ `// Extract LSB and prepend to bits list`
  $x \leftarrow x \gg 1$ `// Shrink state`
end
return x, bits

VI. Security Analysis

A. Security Definition

Unlike information-theoretic steganography which assumes perfect randomness [15], we analyze ANStega under the standard cryptographic model of *computational indistinguishability*. The security of our system relies on the inability of a polynomial-time adversary (Eve) to distinguish the distribution of the stego-text P_S from the cover-text distribution $P_{\mathcal{G}}$ (the language model).

Let $\mathcal{G}(\Phi, s_{<t})$ denote the true next-token distribution of the language model. Previous methods, such as those based on HC or AC, inherently modify this distribution to fit dyadic intervals or integer frequencies, resulting in a non-zero

Algorithm 3: $\mathsf{Sample}(x, P)$: Sample token and update state (Embedding Step)

Input: State x, Probability Distribution P
Output: New State x, Sampled Token s
Initialization: $C \leftarrow \mathsf{Cumulative}(P)$
$r \leftarrow (x \bmod M)/M$
$s \leftarrow \mathsf{InverseTransformSample}(C, r)$ `// Find token s where` $C_{s-1} \leq r < C_s$
`// This is the rANS decoding formula adapted for direct float probabilities`
$x \leftarrow \lfloor x/M \rfloor \times \lceil P_s \cdot M \rceil + (x \pmod M) - \lceil C_s \cdot M \rceil$
return x, s

Algorithm 4: $\mathsf{ExtractStep}(x, P, s)$: Update state with token (ANS Encoding function)

Input: State x, Distribution P, Token s
Output: New State x
Initialization: $C \leftarrow \mathsf{Cumulative}(P)$
$f_s^{\text{scaled}} \leftarrow \lceil P_s \cdot M \rceil$
$C_s^{\text{scaled}} \leftarrow \lceil C_s \cdot M \rceil$
$x \leftarrow \left\lfloor x / f_s^{\text{scaled}} \right\rfloor \cdot M + C_s^{\text{scaled}} + \left(x \bmod f_s^{\text{scaled}}\right)$
return x

Algorithm 5: $\mathsf{Encode}_{\mathcal{G}}$: Main Loop of the ANStega Embedding

Input: Message m, Context Φ, Generative Model $\mathcal{G}$, Seed k_{seed}
Output: Stegotext S, Final State x
Initialization:
$M \leftarrow 2^{\text{precision}}$, $L \leftarrow M \times \alpha$, $H \leftarrow 2L - 1$, $S \leftarrow \varnothing$
$x \leftarrow \mathsf{bits2int}(m[: \lceil \log_2(H) \rceil])$ `// Initialize state with first bits of m`
CSPRNG.Setup(k_{seed})
while generation is not finished **do**
  mask $\leftarrow$ CSPRNG.next()
  $x_{\text{mask}} \leftarrow x \oplus$ mask `// XOR the lower bits of x with mask`
  $P \leftarrow \mathcal{G}(\Phi)$
  $(x, s) \leftarrow \mathsf{Sample}(x_{\text{mask}}, P)$
  $x \leftarrow \mathsf{Expand}(x, m)$
  $S \leftarrow S \parallel s$
  $\Phi \leftarrow \Phi \parallel s$
end
return S, x

Algorithm 6: Decode$_{\mathcal{G}}$: Main Loop of the ANStega Extraction

Input: Stegotext S, Context Φ, Model $\mathcal{G}$, Final State x_{final}, Seed k_{seed}
Output: Decoded Message m
Initialization:
$M \leftarrow 2^{\text{precision}}$, $L \leftarrow M \times \alpha$, $H \leftarrow 2L - 1$, $m \leftarrow \varnothing$, $x \leftarrow x_{\text{final}}$
CSPRNG.setup(k_{seed}), mask_list = []
for each token s in S **do**
  mask $\leftarrow$ CSPRNG.next()
  mask_list.append(mask)
end
for each token s in reverse(S) **do**
  `// 1. Get distribution for the PREVIOUS step`
  $\Phi_{\text{context}} \leftarrow$ prefix of S before s
  $P \leftarrow \mathcal{G}(\Phi_{\text{context}})$
  `// 2. pre-adjust state and extract message bits`
  $(x, \text{out\_bits}) \leftarrow \text{Shrink}(x)$
  `// 3. Update state using the token (ANS encoding)`
  $x \leftarrow \text{ExtractStep}(x, P, s)$
  `// 4. Recover state x via XOR the same mask`
  mask $\leftarrow$ mask_list.pop_back()
  $x \leftarrow x \oplus$ mask
  $m \leftarrow$ out_bits $\| m$
end
Finalization:
out_bits $\leftarrow$ to_binary(x) `// The final state contains the message's first bits`
$m \leftarrow$ out_bits $\| m$
return m

Kullback-Leibler (KL) divergence ($D_{\text{KL}}(P_{\mathcal{G}}||P_S) > 0$) even with perfect randomness.

In contrast, ANStega is designed such that the *intended* sampling distribution is identical to $P_{\mathcal{G}}$. Ideally, assuming access to a True Random Number Generator (TRNG), the residue r_t used for sampling is perfectly uniform, implying $D_{\text{KL}}(P_{\mathcal{G}}||P_S) = 0$. However, in practice, we rely on a CSPRNG. Therefore, we define security via the advantage of an adversary $\mathcal{A}$ in the steganographic distinguishing game:

$$\mathbf{Adv}_{\mathcal{A}}^{\text{stega}} = |\Pr[\mathcal{A}(S) = 1] - \Pr[\mathcal{A}(C) = 1]| \tag{9}$$

where S is the stego-text sequence and C is a cover-text sequence sampled natively from the model. We claim that for any probabilistic polynomial-time (PPT) adversary $\mathcal{A}$, $\mathbf{Adv}_{\mathcal{A}}^{\text{stega}}$ is negligible, provided the underlying CSPRNG is secure.

B. Computational Security Analysis

The security proof proceeds by reduction. We show that distinguishing the stego-text from the cover-text is computationally equivalent to distinguishing the CSPRNG output from a truly random sequence.

1. Ideal Randomness Case (Information-Theoretic Baseline): Consider an idealized version of ANStega, denoted as Π_{ideal}, where the mask R_t is drawn from a uniform distribution $\mathcal{U}\{0, M-1\}$ where $M = 2^{precision}$.

As defined in Eq. (9), the sampling variable is derived via $r \leftarrow [(x \pmod M) \oplus R_t]/M$. Since R_t is perfectly uniform and independent of the state x, the One-Time Pad property ensures that the resulting bits $(x \bmod M) \oplus R_t$ are perfectly uniform over $\{0, \ldots, M-1\}$. Consequently, the normalized residue r is uniform on $[0, 1)$. Because the *Sample* algorithm uses Inverse Transform Sampling on the exact model distribution $P_{\mathcal{G}}$ driven by this uniform r, the sampled token s_t follows $P_{\mathcal{G}}$ exactly. Thus, in the ideal case:

$$P_S^{\text{ideal}}(s_t \mid s_{<t}) \equiv P_{\mathcal{G}}(s_t \mid s_{<t}) \implies D_{\text{KL}}\left(P_{\mathcal{G}} \| P_S^{\text{ideal}}\right) = 0 \tag{10}$$

2. Real-World Case (Pseudo-Randomness): In the actual ANStega protocol Π_{real}, R_t is generated by a CSPRNG seeded with key k_{seed}. If there exists a PPT adversary $\mathcal{A}$ that can distinguish P_S from $P_{\mathcal{G}}$ with non-negligible advantage ϵ, we can construct a distinguisher $\mathcal{B}$ for the CSPRNG. $\mathcal{B}$ receives a sequence of masks and runs the ANStega sampling procedure. If the masks are true random, the output follows $P_{\mathcal{G}}$ (as proven in Case 1); if they are pseudo-random, the output follows P_S. Therefore, $\mathcal{A}$'s ability to distinguish the text implies $\mathcal{B}$'s ability to distinguish the CSPRNG output from random noise.

$$\mathbf{Adv}_{\mathcal{A}}^{\text{stega}} \leq \mathbf{Adv}_{\mathcal{B}}^{\text{PRNG}} \tag{11}$$

Since $\mathbf{Adv}_{\mathcal{B}}^{\text{PRNG}}$ is negligible for a secure CSPRNG, the resulting distribution P_S is computationally indistinguishable from $P_{\mathcal{G}}$.

VII. Experiment

In this section, we empirically evaluate ANStega to address three key questions: (1) How does ANStega fundamentally perform against other entropy coding-based methods and what is the impact of hyperparameters? (2) How robust is its security under theoretical analysis and steganalysis? (3) How do its capacity and efficiency compare with the leading methods? By systematically addressing these questions, we aim to provide a holistic assessment of the superiority of ANStega as a well-balanced steganographic solution.

A. Experimental Setup

*1) **Models and Dataset**:* We evaluate ANStega on six representative generative models—GPT-2 (124M) [40], Llama 3 (8B) [41], Qwen2.5 (3B) [42], DeepSeek-R1-Distilled-Qwen (1.5B), SmolLm2(135M) [43] and Phi-2(2.7B) [44], using the IMDB [45] movie review dataset. The task involves contextual text completion, where 100 text samples are randomly selected, and the first three sentences of each are used as input context.

*2) **Baselines and Parameters**:* To ensure fair comparison, we adopt standardized settings across all methods. Two sampling strategies are used: top-k sampling [46], which selects from the k most probable tokens, and nucleus (top-p) sampling [47], which samples from the smallest set of tokens with cumulative probability above p. For precision-sensitive baselines like AC [11], we use 32-bit precision ($\beta = 32$), while methods like iMEC [15], SparSamp [19] and SA-ANS [16] retain their standard configurations (block sizes of 10, 64 and 16, respectively).

*3) **Implementation Details**:* All experiments are conducted on a high-performance computing platform equipped with an Intel Xeon Gold 6330 CPU and an NVIDIA RTX 4090 GPU. The implementation is developed using PyTorch 2.0 with CUDA 11.8 to ensure efficient and reproducible results.

B. Evaluation Metrics

To comprehensively assess ANStega and other baseline steganographic methods, we evaluate their performance on three key dimensions: capacity, efficiency, and security.

*1) **Capacity**:* We evaluate embedding capacity using two metrics. The Embedding Rate (ER) measures the average number of secret bits embedded per generated token:

$$\mathrm{ER} = \frac{\text{Encoded bits}}{\text{Generated tokens}} \tag{12}$$

The Utilized Entropy Rate (UR) normalizes capacity by comparing actual embedding to the model's theoretical limit:

$$\mathrm{UR} = \frac{\sum_{t=1}^{T} C_t}{\sum_{t=1}^{T} E_t} \tag{13}$$

where C_t is the number of bits embedded at time step t, and E_t is the model's Shannon entropy. Higher ER and UR indicate greater embedding capacity.

*2) **Efficiency**:* We assess computational efficiency through both throughput and overhead. Specifically, we use Embedding Speed (ES) and Generation Speed (GS) to measure bits and tokens processed per second:

$$\mathrm{ES} = \frac{\text{Encoded bits}}{\text{Encoding time}}, \quad \mathrm{GS} = \frac{\text{Generated tokens}}{\text{Encoding time}} \tag{14}$$

To quantify algorithmic overhead, we introduce the Sampling-to-Inference Time Ratio (SITR), which captures the relative cost of steganographic sampling:

$$\mathrm{SITR} = \frac{\text{Sampling time}}{\text{Inference time}} \tag{15}$$

An efficient method should achieve high ES/GS and low SITR.

*3) **Security**:* We evaluate security from three complementary perspectives: randomness quality, statistical fidelity, and practical undetectability.

Since our method's security relies on the indistinguishability of the cryptographic mask from uniform noise, we employ the NIST Statistical Test Suite (STS). We evaluate the bitstreams derived from the per-step sampling variates against this industry-standard battery of tests. Passing these tests confirms that the CSPRNG-driven sampling introduces no statistically detectable bias compared to a true random source.

TABLE II: Impact of α and the cryptographic mask on ANStega's performance (Llama3, Top-p, $p = 1.0$).

α	Mask	ER (bits/token)	UR	ES (bits/s)	GS (tokens/s)	SITR
1	w/o	2.03	0.86	72.60	35.63	1.26E-02
	w/	2.63	1.00	92.11	34.92	1.49E-02
4	w/o	2.44	0.97	87.92	35.94	1.26E-02
	w/	2.54	0.99	88.44	34.79	1.46E-02
16	w/o	2.37	0.97	85.70	36.03	1.27E-02
	w/	2.48	0.98	86.52	34.83	1.47E-02
64	w/o	2.65	1.02	95.22	35.87	1.28E-02
	w/	2.70	1.01	94.38	34.84	1.46E-02

Standard steganographic methods often alter the generative distribution to embed information (e.g., via probability quantization or space partitioning). To quantify this inherent statistical distortion, we compute the *algorithmic* Kullback-Leibler (KL) divergence between the model's original next-token probability distribution and the steganographic sampling distribution at each time step.

To assess real-world detectability, we measure steganalysis accuracy (Acc) by training classifiers to distinguish stego-text from cover-text:

$$\mathrm{Acc} = \frac{\text{Correct predictions}}{\text{Total predictions}} \tag{16}$$

An accuracy near 50% indicates strong resistance to detection.

C. Hyperparameter Analysis of ANStega

To thoroughly evaluate the performance and security of ANStega and to determine its optimal configuration for subsequent experiments, we conduct an in-depth analysis of two key hyperparameters: the scale factor α and the cryptographic mask. The scale factor α defines the normalization range $[L, H]$ (where $L = \alpha \cdot M$) and thus directly impacts the frequency and efficiency of state updates. The mask is a CSPRNG-derived perturbation on the low-order state used to drive sampling randomness.

Our analysis targets two questions: (1) how α and the mask affect *efficiency* and *capacity*; and (2) how they affect the *statistical randomness* of the per-step sampling—which directly impacts security.

1) Capacity and Efficiency: We first measure capacity and efficiency under varying α and with/without the mask, using Llama 3 with Top-p sampling ($p = 1.0$). The results (Table II) lead to following observations. Without the mask, ANStega attains near-entropy capacity only at larger α (e.g., UR = 0.90 at $\alpha = 1$ vs. UR = 1.00 at $\alpha = 64$), indicating that small normalization ranges distort the per-step sampling bits and depress capacity. With the mask enabled, capacity becomes essentially α-insensitive: ER and UR remain at the near-entropy frontier across settings (e.g., UR $\approx$ 0.99–1.01 for $\alpha \in \{1, 4, 16, 64\}$). Importantly, increasing α or enabling the mask does not introduce a computational burden: ES/GS and SITR stay nearly constant across all configurations.

TABLE III: NIST STS results on per-step sampling. (✓: pass, ✗: fail)

Test	**Random/Vanilla**	**ANStega**			
Mask	-	w/o			**w/**
α	-	1	16	32	1–32
ApproximateEntropy	✓	✗	✗	✗	✓
BlockFrequency	✓	✗	✗	✗	✓
CumulativeSums	✓	✗	✓	✓	✓
FFT	✓	✗	✗	✗	✓
Frequency	✓	✗	✓	✓	✓
LinearComplexity	✓	✓	✓	✓	✓
LongestRun	✓	✗	✓	✓	✓
NonOverlappingTemplate	✓	✗	✗	✗	✓
OverlappingTemplate	✓	✗	✗	✗	✓
Rank	✓	✓	✓	✓	✓
Runs	✓	✗	✓	✓	✓
Serial	✓	✗	✗	✗	✓
Universal	✓	✗	✗	✗	✓

2) Security Analysis (NIST Statistical Randomness Test): We evaluated the per-step sampling randomness using the NIST STS battery on one million bits generated from a static distribution $\{0.1, 0.2, 0.3, 0.4\}$. Table III highlights a critical distinction: the unmasked ANStega is sensitive to α, with lower values leading to increased test failures due to residual state structure. Conversely, the masked ANStega is robust to α variations, passing the full NIST suite across all configurations. These results mirror the random baseline, empirically confirming that the CSPRNG mask is essential for ensuring the generated variability is statistically indistinguishable from standard sampling.

Because security with mask is essentially α-insensitive while efficiency has already plateaued at small α, we fix $\alpha = 4$ and enable the mask for all subsequent experiments.

D. Performance Comparison against Traditional Coding Approaches

Tables IV and V contrast the four entropy coding-based steganographic methods on Llama 3 with top-k sampling.

The **Huffman-based** method suffers from significant computational overhead due to the frequent tree reconstruction required for the large vocabulary of Llama 3. Consequently, its embedding speed starts high (47.82 bits/s at $k = 8$) and collapses further to 6.93 bits/s at $k = 1024$ as shown in Table IV. Although its entropy-utilization rate (UR) exceeds 1 due to forcing token probabilities onto dyadic weights, this approximation severely distorts the model distribution. As shown in Table V, the average / max KL divergence reaches $0.72/10.60$ at $k = 8$, signaling a clear statistical fingerprint. The **AC-based** method better balances capacity and speed: UR remains near entropy ($0.98 \sim 1.02$) and ES increases with k. However, rescaling the range per-step still introduces algorithmic distortion, with KL divergence remaining in the $10^{-2} \sim 10^{-4}$ range. The concurrent work **SA-ANS** [16] adopts the ANS framework to improve capacity, achieving significantly lower KL divergence ($\sim 10^{-6}$) than AC. However, it still incurs noticeable sampling overhead (SITR ≈ 0.08) due to its state adjustment mechanism, and its embedding capacity is slightly suboptimal at lower k values (e.g., UR $= 0.89$ at k=8).

ANStega effectively overcomes these limitations. It achieves near-perfect entropy utilization (UR ≈ 1.01) across all settings, avoiding the dyadic approximation artifacts found in Huffman coding. Crucially, as detailed in Table V, ANStega yields zero *algorithmic* KL divergence because the probability distribution is not rescaled or quantized prior to sampling—surpassing both AC and SA-ANS in statistical fidelity. Leveraging constant-time rANS updates and bitwise optimizations, ANStega matches the high throughput of AC and SA-ANS while incurring the lowest sampling overhead (consistent SITR $= 0.01$). This represents an $8\times$ reduction in overhead compared to SA-ANS, securing optimal efficiency alongside provable security.

E. Comparative Analysis with Other Secure Steganographic Methods

To demonstrate that ANStega successfully solves the trilemma, we benchmarked its performance against a range of state-of-the-art, secure steganography methods. The evaluation focuses on the critical dimensions of embedding capacity and computational efficiency, where previous methods have been forced to make significant compromises. The cross-model comparison is presented in Table VI, while a detailed analysis of sampling hyperparameters (p) on Llama 3 is provided in Table VII.

The results clearly demonstrate the superiority of ANStega in terms of **embedding capability**. As an entropy-based encoding method, ANStega achieves nearly perfect entropy utilization (UR ≈ 1.0) across all tested models and sampling configurations. This allows ANStega to attain a significantly higher embedding rate (ER) than competitors. For instance, on Llama 3 with $p = 1.00$, ANStega's ER of 1.93 bits/token is substantially higher than that of SparSamp (1.77), Meteor-reorder (1.42), and iMEC (1.35). This high-capacity performance is robust across different model architectures (Table VI) and scalable with the entropy of the sampling space.

In terms of **computational efficiency**, ANStega demonstrates a decisive advantage. Its $O(1)$ complexity translates to extremely high embedding speed (ES) and a minimal sampling-to-inference time ratio (SITR). As seen in Table VI, under the rigorous setting of $p = 1.0$, ANStega's throughput consistently surpasses all other methods. While SparSamp [19] also exhibits excellent $O(1)$ efficiency with a low SITR, ANStega is unique in achieving this top-tier speed without sacrificing capacity. In stark contrast, methods like iMEC [15] and Meteor [13] incur a devastating computational cost on larger models or full sampling distributions, with SITR values orders of magnitude higher (e.g., SITR > 600 for iMEC on GPT-2), rendering them impractical for real-time applications. Thus, ANStega uniquely occupies the optimal position in the landscape of secure steganography.

TABLE IV: Capacity and efficiency comparison of entropy coding-based steganography on Llama 3.

Top-k	ES (bits/s)				UR				SITR			
	Huffman	**AC**	**SA-ANS** † [16]	**ANStega**	**Huffman**	**AC**	**SA-ANS**	**ANStega**	**Huffman**	**AC**	**SA-ANS**	**ANStega**
8	47.82	46.83	48.66	**50.39**	*1.48*	0.99	0.89	**1.01**	0.07	0.03	0.08	**0.01**
32	53.61	62.75	61.62	**63.45**	*1.36*	1.00	0.93	**1.01**	0.26	0.03	0.08	**0.01**
128	35.79	70.72	71.87	**71.57**	*1.30*	0.98	0.97	**1.01**	1.26	0.03	0.08	**0.01**
512	11.82	75.31	81.93	**83.01**	*1.26*	1.00	0.99	**1.01**	5.40	0.04	0.08	**0.01**
1024	6.93	84.62	81.61	**86.43**	*1.25*	**1.02**	0.99	**1.02**	13.91	0.04	0.08	**0.01**

† For SA-ANS, we calculate the *effective* embedding rate (per-token throughput) based on actual consumed bits rather than the theoretical parameter.

TABLE V: Comparison of algorithmic KL divergence (Ave / Max) across methods on Llama 3.

Top-k	8	128	1024
Huffman	7.23E-01 / 10.60	5.60E-01 / 7.99	5.47E-01 / 8.60
AC	7.92E-02 / 2.41	9.44E-04 / 0.07	2.86E-05 / 0.003
SA-ANS	1.72E-06/8.68E-05	2.07E-06/9.46E-05	3.05E-06/1.08E-04
ANStega	**0 / 0**	**0 / 0**	**0 / 0**

F. Resistance to Steganalysis

To complement our theoretical security guaranties, we conducted empirical steganalysis experiments to assess ANStega's practical undetectability. We constructed a balanced dataset of 20,000 samples generated under identical nucleus sampling settings ($p = 0.95$). Specifically, we generated 10,000 benign cover texts using the base model and 10,000 corresponding stego texts using ANStega. The dataset was partitioned into an 80/20 train-test split. On this data, we trained and evaluated four widely-used steganalysis architectures: a Fully Connected Network (FCN) [48], a Convolutional Neural Network (CNN) [49], and two specialized Bi-LSTM with a classification layer R-BiLSTM-C [50] and BiLSTM-Dense [51].

The results, presented in Table VIII, demonstrate ANStega's security. All four classifiers performed at a statistically equivalent level to random guessing, with accuracies hovering around the 50% mark. This indicates that even sophisticated, deep learning-based detectors were unable to find any reliable statistical artifacts in the stego-text generated by ANStega. These empirical findings provide strong practical evidence for our theoretical claim that ANStega-generated content is indistinguishable from model output.

G. Qualitative Analysis and Cross-Modal Application

In addition to quantitative metrics, it is crucial to assess the qualitative nature of the generated stego-media. To this end, we present examples that demonstrate both the fluency of ANStega's text outputs and its versatility for application in other modalities.

Table IX presents several text continuations generated by ANStega using the Llama 3 model. The examples confirm that the stego-text maintains high semantic coherence with the provided context and exhibits a natural, fluent style, rendering it qualitatively indistinguishable from benign model outputs.

To showcase ANStega's flexibility, we also applied it to image generation. Following the integration methodology of Wang et al. [19], we embedded ANStega into the final sampling step of a Denoising Diffusion Probabilistic Model (DDPM) [52] pre-trained on the FFHQ dataset. As illustrated in Figure 5, the resulting 256×256 stego-images are high-fidelity and visually coherent, demonstrating that the core principles of ANStega can be effectively adapted to different generative processes and data modalities.

VIII. Discussion and Limitations

While ANStega achieves optimal capacity, efficiency, and provable security, we acknowledge limitations inherent to high-capacity steganography.

Robustness vs. Capacity. ANStega is fragile against active attacks (e.g., token insertion or reordering); unlike robust watermarking, it requires the stego-text to be received exactly as generated. However, this fragility is a necessary trade-off for bandwidth, allowing ANStega to achieve embedding rates over $200\times$ higher than robust schemes [53]. It is best suited for reliable channels rather than copyright protection.

Passive Threat Model. We assume a passive adversary who monitors but does not modify traffic. While active interference can disrupt the channel, widespread traffic alteration imposes high operational costs on wardens. Consequently, the passive model remains the standard assumption for high-bandwidth covert communication.

Deployment Requirements. Successful decoding demands strict synchronization of models and sampling parameters; even minor floating-point non-determinism across hardware can cause errors. Standardization is required to mitigate these precision issues [5]. Additionally, as a symmetric system, ANStega requires secure initial seed exchange, though subsequent keys can be rotated via message embedding to minimize overhead.

IX. Conclusion

In this work, we addressed the long-standing Capacity-Efficiency-Security Trilemma that has constrained the field of generative steganography. We identified the root of this problem in the inherent limitations of adapting classical entropy coders like Huffman and Arithmetic Coding. To break this impasse, we introduced ANStega, a steganographic system built upon the powerful paradigm of ANS.

Our main contribution lies not only in the application of ANS, but also in the systematic reconstruction of its core mechanisms to meet the unique requirements of steganography. Through innovative reverse application of state machines, customized streaming architecture, and key cryptographic masking mechanisms, we have designed a system that

TABLE VI: Capacity and efficiency comparison on different models (Top-p, $p = 1.0$).

Metrics	Models	ADG [12]	iMEC [15]	Meteor [13]		Discop [14]		SparSamp [19]	ANStega
				base	reorder	base	huffman		
ER (bits/token)	GPT-2	5.25	4.53	4.67	5.73	2.40	5.78	5.67	5.95
	Qwen2.5	2.80	2.88	2.67	3.67	1.53	3.73	3.48	3.69
	Llama 3	1.63	0.70	1.62	1.67	1.06	2.19	2.37	2.56
	DeepSeek	2.57	2.33	2.57	3.20	1.48	3.41	3.50	3.59
	SmolLm2	4.43	3.46	2.01	4.42	4.10	5.12	5.12	5.50
	Phi-2	3.56	2.57	1.76	3.99	3.30	4.40	4.32	4.67
UR	GPT-2	0.85	0.70	0.77	0.89	0.38	0.95	0.93	**1.00**
	Qwen2.5	0.73	0.68	0.70	0.83	0.40	0.93	0.97	**1.01**
	Llama 3	0.63	0.73	0.66	0.72	0.42	0.90	0.97	**1.00**
	DeepSeek	0.69	0.73	0.70	0.83	0.39	0.92	0.96	**0.99**
	SmolLm2	0.81	0.68	0.38	0.84	0.75	0.94	0.94	**1.01**
	Phi-2	0.76	0.68	0.38	0.86	0.73	0.95	0.93	**1.01**
ES (bits/s)	GPT-2	7.82	0.91	636.57	0.84	155.79	22.42	821.56	**879.79**
	Qwen2.5	3.46	0.05	95.74	0.5	29.94	4.76	127.18	**136.82**
	Llama 3	1.61	0.02	57.10	0.41	22.35	3.31	84.28	**94.67**
	DeepSeek	1.82	0.04	115.88	0.31	32.97	4.62	157.14	**169.51**
	SmolLm2	6.61	0.50	51.32	0.60	137.10	15.77	187.82	**212.31**
	Phi-2	7.30	0.35	48.07	0.83	128.59	13.06	173.12	**186.81**
GS (tokens/s)	GPT-2	1.49	0.20	136.45	0.15	65.01	3.88	144.69	**147.76**
	Qwen2.5	1.24	0.02	35.65	0.14	19.60	1.28	36.52	**37.05**
	Llama 3	0.98	0.03	35.16	0.25	21.02	1.51	35.53	**36.91**
	DeepSeek	0.71	0.02	45.09	0.10	22.29	1.32	46.06	**47.17**
	SmolLm2	1.48	0.14	25.47	0.13	33.39	3.07	36.62	**38.56**
	Phi-2	2.04	0.13	27.28	0.20	38.94	2.96	**40.01**	39.98
SITR	GPT-2	93.75	640.56	0.11	730.36	1.16	30.81	**0.03**	0.04
	Qwen2.5	28.02	1891.82	0.03	245.61	0.77	24.67	**0.01**	**0.01**
	Llama 3	35.51	1151.72	0.03	129.53	0.65	20.86	**0.01**	**0.01**
	DeepSeek	59.77	1774.04	0.03	424.71	0.97	29.52	**0.01**	**0.01**
	SmolLm2	22.77	187.50	0.33	162.29	0.03	10.27	**0.01**	**0.01**
	Phi-2	18.07	261.24	0.40	182.69	0.03	11.94	**0.01**	**0.01**

TABLE VII: Capacity and efficiency comparison on Llama 3 (Top-p).

Metrics	p	ADG [12]	iMEC [15]	Meteor [13]		Discop [14]		SparSamp [19]	ANStega
				base	reorder	base	huffman		
ER (bits/token)	0.80	0.64	0.85	0.66	0.73	0.72	1.01	0.90	1.20
	0.95	1.14	1.35	1.22	1.42	0.95	1.69	1.77	1.93
	1.00	1.63	0.70	1.62	1.67	1.06	2.19	2.37	2.56
UR	0.80	0.52	0.86	0.55	0.61	0.58	0.85	0.99	**1.00**
	0.95	0.60	0.79	0.64	0.73	0.48	0.87	0.99	**1.01**
	1.00	0.63	0.73	0.66	0.72	0.42	0.90	0.97	**1.00**
ES (bits/s)	0.80	21.45	26.31	22.00	23.47	23.90	34.53	30.17	**43.98**
	0.95	37.51	39.42	40.37	33.18	31.36	55.00	59.20	**64.21**
	1.00	1.61	0.02	57.10	0.41	22.35	3.31	84.28	**94.67**
GS (tokens/s)	0.80	33.43	31.10	33.09	32.16	33.07	34.15	33.21	**33.92**
	0.95	32.95	29.19	33.06	23.32	32.94	32.62	33.29	**33.31**
	1.00	0.98	0.03	35.16	0.25	21.02	1.51	35.53	**36.91**
SITR	0.80	**0.01**	0.08	0.02	0.05	0.02	**0.01**	**0.01**	**0.01**
	0.95	0.02	0.15	0.02	0.46	0.02	0.02	**0.01**	**0.01**
	1.00	35.51	1151.72	0.03	129.53	0.65	20.86	**0.01**	**0.01**

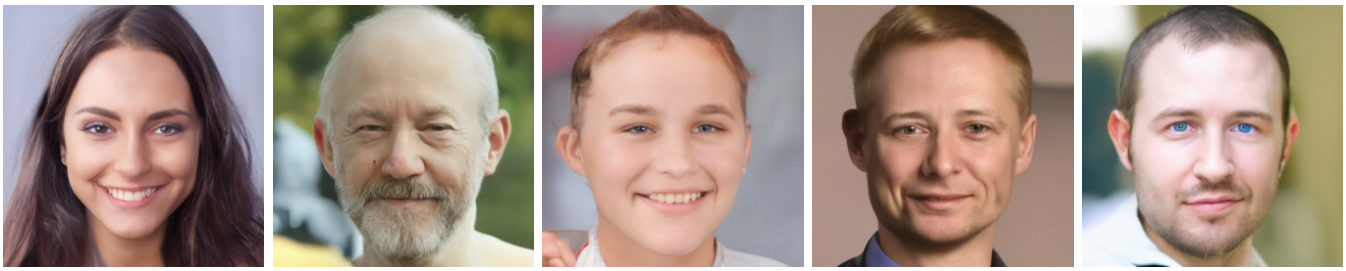

Fig. 5: Representative 256×256 stego-faces synthesized by a DDPM equipped with ANStega. Despite their natural appearance, each image conceals roughly 463 kbits of payload.

TABLE VIII: Steganalysis accuracy against ANStega.

	FCN	CNN	R-BiLSTM-C	BiLSTM-Dense
Acc	49.92%	50.28%	49.35%	50.15%

TABLE IX: Context and generation examples.

Context	ANStega Generation
Beneath the quiet sky, dreams wander like fireflies, glowing softly in the silence of hope.	Yet, in the stillness of the night, what lies hidden, unseen by mortal eyes, is a dance of shadows, an art of darkness, an eternal sway of the pierced heart.
Even the darkest night cannot extinguish a heart that burns with purpose.	Every experience, no matter how difficult, is an opportunity to learn, grow, and rise above the challenges that we face. Courage and perseverance can help us overcome the impossible and lead us to new heights of success and fulfillment.
Titanic is not just a love story set against tragedy—it's a cinematic elegy to fleeting beauty, human hubris, and the moments that define a lifetime.	James Cameron's sweeping epic is a masterclass in filmmaking, a testament to the power of storytelling, and a reminder that even the most epic of tragedies can be told with a disparate array of miniatures, models, and digital wizardry.

successfully balances the three competing goals. Our extensive theoretical analysis and empirical evaluation demonstrate that ANStega is the first method to simultaneously achieve the following three objectives: (1) near-optimal embedding capacity, (2) constant-time $O(1)$ computational efficiency, and (3) provable security. Beyond raw metrics, ANStega preserves the stylistic fidelity of natural text and yields visually coherent diffusion images, demonstrating its applicability across modalities. These results position ANStega provides both a practical, high-performance tool for secure communication and a new, powerful paradigm for steganographic research. We hope that our work will provide valuable insights for the next generation of secure, efficient, and high-capacity covert communication systems.

ACKNOWLEDGMENT

The authors thank the reviewers for their valuable comments. This work was supported in part by the Natural Science Foundation of China under Grant 62302146, 62472398, and U2336206.

REFERENCES

[1] U. S. Congress, "Kids online safety act," https://www.congress.gov/bill/118th-congress/house-bill/7891, April 2024.

[2] U. K. Parliament, "Online safety act 2023," https://bills.parliament.uk/bills/3137, 2023.

[3] D. Cole, "We kill people based on metadata," https://www.nybooks.com/online/2014/05/10/we-kill-people-based-metadata/, May 2014.

[4] G. J. Simmons, "The prisoners' problem and the subliminal channel," in *Annual International Cryptology Conference*, 1983. [Online]. Available: https://api.semanticscholar.org/CorpusID:2674577

[5] L. A. Bauer, J. K. Howes, S. A. Markelon, V. Bindschaedler, and T. Shrimpton, "Leveraging generative models for covert messaging: Challenges and tradeoffs for "dead-drop" deployments," in *Proceedings of the Fourteenth ACM Conference on Data and Application Security and Privacy*, ser. CODASPY '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 67–78. [Online]. Available: https://doi.org/10.1145/3626232.3653264

[6] 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, 2019.

[7] F. Dai and Z. Cai, "Towards near-imperceptible steganographic text," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, A. Korhonen, D. Traum, and L. Màrquez, Eds. Florence, Italy: Association for Computational Linguistics, Jul. 2019, pp. 4303–4308. [Online]. Available: https://aclanthology.org/P19-1422/

[8] T. Van Le and K. Kurosawa, "Bandwidth optimal steganography secure against adaptive chosen stegotext attacks," in *Information Hiding*, J. L. Camenisch, C. S. Collberg, N. F. Johnson, and P. Sallee, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 297–313.

[9] K. Yang, K. Chen, W. Zhang, and N. Yu, "Provably secure generative steganography based on autoregressive model," in *International Workshop on Digital Watermarking*, 2018. [Online]. Available: https://api.semanticscholar.org/CorpusID:59524394

[10] K. Chen, H. Zhou, H. Zhao, D. Chen, W. Zhang, and N. Yu, "When provably secure steganography meets generative models," 2019. [Online]. Available: https://arxiv.org/abs/1811.03732v2

[11] Z. Ziegler, Y. Deng, and A. M. Rush, "Neural linguistic steganography," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019, pp. 1210–1215.

[12] S. Zhang, Z. Yang, J. Yang, and Y. Huang, "Provably secure generative linguistic steganography," in *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, C. Zong, F. Xia, W. Li, and R. Navigli, Eds. Online: Association for Computational Linguistics, Aug. 2021, pp. 3046–3055. [Online]. Available: https://aclanthology.org/2021.findings-acl.268

[13] G. Kaptchuk, T. M. Jois, M. Green, and A. D. Rubin, "Meteor: Cryptographically secure steganography for realistic distributions," *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, 2021. [Online]. Available: https://api.semanticscholar.org/CorpusID:235309313

[14] J. Ding, K. Chen, Y. Wang, N. Zhao, W. Zhang, and N. H. Yu, "Discop: Provably secure steganography in practice based on "distribution copies"," *2023 IEEE Symposium on Security and Privacy (SP)*, pp. 2238–2255, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:260002610

[15] C. S. de Witt, S. Sokota, J. Z. Kolter, J. N. Foerster, and M. Strohmeier, "Perfectly secure steganography using minimum entropy coupling," in *The Eleventh International Conference on Learning Representations*, 2023. [Online]. Available: https://openreview.net/forum?id=HQ67mj5rJdR

[16] Y. Liu, C. Xu, F. Yang, P. Zhang, and L. Wang, "Linguistic steganography via self-adjusting asymmetric number system," *Computational Linguistics*, pp. 1–35, 09 2025. [Online]. Available: https://doi.org/10.1162/coli.a.22

[17] G. Liao, J. Yang, W. Shao, and Y. Huang, *A framework for designing provably secure steganography*. USA: USENIX Association, 2025.

[18] M. Bai, K. Pang, G. Liao, J. Yang, and Y. Huang, "Shimmer: a provably secure steganography based on entropy collecting mechanism," in *Proceedings of the 34th USENIX Conference on Security Symposium*, ser. SEC '25. USA: USENIX Association, 2025.

[19] Y. Wang, G. Pei, K. Chen, J. Ding, C. Pan, W. Pang, D. Hu, and W. Zhang, "SparSamp: Efficient provably secure steganography based on sparse sampling," in *34th USENIX Security Symposium (USENIX Security '25)*. USENIX Association, Aug. 2025.

[20] T. M. Jois, G. Beck, and G. Kaptchuk, "Pulsar: Secure steganography for diffusion models," in *Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security*, ser. CCS '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 4703–4717. [Online]. Available: https://doi.org/10.1145/3658644.3690218

[21] Y. Peng, D. Hu, Y. Wang, K. Chen, G. Pei, and W. Zhang, "Stegaddpm: Generative image steganography based on denoising diffusion probabilistic model," in *Proceedings of the 31st ACM International Conference on Multimedia*, ser. MM '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 7143–7151. [Online]. Available: https://doi.org/10.1145/3581783.3612514
[22] A. D. Ker, "Improved detection of lsb steganography in grayscale images," in *International workshop on information hiding*. Springer, 2004, pp. 97–115.
[23] M. Boroumand, M. Chen, and J. Fridrich, "Deep residual network for steganalysis of digital images," *IEEE Transactions on Information Forensics and Security*, vol. 14, no. 5, pp. 1181–1193, 2019.
[24] R. Anderson and F. Petitcolas, "On the limits of steganography," *IEEE Journal on Selected Areas in Communications*, vol. 16, no. 4, pp. 474–481, 1998.
[25] J. Duda, "Asymmetric numeral systems," 2009. [Online]. Available: https://arxiv.org/abs/0902.0271
[26] https://github.com/facebook/zstd.
[27] https://github.com/lzfse/lzfse.
[28] https://github.com/google/pik.
[29] D. E. Gladding, S. Gopalakrishnan, S. D. Kumbhani, and L. Hsu-Kuei, "Features of range asymmetric number system encoding and decoding," Jan. 25 2022, uS Patent 11,234,023.
[30] G. J. Simmons, "The prisoners' problem and the subliminal channel," in *Advances in Cryptology: Proceedings of Crypto 83*. Springer, 1984, pp. 51–67.
[31] N. J. Hopper, J. Langford, and L. von Ahn, "Provably secure steganography," in *Advances in Cryptology — CRYPTO 2002*, M. Yung, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2002, pp. 77–92.
[32] T. Fang, M. Jaggi, and K. Argyraki, "Generating steganographic text with lstms," in *Proceedings of ACL 2017, Student Research Workshop*, 2017, pp. 100–106.
[33] 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, 2019.
[34] G. Kaptchuk, T. M. Jois, M. Green, and A. D. Rubin, "Meteor: Cryptographically secure steganography for realistic distributions," in *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, 2021, pp. 1529–1548.
[35] S. Zhang, Z. Yang, J. Yang, and Y. Huang, "Provably secure generative linguistic steganography," in *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, 2021, pp. 3046–3055.
[36] J. Duda, "Asymmetric numeral systems: entropy coding combining speed of huffman coding with compression rate of arithmetic coding," *arXiv preprint arXiv:1311.2540*, 2013.
[37] Y. Collet and M. Kucherawy, "Zstandard Compression and the application/zstd Media Type," RFC 8478, Oct. 2018. [Online]. Available: https://www.rfc-editor.org/info/rfc8478
[38] J. Sneyers, J. Alakuijala, L. Versari, Z. Szabadka, S. Boukortt, A. Cohen-Tidhar, M. Firsching, E. Kliuchnikov, T. Lev-Ami, E. Portis, T. Richter, and O. Watanabe, "The jpeg xl image coding system: History, features, coding tools, design rationale, and future," 2025. [Online]. Available: https://arxiv.org/abs/2506.05987
[39] T. Cover and J. Thomas, *Elements of information theory*. Wiley-Interscience, 2006.
[40] https://huggingface.co/openai-community/gpt2.
[41] https://huggingface.co/meta-llama/Meta-Llama-3-8B-Instruct.
[42] https://huggingface.co/Qwen/Qwen2.5-3B-Instruct.
[43] https://huggingface.co/HuggingFaceTB/SmolLM2-135M.
[44] https://huggingface.co/microsoft/phi-2.
[45] https://ai.stanford.edu/~amaas/data/sentiment/.
[46] A. Holtzman, J. Buys, M. Forbes, A. Bosselut, D. Golub, and Y. Choi, "Learning to write with cooperative discriminators," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, I. Gurevych and Y. Miyao, Eds. Melbourne, Australia: Association for Computational Linguistics, Jul. 2018, pp. 1638–1649. [Online]. Available: https://aclanthology.org/P18-1152
[47] A. Holtzman, J. Buys, L. Du, M. Forbes, and Y. Choi, "The curious case of neural text degeneration," in *International Conference on Learning Representations*, 2020. [Online]. Available: https://openreview.net/forum?id=rygGQyrFvH
[48] Z. Yang, Y. Huang, and Y.-J. Zhang, "A fast and efficient text steganalysis method," *IEEE Signal Processing Letters*, vol. 26, no. 4, pp. 627–631, 2019.
[49] ——, "Ts-csw: text steganalysis and hidden capacity estimation based on convolutional sliding windows," *Multimedia Tools Appl.*, vol. 79, no. 25–26, p. 18293–18316, Jul. 2020. [Online]. Available: https://doi.org/10.1007/s11042-020-08716-w
[50] Y. Niu, J. Wen, P. Zhong, and Y. Xue, "A hybrid r-bilstm-c neural network based text steganalysis," *IEEE Signal Processing Letters*, vol. 26, no. 12, pp. 1907–1911, 2019.
[51] H. Yang, Y. Bao, Z. Yang, S. Liu, Y. Huang, and S. Jiao, "Linguistic steganalysis via densely connected lstm with feature pyramid," *Proceedings of the 2020 ACM Workshop on Information Hiding and Multimedia Security*, 2020. [Online]. Available: https://api.semanticscholar.org/CorpusID:219980226
[52] J. Choi, S. Kim, Y. Jeong, Y. Gwon, and S. Yoon, "Ilvr: Conditioning method for denoising diffusion probabilistic models," in *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021, pp. 14 347–14 356.
[53] M. Bai, J. Yang, K. Pang, X. Xu, Z. Yang, and Y. Huang, "Provably robust and secure steganography in asymmetric resource scenario," in *2025 IEEE Symposium on Security and Privacy (SP)*, 2025, pp. 1438–1456.

APPENDIX A
NORMALIZATION MECHANISM OF THE STREAMING RANS ENCODER

A. Derivation of the Normalization Condition

A fundamental requirement of the rANS encoder is that after encoding any symbol s, the new state $\mathcal{C}(s,x)$ must remain within the normalization interval I. This can be expressed as: $L \leq \mathcal{C}(s,x) < 2L$. To avoid a computationally expensive trial-and-error loop to enforce this condition, we derive an equivalent but far simpler direct condition on x. The derivation proceeds as follows. First, we substitute the definitions of $\mathcal{C}(s,x)$ and $L = \alpha \times M$ into inequality (1):

$$\alpha \times M \leq \left\lfloor \frac{x}{f_s} \right\rfloor + C_s + (x \bmod f_s) < 2 \times \alpha \times M \quad (17)$$

Dividing all parts of the inequality by M yields:

$$\alpha \leq \left\lfloor \frac{x}{f_s} \right\rfloor + [C_s + (x \bmod f_s)]/M < 2\alpha \quad (18)$$

Let us analyze the fractional term $[C_s + (x \bmod f_s)]/M$. By definition, $C_s + f_s \leq M$. Since $0 \leq (x \bmod f_s) < f_s$, it follows that $0 \leq C_s + (x \bmod f_s) < C_s + f_s \leq M$. Therefore, the value of the fractional term is bounded within the range $[0, 1)$.

$$0 \leq [C_s + (x \bmod f_s)]/M < 1 \quad (19)$$

Given that $\lfloor x/f_s \rfloor$ is an integer and the fractional term is in $[0, 1)$, the inequality18 can be simplified by considering only the integer parts:

$$\alpha \leq \left\lfloor \frac{x}{f_s} \right\rfloor < 2\alpha \quad (20)$$

Based on the properties of the floor function, inequality (5) is equivalent to:

$$\alpha \leq \frac{x}{f_s} < 2\alpha \quad (21)$$

Finally, multiplying all parts by f_s gives the direct constraint on the state x:

$$\alpha f_s \leq x < 2\alpha f_s \tag{22}$$

APPENDIX B
ARTIFACT APPENDIX

This appendix provides a self-contained guide for evaluating the artifact for this paper.

A. Description & Requirements

This section details the necessary components to set up the experimental environment and run the artifact.

1) How to access: The artifact is provided as a supplementary .zip file. A permanent, archived version is publicly accessible via the Zenodo repository at https://doi.org/10.5281/zenodo.17943483.

2) Hardware dependencies:

- CPU: A modern multi-core CPU (e.g., Intel Xeon Gold 6330, as used in the paper).
- GPU: An NVIDIA GPU with sufficient VRAM is required for running the language models. An NVIDIA RTX 4090 (24GB) was used for the paper's experiments. A GPU with at least 16GB of VRAM is recommended for the larger models (Llama 3 8B). The gpt2 model can be run on smaller-VRAM GPUs.

3) Software dependencies:

- Python ≥ 3.10
- PyTorch (CUDA 11.8 version is specified in `requirements.txt`)
- All other Python dependencies are listed in `requirements.txt`.

4) Benchmarks:

- **Models:** The artifact is designed to work with Hugging Face-compatible causal language models. The paper evaluates:
  - GPT-2 (`openai-community/gpt2`)
  - Llama 3 8B (`meta-llama/Meta-Llama-3-8B-Instruct`)
  - Qwen2.5 3B (`Qwen/Qwen2.5-3B-Instruct`)
  - DeepSeek-R1-Distilled-Qwen 1.5B (`deepseek-ai/DeepSeek-R1-Distill-Qwen-1.5B`)
  - SmolLm2 (`HuggingFaceTB/SmolLM2-135M`)
  - Phi-2 (`microsoft/phi-2`)

  The scripts will automatically download `gpt2` if no local path is provided. Other models must be downloaded by the user from Hugging Face.
- **Datasets:**
  - `imdb_context.xlsx`: (Provided) Contains 1,000 contexts from the IMDB movie review dataset. The batch script uses the first 100 entries for evaluation.
  - `message.txt`: (Provided) A binary string used as the secret message payload.

B. Artifact Installation & Configuration

[Estimated time: 10-15 minutes, plus model download time]

1) **Unzip the artifact:**
   ```
   unzip anstega_artifact.zip
   cd anstega_artifact
   ```
2) **Create a Python virtual environment:**
   ```
   python3 -m venv venv
   source venv/bin/activate
   ```
3) **Install dependencies:** Install all required Python packages using the provided `requirements.txt` file.
   ```
   pip install -r requirements.txt
   ```
4) **Model Preparation:**
   - **Option 1 (Quick Test):** No action needed. The scripts `run_single.py` and `run_batch.py` will default to loading `openai-community/gpt2` from Hugging Face if the specified `model_name` path is not found.
   - **Option 2 (Full Reproduction):** To reproduce results for other models (e.g., Llama 3), manually download the model from Hugging Face and save it to a local directory. You will then need to update the `model_name` variable in `run_single.py` or `run_batch.py` to point to this local path (e.g., `model_name = './LLaMA3_8B/'`).

C. Experiment Workflow

The evaluation workflow is straightforward:

1) **Installation:** Set up the environment and dependencies as described in Section B.
2) **Functional Test:** Run `run_single.py` to perform a single encoding and decoding pass. This verifies that the core ANStega algorithm is working correctly.
3) **Performance Evaluation:** Run `run_batch.py` to execute the algorithm over 100 different contexts. This script computes and reports the average performance metrics (ER, UR, ES, GS, SITR) that directly correspond to the results presented in the paper.

D. Major Claims

This artifact supports the following major claims from the paper:

- **(C1) Optimal Capacity:** ANStega achieves near-optimal embedding capacity, approaching the Shannon entropy limit (Utilization Rate, UR $\approx$ 1.0) and outperforming prior secure methods. This is proven by experiment (E2) and corresponds to the 'ER' (bits/token) and 'UR' metrics in **Tables V and VI**.
- **(C2) Optimal Efficiency:** ANStega has $O(1)$ embedding complexity, resulting in high Embedding Speed (ES), Generation Speed (GS), and a minimal Sampling-to-Inference Time Ratio (SITR) that is competitive with AC-based methods and vastly superior to other provably secure methods. This is proven by experiment (E2) and

corresponds to the 'ES', 'GS', and 'SITR' metrics in **Tables IV, V, and VI**.

E. Evaluation

This section provides the operational steps to validate the artifact and reproduce the paper's key results.

Experiment (E1): Functional & Correctness Test: [∼5 human-minutes + <5 compute-minutes]

This experiment validates the correctness of the ANStega implementation by encoding a message and verifying that it can be perfectly decoded.

- **[Preparation]**
  1) Ensure the environment is set up (Section B).
  2) No other preparation is needed. The script `run_single.py` is configured to use `gpt2` by default if no other model is found.
- **[Execution]** Run the single test script:
  `python run_single.py`
- **[Results]** The script will:
  1) Print its configuration (token_num, precision, etc.).
  2) Print the auto-selected device (e.g., `cuda`).
  3) Print the generated stego text.
  4) Print a `result_dict` with the performance metrics (ER, UR, ES, GS, SITR) for this single run.
  5) Crucially, it will end by printing:
     `success!`

  This "success!" message confirms that the `outbits_list` from decoding perfectly matched the `encoded_message` from encoding.

Experiment (E2): Performance & Capacity Evaluation (Batch): [∼5 human-minutes + 30-90 compute-minutes, depending on GPU/model]

This experiment reproduces the core performance metrics (ER, UR, ES, GS, SITR) presented in **Tables IV, V, and VI** by running a batch evaluation over 100 contexts.

- **[Preparation]**
  1) Ensure the environment is set up (Section B).
  2) Ensure the `imdb_context.xlsx` and `message.txt` files are in the same directory.
  3) (Optional but recommended) To reproduce the paper's main results for larger models, download the Qwen2.5-3B-Instruct model (or other models like Qwen) from Hugging Face... and edit `run_batch.py`. Change the model_name variable to point to its local path:
     `model_name = '/path/to/your/model/'`
- **[Execution]** Run the batch test script:
  `python run_batch.py`
  A `tqdm` progress bar will show the progress as it iterates through the 100 contexts.
- **[Results]** After the batch run is complete, the script will print a final `result_dict` containing the *average* metrics over all 100 runs. Example (values will vary based on hardware and model): These values correspond to the metrics for **Claims C1 and C2**. The evaluator can compare these generated values directly with the rows for ANStega in Tables IV, V, and VI of the paper (for the corresponding model). *Note: Exact speed metrics (ES, GS) will vary based on hardware, but the ER and UR values should be very close to the paper's reported results.*

F. Customization

The experiments can be customized by modifying the parameters at the top of `run_single.py` and `run_batch.py`.

- `model_name`: Path to the local Hugging Face model directory.
- `seed`: Random seed for reproducibility.
- `precision`: ANS precision parameter.
- `ans_t`: ANS scale factor (referred to as α in the paper, Table II).
- `top_p`: Nucleus sampling (top-p) value.
- `token_num`: Number of tokens to generate.