Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Parallel Accelerated Steganographic Polar Coding Architecture Based on GPU

Xiaolei He
University of Chinese Academy of Sciences
Luyao Wang
University of Chinese Academy of Sciences
Haocheng Fu
fuhaocheng@iie.ac.cn
University of Chinese Academy of Sciences
Yun Cao
University of Chinese Academy of Sciences
Meineng Zhu
University of International Relations

Research Article

Keywords: covert communication, steganographic codes, polar codes, successive cancellation

Posted Date: July 3rd, 2026

DOI: https://doi.org/10.21203/rs.3.rs-9630729/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

Version of Record: A version of this preprint was published at Journal on Information Security on October 14th, 2026. See the published version at https://doi.org/10.1186/s13635-026-00249-3.

Parallel Accelerated Steganographic Polar Coding Architecture Based on GPU

Xiaolei He[1,2], Luyao Wang[1,2], Haocheng Fu[1,2*], Yun Cao[1,2], Meineng Zhu[3]

[1*]Institute of Information Engineering, Chinese Academy of Sciences,Beijing,100093,China.
[2]School of Cyber Security, University of Chinese Academy of Sciences,Beijing,100049,China.
[3]School of Cybersecurity, University of International Relations,Beijing,100091,China.

*Corresponding author(s). E-mail(s): fuhaocheng@iie.ac.cn;
Contributing authors: hexiaolei@iie.ac.cn; wangluyao@iie.ac.cn; caoyun@iie.ac.cn; zmneng@uir.edu.cn;

Abstract

High-throughput steganographic coding remains difficult because syndrome-trellis codes (STC) incur high computational cost at practical security settings, while steganographic polar codes retain strong serial dependencies in their successive decoding schedule. This paper proposes CuSPC, a GPU-oriented steganographic polar coding architecture that restructures simplified successive cancellation (SSC) execution, introduces an ordinary-node (Ord) design for short residual subtrees, and combines inter-cover parallelism with interleaved global-memory access. The implementation maps intra-node, inter-node, and inter-cover parallelism to the CUDA execution model while preserving the distortion-minimizing formulation of polar steganographic coding. Experiments show that CuSPC reaches up to 30 Mb/s and improves throughput by about 30 times over the CPU serial baseline, while keeping the coding loss below 2% and maintaining steganalysis performance comparable to mainstream baselines.

Keywords: covert communication, steganographic codes, polar codes, successive cancellation

1 Introduction

As a covert communication technique, the core objective of steganography is to embed secret information into digital media without arousing the suspicion of third parties [1, 2]. With the rapid development of digital imaging and multimedia technologies, the mainstream paradigm of steganography has shifted from the early focus on preserving high-order statistical features to a framework of minimal embedding distortion. Within this framework, the sender relies on a designed distortion function [3–8] to assign modification costs to cover elements, modeling the steganographic process as a combinatorial optimization problem of minimizing a constrained additive distortion. As the core algorithm for information embedding, the steganographic encoder is responsible for finding the optimal modification path for the cover under a given distortion distribution. Consequently, the encoding algorithm not only directly determines the system?s upper bound of security for a specific payload but also serves as the crucial link between the theoretical distortion function and the actual modification of the cover.

Steganographic coding has evolved to approach optimal embedding efficiency under tight distortion constraints. Early steganographic coding schemes were primarily based on linear channel code design and were only applicable to scenarios with constant distortion distributions where all modification positions had the same cost [9, 10]. However, the complex local textures and strong statistical correlations of natural media covers result in modification costs exhibiting high spatial non-uniformity. To overcome this limitation, wet paper coding (WPC) [11, 12] was introduced into the steganography framework, effectively resolving the binary distortion constraint where modification costs are either non-zero or infinite. The emergence of content-adaptive steganography has extended distortion costs to positive real numbers, challenging steganographic coding to find the optimal trade-off between embedding capacity and total distortion under continuous distributions.

The syndrome-trellis code (STC) [13, 14] is the first near-optimal code based on the Viterbi algorithm and has been widely integrated into mainstream steganographic schemes over the past decade due to its flexible applicability and capacity to approach the rate-distortion bound. However, the long-standing dominance of STC has also led to new technical bottlenecks. STC is essentially a Viterbi decoder for convolutional codes. Although its decoding process can be accelerated through block-based parallel strategies, to strictly approach the rate-distortion bound, STC requires a large constraint length (h), which leads to an exponential expansion of the state space ($\mathcal{O}(2^h)$). This creates an irreconcilable conflict between embedding efficiency and real-time throughput. In traditional small-scale digital image steganography, such millisecond-level delays are tolerable. However, in emerging covert communication systems designed for the high-speed network and high-definition streaming era, systems must maintain extremely high anti-analysis capabilities while delivering high throughput to avoid communication latency. In this context, the exponentially increasing computational overhead of STC becomes a barrier to technological implementation.

The introduction of polar codes [15] provides a theoretical path around this bottleneck. Unlike STC, steganographic polar codes (SPC) [16–19] can approach the rate-distortion bound with logarithmic-linear decoding complexity ($\mathcal{O}(N \log N)$).

However, this algorithmic advantage does not directly translate into high throughput in practice. SPC encoding still relies on a depth-first traversal of the polar tree, and the resulting data dependencies limit concurrency on conventional central processing units (CPUs). Existing CPU-side optimizations, such as segmented parallelism [20] and sub-polarized steganographic codes [18], improve throughput only partially because thread-level scalability and memory-access efficiency remain limited.

To overcome these computational bottlenecks, we can draw valuable cross-disciplinary insights from the hardware acceleration of standard polar codes. Since standard polar codes have been established as the coding scheme for 5G control channels, the communications community has conducted extensive research into low-level optimizations for their execution efficiency. Although steganographic polar codes serve a completely different purpose, they share the same underlying mathematical topology. Motivated by the maturation of the simplified successive cancellation (SSC) algorithm and its evolved variants [21, 22], high-performance polar decoders based on graphics processing units (GPU) [23–25] have emerged, successfully overcoming the serial latency of the polar tree. The single instruction multiple thread (SIMT) model inherent to GPU architectures shares a natural topological isomorphism with the concurrent computation requirements of the vast number of independent nodes in the polar tree structure. Unlike standard communication protocols, multimedia steganography processes cover sequences that are substantially longer than typical polar code blocks. This massive scale makes GPUs highly advantageous for exploiting large-scale data-level parallelism (DLP), thereby significantly boosting throughput.

Motivated by this gap, this paper proposes CuSPC, a GPU-oriented steganographic polar coding architecture for heterogeneous computing. Built on the SSC algorithm [21, 22], CuSPC restructures the execution schedule of the polar tree for steganographic coding rather than treating GPU acceleration as a direct implementation add-on. It combines fine-grained node-level parallelism with coarse-grained inter-cover scheduling so that the decoding structure of SPC can be executed efficiently on the CUDA SIMT model. The main contributions of this paper are as follows:

1. We reformulate SSC execution for GPU-oriented steganographic coding by exposing both intra-node and inter-node parallelism in the polar tree. To reduce scheduling overhead on short residual subtrees, we introduce an ordinary-node (Ord) design that truncates recursive traversal and evaluates local candidates in parallel.
2. We propose an inter-cover scheduling mechanism tailored to the SIMT execution model. By combining cover segmentation with interleaved memory layout, the proposed architecture improves global-memory coalescing and increases hardware occupancy for large covers.
3. We implement an end-to-end CuSPC encoder and evaluate it on representative devices. The proposed design reaches up to 30 Mb/s, improves throughput by about 30× over the CPU serial baseline, and preserves near-baseline embedding efficiency and steganalysis performance.

The remainder of this paper is organized as follows. Section 2 reviews optimal embedding and the polar-coding background. Sections 3 and 4 present the proposed

Table 1 Main symbols used in the steganographic and polar-coding formulations.

Symbol	Domain	Meaning
$\mathbf{x}, \mathbf{y}$	Steganography	Cover sequence and stego sequence of length N.
$\mathbf{m}$	Steganography	Secret message to be embedded.
$\rho_i(x_i, y_i), D(\mathbf{x}, \mathbf{y})$	Steganography	Local distortion and total additive distortion.
$\pi_\lambda(y_i \mid x_i)$	Steganography	Optimal modification probability induced by the Lagrange multiplier λ.
$\mathbf{H}$	Steganography	Linear constraint matrix that enforces the message constraint $\mathbf{yH} = \mathbf{m}$.
$\mathbf{u}, \mathbf{p}, \mathbf{q}$	Polar coding	Source vector, transmitted codeword, and observed sequence in the decoding view.
$\mathbf{G}_N$	Polar coding	Polar generator matrix $\mathbf{B}_N\mathbf{F}_2^{\otimes n}$.
$\mathcal{A}, \mathcal{A}^c$	Polar coding	Information-bit index set and frozen-bit index set.
$\mathbf{u}_{\mathcal{A}^c}$	Polar coding	Frozen-bit sequence shared by the sender and receiver.
$\mathbf{G}'_{\mathcal{A}^c}$	Polar coding	Submatrix used to express the frozen-bit constraint in the decoding formulation.
$L_N^{(i)}$	Polar coding	Log-likelihood ratio (LLR) of the ith polarized bit channel.
d, M	CuSPC design	Number of cover segments for inter-cover parallelism and truncation threshold for Ord nodes.
$\lvert\mathcal{N}\rvert$	CuSPC analysis	Number of scheduled nodes in the SSC traversal after pruning.
e, τ	Evaluation	Embedding efficiency and encoding throughput.

parallel architecture and its GPU implementation. Section 5 reports numerical and steganalysis results. Section 6 discusses runtime and memory scaling. Section 7 concludes the paper.

2 Preliminaries

This section reviews the mathematical model of optimal steganographic coding and the theoretical limit of distortion-minimizing embedding. It then summarizes the duality that connects steganographic embedding to polar decoding. Finally, it reviews the recursive construction of polar codes and the serial bottleneck of SC/SSC execution.

2.1 Theory of Optimal Steganographic Coding

In this paper, we use the term "steganographic coding" specifically to refer to the process of finding the optimal cover modification sequence under a given distortion distribution, as distinct from the encoding and decoding operations of polar codes in the field of communications. Throughout this paper, bold lowercase letters (e.g., $\mathbf{x}$) denote vectors, bold uppercase letters (e.g., $\mathbf{B}_N$) denote matrices, and italic letters (e.g., $\mathcal{X}$) denote sets. Table 1 summarizes the main symbols used throughout the paper.

Specifically, we define $\mathbf{x} = (x_1, x_2, \ldots, x_N) \in \mathcal{X}^N$ and $\mathbf{y} = (y_1, y_2, \ldots, y_N) \in \mathcal{Y}^N$ as the original digital cover sequence and the stego sequence of length N, respectively, where $\mathcal{X}$ is the dynamic range of the cover elements. The secret message to be embedded is denoted by $\mathbf{m}$, and $\mathcal{I}_i \subset \mathcal{X}$ represents the set of permissible modification operations for the corresponding cover element x_i. In the polar channel domain, $\mathbf{p}$ and $\mathbf{q}$ denote the transmitted codeword and the received signal, respectively. Furthermore, the function $\mathcal{B}(x) = x \bmod 2$ is used to extract the least significant bit (LSB) of the input value, while $\mathcal{H}[\cdot]$ is dedicated to performing maximum likelihood hard decisions on the log-likelihood ratio (LLR).

Within the framework of minimizing distortion in modern steganography, the most typical application paradigm is the payload-limited sender (PLS) model [14]. Given a cover sequence $\mathbf{x}$ and its modification set $\mathcal{I}_i$, the mathematical essence of steganographic coding lies in solving for an optimal conditional state transition probability distribution π that minimizes the overall statistical distortion resulting from communication, while satisfying the secret information embedding capacity. This constrained optimization problem can be formally expressed as

$$\arg\min_{\pi} E_{\pi}(D) = \arg\min_{\pi} \sum_{\mathbf{y}\in\mathcal{Y}^N} \pi(\mathbf{y}|\mathbf{x}) D(\mathbf{x},\mathbf{y}). \tag{1}$$

Simultaneously, the entropy constraint on embedding capacity must be satisfied:

$$H(\pi) = -\sum_{\mathbf{y}\in\mathcal{Y}^N} \pi(\mathbf{y}|\mathbf{x}) \log_2 \pi(\mathbf{y}|\mathbf{x}) = |\mathbf{m}|, \tag{2}$$

where $|\mathbf{m}|$ denotes the total bit length of the secret information to be embedded, and $D(\mathbf{x},\mathbf{y})$ is defined as the total embedding distortion at the sequence level. Assuming that the local distortion costs $\rho_i(x_i, y_i)$ incurred when each cover element x_i is modified to y_i are mutually independent and linearly additive, which is

$$D(\mathbf{x},\mathbf{y}) = \sum_{i=1}^{N} \rho_i(x_i, y_i). \tag{3}$$

According to the maximum entropy principle in information theory, the global optimal solution to the above constrained additive distortion optimization problem asymptotically follows the Gibbs distribution [26]. For any cover element x_i, the analytical expression for its optimal modification probability π_λ is given by

$$\pi_\lambda(y_i|x_i) = \frac{\exp[-\lambda\rho_i(x_i, y_i)]}{\sum_{y'\in\mathcal{I}_i} \exp[-\lambda\rho_i(x_i, y')]}, \tag{4}$$

where $\lambda > 0$ is the Lagrangian multiplier controlling the overall information embedding rate [27].

Equation 4 establishes the theoretical limit distribution for optimal steganographic coding from an information-theoretic perspective. The core strength of SPC lies in their ability to closely approximate this physical limit distribution by constructing specific algebraic coding structures.

2.2 Theoretical Framework of Steganographic Polar Codes

The theoretical foundation of SPC stems from the duality between constrained steganographic embedding and channel decoding. Under the additive distortion model, the search for the optimal stego sequence can be written as a constrained maximum-likelihood problem over independent binary modification channels. Polar codes then provide a structured decoding mechanism for solving this problem.

Polar codes are linear block codes based on channel polarization theory. The core idea is to use a recursive transformation to combine N independent binary discrete memoryless channels (B-DMCs) into N polar bit channels [15]. As the code length

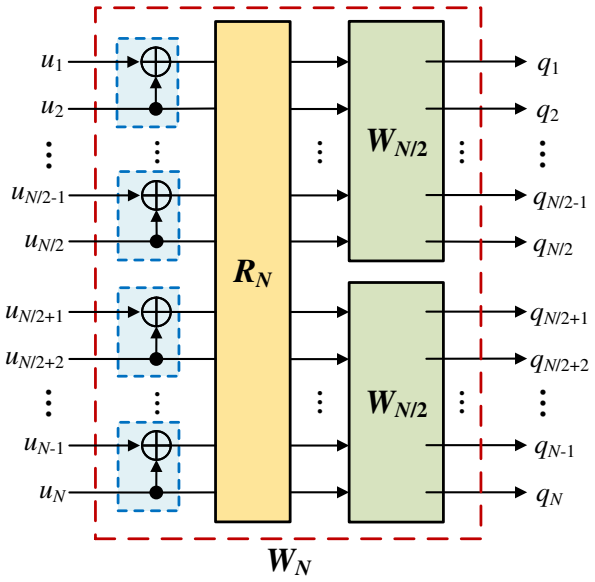


Fig. 1 The recursive structure of polar codes. The basic unit of polar codes W2 is depicted within the blue zone.

N increases, the capacities of these composite channels exhibit a polarized trend where some approach perfect reliability (capacity 1) while others approach pure noise (capacity 0).

2.2.1 Polar Mapping and Steganographic Basics

In polar communication systems, polar codes are typically defined by the parameters $(N, K, \mathcal{A}^c, u_{\mathcal{A}^c})$, where the code length is $N = 2^n$, K is the information bit length, $\mathcal{A}^c$ is the index set of frozen bits, and $\mathcal{A}$ is its complement (the set of information bits). For a source vector u of length N, its basic generation process is realized via modular-2 matrix multiplication:

$$\mathbf{p} = \mathbf{u}\mathbf{B}_N\mathbf{F}_N^{\otimes n} = \mathbf{u}\mathbf{G}_N, \tag{5}$$

where B_N is the bit-reversing permutation matrix, and $F_2^{\otimes n}$ denotes the nth Kronecker power of the core polarization matrix $F_2 = \left[\begin{smallmatrix} 1 & 0 \\ 1 & 1 \end{smallmatrix}\right]$.

In SPC, this process is reinterpreted for embedding, as shown in Fig. 1. The information-bit set $\mathcal{A}$ carries the secret message, while the frozen-bit set $\mathcal{A}^c$ is filled with the shared pseudorandom sequence $u_{\mathcal{A}^c}$. The embedding task is therefore to find a source vector $\mathbf{u}$ whose generated codeword $\mathbf{p}$, interpreted as the stego sequence $\mathbf{y}$, satisfies the message constraint while minimizing distortion with respect to the cover $\mathbf{x}$. This algebraic structure leads naturally to a recursive decoding procedure.

2.2.2 SC Embedding and its Inherent Serial Bottleneck

To find the optimal source vector u described above, the SPC encoder must invoke the successive cancellation (SC) algorithm[28] of the polar code at the embedding stage. During the SC embedding process, the encoder utilizes the recursive structure of the polar code to estimate each source bit $\hat{u}_i$ sequentially, one by one. For the ith bit, its estimate $\hat{u}_i = \mathcal{H}[L_N^{(i)}]$ depends strictly on the current channel?s equivalent log-likelihood ratio (LLR, denoted as $L_N^{(i)}$) as well as the sequence of all previously estimated bits $\hat{u}_1^{i-1}$. The recursive update formula for the LLR is as follows

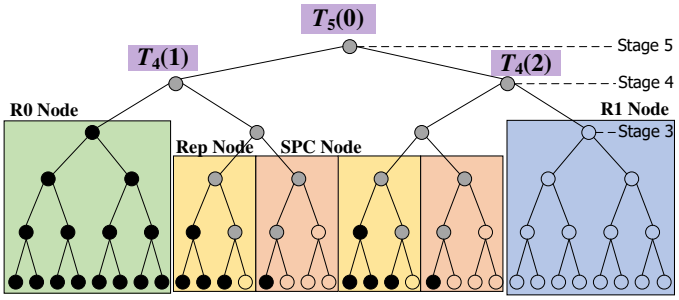
$$L_{2N}^{(2i-1)} = 2\tanh^{-1}\left(\tanh\left(L_N^{(i)}/2\right)\tanh\left(L_N^{(i+N)}/2\right)\right),$$
$$L_{2N}^{(2i)} = (1-2\hat{u}_{2i-1})L_N^{(i)} + L_N^{(i+N)}.$$

Because the estimate of bit i depends on the previous $i-1$ decisions, SC decoding remains strongly serial and its latency grows with the code length. This limits throughput and makes direct parallelization difficult. The SC list (SCL) decoder [29] alleviates this issue by maintaining l candidate paths, expanding them to $2l$ candidates at each information bit, and pruning back to the l most likely paths. The final estimate $\hat{\mathbf{u}}$ is selected from the surviving paths.

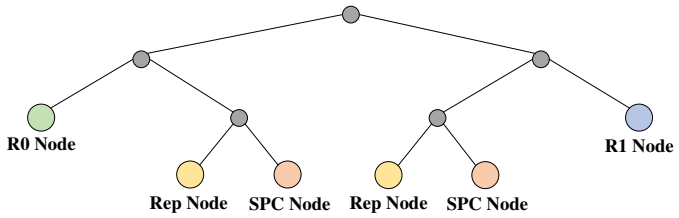
2.3 Tree Representation and Node Pruning of SPC

To decouple the strict serial dependencies of the aforementioned bit-level computations, it is necessary to seek breakthroughs in the underlying mechanisms by exploiting the tree-like topological structure inherent in polar codes. For an SPC of code length $N = 2^n$, its SC embedding process can be modeled as a complete binary tree $T_n(0)$ of depth n. In this tree topology, the root node represents the complete codeword of total length N, while the internal nodes at each level represent the local subcodes obtained through recursive decomposition. Specifically, the parent node $T_t(\phi)$ at stage t $(0 \le t < n)$ can be decomposed into a left child node $T_{t-1}(2\phi)$ and a right child node $T_{t-1}(2\phi+1)$, corresponding respectively to the upper and lower branch operations in the polarization transformation process.

In the standard SC steganographic embedding algorithm, the encoder must perform a depth-first search (DFS), traversing the entire tree layer by layer down to all leaf nodes and estimating the secret information sequentially, as shown in Fig. 2. This leaf-by-leaf traversal approach involves significant computational redundancy, as shown in Figure 2a. The SSC algorithm identifies specific distribution patterns of frozen bits and information bits within the tree structure, directly classifying subtrees satisfying specific conditions as special types of macro-nodes. For these special nodes, the encoder can output the estimated results of the local sequence directly at the parent node level without recursively traversing down to the leaf nodes, thereby achieving large-scale pruning of the polar tree. The basic SSC algorithm defines the following four core node patterns [21, 22]:



(a) Tree representation of SC decoder



(b) SSC decoder with typical identified nodes

Fig. 2 Tree representation of a (32,16) polar code based on SC and SSC decoder, where short codes are divided into 6 child nodes with special patterns whose source bits can be estimated directly. Nodes where all source bits are frozen or unfrozen are denoted in black and white, respectively, while other nodes are depicted in gray. **(a)** Tree representation of SC decoder. **(b)** SSC decoder with typical identified nodes.

1. *Rate-0 Node.* In the subcode corresponding to this node, all source bits are frozen bits. Since the values of the frozen bits are shared in advance (typically 0), the output codeword of this node is entirely determined by the frozen bits. Therefore, the encoder can derive the result directly without calculating any distorted LLR, significantly reducing the computational complexity to $\mathcal{O}(1)$.
2. *Rate-1 Node.* All source bits in this node are information bits. In this case, no polarization freezing occurs in the channel, which is equivalent to uncoded transmission. Consequently, to minimize the embedding distortion for this block, the encoder simply applies maximum likelihood (ML) hard decisions to the input LLR sequence, yielding the locally modified sequence $\hat{p}$:

$$\hat{p}_i = \mathcal{H}(L_{N'}^{(i)}), \tag{6}$$

where $\mathcal{H}(\cdot)$ is the hard decision function. This process can be executed entirely in parallel.
3. *Repetition Node.* For a Rep node of length N', only the last bit is an information bit carrying the secret message, and the rest are frozen bits. Consequently, there are only two valid candidate local codewords, p_0 and p_1:

$$p_0 = (u_1, u_2, \cdots, u_{N'-1}, 0)G_{N'}$$

$$p_1 = (u_1, u_2, \cdots, u_{N'-1}, 1)G_{N'}$$

The SPC encoder calculates the total distortion introduced by these two modification schemes and selects the candidate codeword with the larger metric value η^{Rep} (i.e., the one with the smaller distortion) as the final judgment $\hat{p} = (\hat{p}_1, \hat{p}_2, \ldots, \hat{p}_{N'})$:

$$\eta^{\mathrm{Rep}} = \sum_{i=1}^{N'} (1 - 2\hat{p}_i) L_{N/N'}^{(i)}. \tag{7}$$

4. *Single Parity Check Node (SPCN).* In this node, only the first source bit is a frozen bit u', while the remaining bits are information bits. The sequence $p = (p_1, p_2, \ldots)$ generated by this node must satisfy the parity check condition:

$$\langle \mathbf{p} \rangle = \bigoplus_i p_i = u', \tag{8}$$

where $\bigoplus$ denotes modulo-2 addition. During steganographic embedding, the encoder first performs fast parallel hard decision-making, similar to the Rate-1 node. If the preliminary decision does not satisfy the parity condition, the encoder locates the LLR with the smallest absolute value within this node and forces a flip of the corresponding estimated bit. This effectively minimizes local embedding distortion while satisfying the parity constraint.

By recursively decomposing the cover sequence into the four aforementioned special nodes, the SSC algorithm significantly reduces computational complexity. More importantly, it transforms the original bit-level serial operations into highly independent, node-level block operations. This local structural characteristic provides the algorithmic mapping foundation for the parallel steganographic coding architecture proposed in this paper, enabling the realization of high-throughput parallel acceleration on GPU platforms.

While SSC reduces the amount of computation through subtree pruning, its default depth-first traversal still imposes temporal dependencies across nodes. As a result, lower algorithmic complexity does not automatically yield high throughput. This limitation is especially pronounced for large covers, where serial waiting between nodes restricts hardware utilization. CuSPC addresses this gap by exposing parallelism at the intra-node, inter-node, and inter-cover levels and mapping these forms of parallelism to the GPU SIMT execution model.

3 Parallel Coding Architecture

This section presents the CuSPC architecture. It first formalizes the connection between additive-distortion steganographic coding and constrained polar decoding. It then introduces the three-level scheduling strategy of CuSPC, covering intra-node, inter-node, and inter-cover parallelism.

3.1 Problem Formulation

To solve the binary PLS problem, we first provide the following definitions for convenience:

$$\pi_i(y_i = x_i|x_i) = \pi_i^{(0)}, \quad \pi_i(y_i = \bar{x}_i|x_i) = \pi_i^{(1)},$$
$$\rho_i(y_i = x_i, x_i) = \rho_i^{(0)}, \quad \rho_i(y_i = \bar{x}_i, x_i) = \rho_i^{(1)}.$$

Based on Equation (4), the distortion in binary additive steganographic coding can be expressed as:

$$\rho_i^{(0)} - \rho_i^{(1)} = \frac{1}{\lambda} \ln \left[\frac{1}{\pi_i^{(0)}} - 1 \right], \tag{9}$$

$$\rho_i^{(1)} - \rho_i^{(0)} = \frac{1}{\lambda} \ln \left[\frac{1}{\pi_i^{(1)}} - 1 \right]. \tag{10}$$

When $\mathbf{x} \in \mathcal{X}$ is given with fixed probability $1/|\mathcal{X}|$, the total distortion in PLS problem can be further expressed by substituting Equation (9) and (10) that

$$\begin{aligned} D(\mathbf{x}, \mathbf{y}) &= \sum_{\substack{i=1 \\ y_i = x_i}}^{N} \rho_i^{(0)} + \sum_{\substack{i=1 \\ y_i \neq x_i}}^{N} \rho_i^{(1)} \\ &= \sum_{i=1}^{N} \rho_i^{(1)} + \frac{1}{\lambda} \sum_{\substack{i=1 \\ y_i = x_i}}^{N} \left[\ln(\pi_i^{(1)} - \pi_i^{(0)}) \right] \\ &= \sum_{i=1}^{N} \rho_i^{(0)} + \frac{1}{\lambda} \sum_{\substack{i=1 \\ y_i \neq x_i}}^{N} \left[\ln(\pi_i^{(0)} - \pi_i^{(1)}) \right], \end{aligned}$$

thereafter we can deduce that

$$\begin{aligned} D(\mathbf{x}, \mathbf{y}) = \frac{1}{2} \sum_{i=1}^{N} \left[\rho_i^{(0)} + \rho_i^{(1)} \right] + \\ \frac{1}{2\lambda} \left[\sum_{\substack{i=1 \\ y_i = x_i}}^{N} \ln \left[\frac{\pi_i^{(1)}}{\pi_i^{(0)}} \right] + \sum_{\substack{i=1 \\ y_i \neq x_i}}^{N} \ln \left[\frac{\pi_i^{(0)}}{\pi_i^{(1)}} \right] \right], \end{aligned}$$

and the optimal stego sequence can be calculated according to Equation (8) and (9) that

$$\hat{\mathbf{y}}_\lambda = \arg \min_{\substack{\hat{\mathbf{y}} \in \mathcal{Y} \\ \hat{\mathbf{y}}\mathbf{H} = \mathbf{m}}} D(\mathbf{x}, \hat{\mathbf{y}})$$

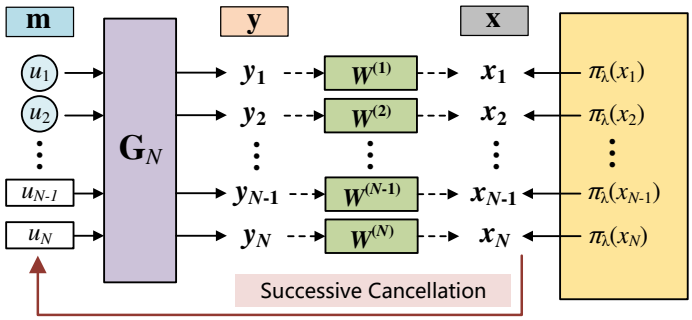


Fig. 3 The basic structure of the steganographic coding model of SPC.Data flows are depicted in the figure with arrows.

$$
\begin{aligned}
&= \arg \min_{\substack{\hat{\mathbf{y}}\in\mathcal{Y} \\ \hat{\mathbf{y}}\mathbf{H}=\mathbf{m}}} \left[\sum_{\substack{i=1 \\ \hat{y}_i=x_i}}^{N} \ln \left[\frac{\pi_i^{(1)}}{\pi_i^{(0)}} \right] + \sum_{\substack{i=1 \\ \hat{y}_i\neq x_i}}^{N} \ln \left[\frac{\pi_i^{(0)}}{\pi_i^{(1)}} \right] \right] \\
&= \arg \max_{\substack{\hat{\mathbf{y}}\in\mathcal{Y} \\ \hat{\mathbf{y}}\mathbf{H}=\mathbf{m}}} \sum_{i=1}^{N} \left(1 - 2\left|\hat{y}_i - x_i\right|\right) \ln \left[\frac{\pi_i^{(0)}}{\pi_i^{(1)}} \right],
\end{aligned}
$$

where matrix $\mathbf{H}$ represents a shared linear transformation for receivers to decoding $\mathbf{m}$ [27]. Since the modification of each cover element is independent, and $\hat{y}_i$ equals either x_i or $\bar{x}_i$, the equation above can be maximized by

$$\hat{\mathbf{y}}_\lambda = \arg\max_{\hat{\mathbf{y}}\in\mathcal{Y}} \sum_{i=1}^{N} \ln \pi_i(y_i|x_i) \tag{11}$$

$$\text{subject to } \hat{\mathbf{y}}_\lambda \mathbf{H} = \mathbf{m}. \tag{12}$$

For a polar code $(N, K, \mathcal{A}, \mathbf{u}_{\mathcal{A}^c})$, assume that the sender transmits codeword $\mathbf{p}$ generated from source word $\mathbf{u}$ and the receiver observes $\mathbf{q}$. The ML estimate of $\mathbf{p}$ can then be written as

$$\hat{\mathbf{p}}_\lambda = \arg\max_{\hat{\mathbf{p}}\in\mathcal{P}} \sum_{i=1}^{N} \ln W(q_i|\hat{p}_i) \tag{13}$$

$$\text{subject to } \hat{\mathbf{p}}_\lambda \mathbf{G}'_{\mathcal{A}^c} = \mathbf{u}_{\mathcal{A}^c}, \tag{14}$$

where $\mathbf{G}'_{\mathcal{A}^c}$ is the submatrix of inverse matrix of $\mathbf{G}_N$ formed by the columns with indices in $\mathcal{A}^c$.

The optimization in (11) has the same constrained-ML structure as (13): both maximize a sum of per-symbol log-likelihood terms under a linear constraint. Following Li *et al.*[19], each binary modification position can be interpreted as an independent binary-input channel whose transition probabilities are determined by $\pi_\lambda(y_i|x_i)$. In

the symmetric binary case, this channel reduces to a BSC with crossover probability induced by the optimal embedding distribution.

Proposition 1 *Given a cover sequence* $\mathbf{x}$*, additive distortion profile* $\boldsymbol{\rho}$*, and payload constraint* $|\mathbf{m}|$*, the binary PLS embedding problem can be mapped to a constrained ML decoding problem in which the cover plays the role of the observation, the stego sequence is the decoded codeword, and the message/frozen-bit constraint is enforced through the polar code structure.*

Proof sketch. Equation (11) shows that optimal embedding selects $\hat{\mathbf{y}}$ by maximizing the log-probability of independent local transitions under the linear constraint $\hat{\mathbf{y}}\mathbf{H} = \mathbf{m}$. Equation (13) has the same optimization form, with $W(q_i|\hat{p}_i)$ replacing $\pi_i(y_i|x_i)$. By identifying $\mathbf{q}$ with the observed cover $\mathbf{x}$ and $\hat{\mathbf{p}}$ with the stego sequence $\hat{\mathbf{y}}$, the embedding problem becomes a constrained decoding problem over independent binary channels. Polar decoding, and in particular SC/SSC decoding, therefore provides a structured way to realize the embedding rule.

This decoding interpretation leads to the SPC framework shown in Fig. 3:

1. Compute the optimal modification distribution π_λ from $\mathbf{m}$ and $\boldsymbol{\rho}$, and associate each position with an independent binary modification channel;
2. Perform channel polarization on these non-uniform binary channels and estimate their reliabilities;
3. Assign the message and frozen-bit constraints according to the selected information set;
4. Recover the stego sequence $\mathbf{y}$ through SC/SSC decoding using the corresponding initial LLRs.

This equivalence links steganographic coding to polar decoding at the execution level. It does not imply that practical SC/SSC implementations are identical to exact ML decoding for finite lengths, but it does justify using mature polar-decoding machinery as the computational backbone of steganographic embedding.

3.2 Parallelization Strategies for Steganographic Polar Codes

Based on the preceding analysis, the computational efficiency of SPC encoding is primarily constrained by the encoding and decoding algorithms of polar codes. Although the SSC algorithm successfully transforms inefficient bit-by-bit estimation into efficient node-by-node estimation by pre-identifying specific nodes, existing serial implementations face two major challenges when directly porting the SSC algorithm to the SIMT parallel architecture of a GPU.

1. Underutilization of intrinsic parallelism potential. At specific levels of the polarization transformation, computations between nodes in different subtrees (inter-node) and bit-level operations within a specific node (intra-node) are completely decoupled in terms of data dependencies. Traditional DFS scheduling fails to exploit this topological property, resulting in an inability to fully unleash the parallel computing performance of modern hardware under the SIMT programming model.

2. Scheduling overhead of fine-grained nodes. Relying solely on the four basic SSC node types for pruning leaves a large number of short nodes at the lower levels of the polar tree. Under the SIMT programming model, frequently waking up the kernel to process these fine-grained nodes triggers severe control divergence and thread scheduling overhead. The resulting additional latency undermines the performance gains of parallel computation.

To address these challenges, this section details the two-level fine-grained parallelization strategy of the CuSPC architecture, encompassing both intra-node and inter-node levels. Given that list decoding offers only marginal security gains in steganographic scenarios, which are grossly disproportionate to its exponentially expanding computational complexity [19], this paper adopts an SSC-based non-list decoding structure, thereby maximizing encoding throughput while ensuring near-optimal security.

3.2.1 Intra-Node Parallelism with Ordinary Nodes

To mitigate the scheduling overhead caused by fine-grained nodes, this paper introduces a GPU-optimized node type called the ordinary node (Ord). During the recognition and pruning stages of the polar tree, for subtrees of length M that cannot be classified as a standard SSC special node (Rate-0, Rate-1, Rep, SPCN), CuSPC no longer enforces recursive probing but instead truncates them directly and marks them as Ord nodes. We then employ a highly parallel exhaustive search strategy for block-level processing. For an Ord node of length M, let its set of candidate source sequences be $\mathcal{V}$. We utilize the high-concurrency characteristics of GPU warps to concurrently compute the steganographic metric η^{Ord} for all 2^M candidate sequences $\mathbf{v}'$ and directly determine the optimal local steganographic sequence $\tilde{\mathbf{v}}$.

$$\tilde{\mathbf{v}} = \arg\max_{\mathbf{v}' \in \mathcal{V}} \eta^{Ord}(\mathbf{v}') \tag{15}$$

$$\text{where } \eta^{Ord}(\mathbf{v}') = \sum_{i=1}^{M} (1 - 2v'_i) L^{(i)}_{N/M} \tag{16}$$

In the specific CUDA implementation, we set the truncation threshold M to 8 (i.e., constructing Ord-8 nodes) to achieve an optimal balance between the size of the search space and the number of nodes. Furthermore, for steganographic coding scenarios, this paper redefines and deeply adapts the internal parallel logic of the standard SSC node within the steganographic context.

1. *Fully frozen node (corresponding to the Rate-0 node in standard polar codes).* All source bits within this node are frozen bits, meaning they are entirely filled with a pseudorandom sequence shared by both sender and receiver and do not carry the secret message $\mathbf{m}$. Since its output codeword is a priori determined, this paper adopts an offline pre-calculation strategy, generating this segment directly before the start of the steganographic coding and thereby reducing the computational complexity to strictly $\mathcal{O}(1)$.
2. *Dual-Candidate Node (DC/Rep Node).* Only the last bit is an information bit carrying $\mathbf{m}$. At this point, there are only two valid candidate sequences, p_0 and p_1.

CuSPC allocates parallel threads to simultaneously evaluate the metrics of both candidates and performs a branch-free best-fit decision.

$$\eta(\mathrm{DC}) = \sum_{i=1}^{N'} (1 - 2\hat{p}_i) L_{N/N'}^{(i)}. \tag{17}$$

3. *Full-payload node (corresponding to the Rate-1 node in standard polar codes).* All source bits within this node are information bits, all used to carry the secret message $\mathbf{m}$. This is equivalent to transmission without coding constraints. To minimize local embedding distortion, CuSPC schedules multiple threads to perform fully parallel maximum likelihood hard decisions on the input LLR sequence.

$$\hat{p}_i = \mathcal{H}(L_{N/N'}^{(i)}). \tag{18}$$

4. *Single Parity Check Node (SPCN).* Only the first bit is subject to the frozen bit constraint, while the remaining bits carry the secret information. CuSPC first performs fully parallel hard decision-making in Rate-1 mode. Subsequently, it executes the parity-check equations in parallel. If the parity check fails, it utilizes the GPU?s efficient reduction algorithm to rapidly locate the index with the smallest absolute value of the LLR and performs parallel bit flipping.

3.2.2 Inter-Node Parallelism

In addition to fine-grained operations within nodes, the topological structure of the polar tree offers substantial inter-node parallelism. For sibling nodes at the same stage t, LLR and partial-sum updates depend only on their local parent branches, so different branches can be processed independently. As shown in Fig. 4, CuSPC maps decoupled node-level tasks at the same stage to different CUDA thread blocks or thread bundles for concurrent execution. This breaks part of the serial waiting imposed by DFS traversal and increases overall throughput.

3.3 Inter-Cover Parallelism and Segmentation Scheduling Mechanism

Concurrency strategies based on cover segmentation (such as the CPU-STC scheme by Su et al. [20]) have been proven to effectively reduce the encoder?s state space. Inspired by this, this paper introduces a macro-level inter-cover parallelism mechanism for SPC. However, if such coarse-grained segmentation tasks, suitable for CPUs, are directly mapped to GPUs, the concurrent execution of multiple independent SPC trees will generate a large volume of intermediate state data. This leads to severe uncoalesced memory access, significantly degrading the memory bandwidth utilization of the GPU's Streaming Multiprocessors (SMs). Consequently, the core breakthrough of this paper?s inter-cover parallel architecture lies in reconstructing a thread-block-adaptive independent scheduling and interleaved memory access architecture at the CUDA level after proportionally partitioning the original N-length stego cover into d

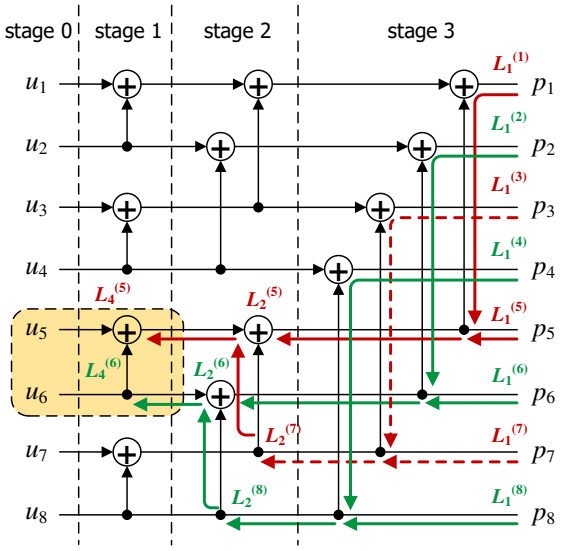


Fig. 4 The parallel update of LLRs and partial sums for polar code decoding at different stages.

independent sub-covers. This architecture arranges intermediate variables that belong to different vector segments but share isomorphic positions within the polar tree into an interleaved and contiguous layout within physical video memory. This underlying scheduling seamlessly aligns SPC?s segment encoding logic with the SIMT execution model, maximizing the concurrent computing power of the GPU when processing large-capacity vectors while ensuring coalesced global memory access.

The inter-cover parallelism mechanism enhances system-level performance in two main aspects:

1. Maximizing SMs utilization. By decoupling d independent steganography tasks and mapping them to independent CUDA thread blocks, coarse-grained task parallelism is achieved. This decoupling design eliminates the synchronization overhead between fragments, enabling the GPU scheduler to efficiently distribute massive computational tasks across individual SMs. This significantly improves hardware utilization and effectively masks global memory access latency through high-concurrency threading.
2. Optimization of Resource Granularity and Computational Density. The reduction in sub-cover length ($N \rightarrow N/d$) proportionally reduces the scale of intermediate variables required per task. This refinement of resource granularity greatly alleviates the pressure on registers and on-chip caches per thread, while also allowing each SM to physically host more concurrent thread bundles, thereby achieving higher computational throughput under limited on-chip resource constraints.

However, architectural design often involves trade-offs. As the number of segments d increases, although hardware concurrency grows linearly, the reduction in subcode length N/d causes the channel polarization effect to decline non-linearly, leading to a slight increase in embedding distortion (i.e., a compromise in security). Therefore,

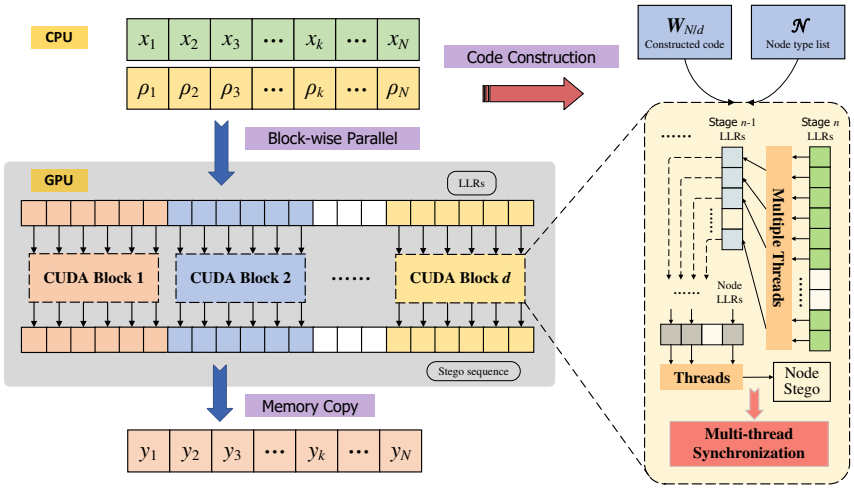


Fig. 5 The heterogeneous cooperative architecture of CuSPC for steganographic polar coding.

in practical streaming covert communication systems, the segmentation parameter d must be selected dynamically based on latency tolerance and security thresholds to maximize the coding rate.

4 Design of a GPU-Based Parallel SPC Encoder

Building on Section 3, this section presents CuSPC, a high-throughput steganographic polar encoder based on the Compute Unified Device Architecture (CUDA) framework. It explains how the proposed parallel scheduling strategy is translated into an efficient implementation, focusing on heterogeneous co-processing, memory access optimization, and compact data structures.

4.1 Architecture Overview

The architecture of the proposed CuSPC is illustrated in Fig. 5, in which all transactions in steganographic coding are scheduled by the CPU, and the GPU acts as a co-processor to accelerate calculations with its parallelization performance.

The steganographic coding flow of CuSPC is divided into a construction stage and an estimation stage. In the construction stage, the optimal modification probabilities $\pi_i(y_i|x_i)$ are computed and used as input to the encoder. Because the embedding entropy varies monotonically with λ in Equation 4, the optimal λ can be obtained by binary search, with entropy evaluation accelerated on the GPU. For code construction, the resulting steganographic channels are approximated as BECs following [19], and their reliabilities are ranked through Bhattacharyya parameters. This BEC model is used only for construction and reliability ordering; the embedding interpretation in Section 3 remains based on the binary modification probabilities $\pi_i(y_i|x_i)$. The constructed code length must match the segmented cover length.

In the next phase, the d cover segments are encoded concurrently, utilizing inter-cover parallelism to achieve significant acceleration. Therefore thread synchronizations are not necessary across each segment, and d segments are encoded in different CUDA blocks. Meanwhile, inter-node and intra-node parallelisms are performed in each CUDA block with multi-thread parallelization and synchronization. The cover and hidden message are encoded serially node-by-node and the intermediate LLRs and partial sums are accessed by each thread in an interleaving manner, which are all stored in the global memory of the GPU. After two stages of computing, the segmented stego array is copied from the global memory to the host memory and concatenated to the output stego.

4.2 Global Memory Scheduling and Interleaved Memory Access Optimization

During the execution of polarized steganographic coding, the iterative updates of the LLR and partial sums constitute the system?s primary memory access overhead. Based on the tree representation of channel polarization, only one LLR array is needed for each stage of polarization in recursion. Therefore, for a PSC with the length of $N = 2^n$, $N-1$ elements are enough for storing the intermediate LLR required for updating the child nodes, which can be arranged in a `float` array with single dimension as:

$$\underbrace{L_1^{(1)}}_{\text{stage 0}}, \underbrace{L_2^{(1)} L_2^{(2)}}_{\text{stage 1}}, \underbrace{L_4^{(1)} L_4^{(2)} L_4^{(3)} L_4^{(4)}}_{\text{stage 2}}, \cdots, \underbrace{L_N^{(1)} L_N^{(2)} \cdots L_N^{(N)}}_{\text{stage } n}.$$

And partial sums can be stored in an array that contains $2(N-1)$ elements in `uint8` data format as 2 vectors of codeword bits output at single polar encoding.

Consequently, the efficiency of low-level memory scheduling for such high-frequency accesses to state variables directly determines the throughput ceiling of the GPU parallel architecture. Given that the cover sequence lengths in multimedia steganography scenarios typically reach the megabit scale, the vast amount of intermediate data generated during execution far exceeds the physical capacity limits of the GPU?s on-chip shared memory. Constrained by this hardware limitation, the CuSPC architecture relies on global memory as the core data exchange medium, trading high access latency for larger capacity.

To effectively mitigate the high access latency of global memory, the CUDA programming model requires that memory reads adhere to the principle of coalesced access. That is, the 32 threads within the same warp should, whenever possible, concurrently address physically contiguous memory segments, thereby merging dispersed memory requests into a single bus transaction. However, the butterfly operation at the core of polarized coding exhibits an inherent strided access characteristic. At the nth level of polarization, the physical distance between operands increases exponentially (2^n). This non-contiguous addressing pattern disrupts coalesced memory access, reducing effective GPU memory bandwidth utilization. To address the aforementioned memory access bottleneck, CuSPC proposes an interleaved memory access strategy tailored to CUDA thread blocks. By restructuring the physical storage topology of the data, this strategy maps the logical strided accesses in the deep layers of the polar

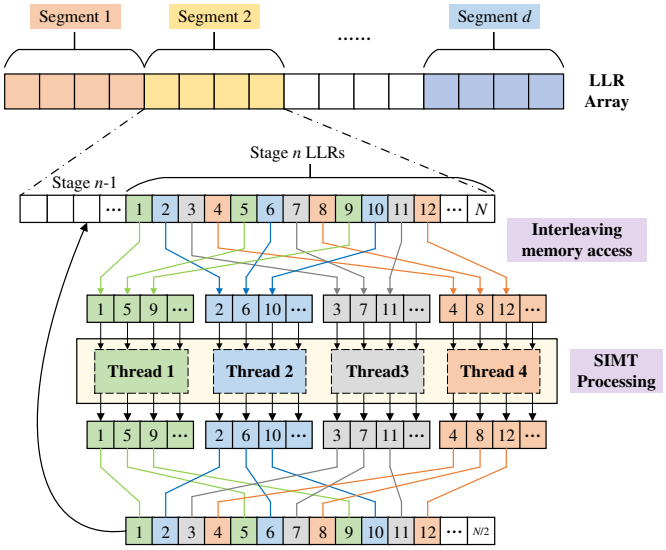


Fig. 6 Illustration of coalesced memory access with interleaving for inter-cover parallelism (4 threads per block).

tree to continuous reads at the physical memory level. The underlying mapping logic is as follows:

1. Physical data reorganization. Building upon the cover segmentation strategy described in Section 3.3, CuSPC stores the data from d independent stego covers in an interleaved manner within physical memory. The data array is no longer arranged linearly according to the order of a single polar tree. Instead, the ith element from the d trees that occupy the same topological position is stored consecutively, followed immediately by the $(i+1)$th element from all trees.
2. Warp mapping reconstruction. When constructing the computational grid, a specific mapping relationship is established between warps and data indices. The kth warp is assigned to concurrently process a set of local nodes with the same logical index k across the d independent polar trees.
3. Implementation of coalesced memory access. Under this layout, when performing polarization transformations or LLR updates, adjacent threads t and $t+1$ within the same warp will concurrently access the same logical index position in the tth and $(t+1)$th segments, respectively. Thanks to interleaved storage, these two logically parallel requests are directly mapped to contiguous physical memory addresses.

This optimization mechanism effectively overcomes the discrete addressing limitations inherent in deep-level polar tree computations. It transforms these access patterns into coalesced memory accesses that align with the GPU?s SIMT architecture, thereby minimizing unaligned transactions and significantly boosting overall memory bandwidth utilization. Fig. 6 gives the example of interleaving memory access with

4 threads per block. For each step of LLR and partial sum recursion on each block, coalesced memory accesses can be achieved with interleaving.

4.3 Storing Node Types and Frozen Bits

In the proposed architecture, information on node types and frozen bits is pre-calculated in the code construction stage on the host side, but it is heavily required by kernel functions on the GPU during execution. To maximize the available bandwidth for critical intermediate variables like LLRs, the control flow information must be strictly lightweight. Rather than using conventional arrays of structures, CuSPC employs hardware-aware, bit-level compressed data structures to represent the topology of SSC nodes and frozen bit sequences.

4.3.1 Bit-field Compression of Node Topology.

The i-th node $\mathcal{N}_i$, identified by the method described in Section 3.2.1, can be logically represented by a three-element tuple:

$$\mathcal{N}_i : \ (\mathcal{T}_i, s_i, k_i) \tag{19}$$

Here, $\mathcal{T}_i$ is an integer denoting the type of the node, which enumerates a finite set of states including Rate-0, Rate-1, Rep, SPCN, and the proprietary Ord node. The variable s_i indicates the starting offset of the node within the current stage to ensure precise data pointer alignment. The variable k_i represents the length of the current data block, which is determined by the node's absolute depth level within the polar tree.

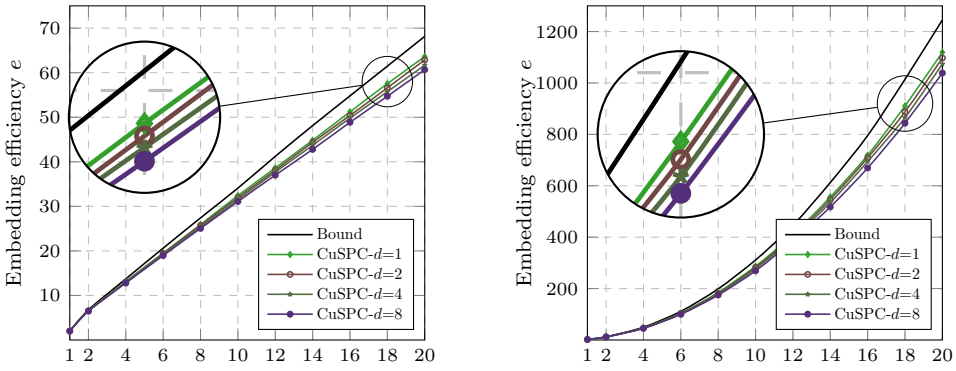
For unified data storage, a straightforward implementation might utilize a 64-bit integer (`uint64`) format to avoid the potential overflow of s_i and k_i. However, to avoid the memory alignment overhead and bus waste associated with such bloated structures, CuSPC employs bit-field technology. This approach encapsulates the three-dimensional core attributes of a single node $(\mathcal{T}_i, s_i, k_i)$ into a single 32-bit unsigned integer (`uint32_t`). All compressed information regarding node topologies is then concatenated as a compact vector and allocated in the global memory of the GPU. During high-frequency kernel iterations, GPU threads do not need to initiate multiple memory reads. Instead, they resolve the aforementioned configuration parameters within a single clock cycle using efficient hardware bit manipulation instructions such as shifts and masks. This design not only reduces the frequency of global memory accesses but also effectively alleviates the resource pressure on on-chip registers.

4.3.2 Bit-packing of Frozen Bits.

Frozen bits and their corresponding indices are the most important variables governing decisions during steganographic coding. Because the messages processed in each CUDA block differ, a standard implementation might generate a state vector of length N/d in an `int8` data format for each segmented cover. In this naive array, a value of -1 typically represents a non-frozen channel, while 0 or 1 denotes the pre-shared frozen bit values.

Table 2 Hardware Resources and Parameters of the Devices Used for Evaluations.

Device	CPUs			GPUs				
	CPU	Frequency (GHz)	Memory (GB)	GPU	Base Clock (Mbps)	Memory (GB)	SMs	CUDA Cores
Device1	Intel i7-6700K	4.00	16	NVIDIA GTX 980M	1038	8	12	1536
Device2	Intel Xeon Gold 5118	2.30	48	NVIDIA Tesla V100	1245	32	80	5120



(a) Linear distortion profile

(b) Square distortion profile

Fig. 7 Embedding efficiency of different steganographic codes with fixed cover length $N = 2^{20}$ under various distortion profiles.

The frozen sequence $\mathbf{u}_{\mathcal{A}^c}$ serves as a shared key between the sender and receiver and directly determines the final form of special nodes such as Rate-0 and Rep. Storing these indicators and one-dimensional Boolean values using standard byte (`int8`) arrays results in significant space wastage and undermines continuous memory access. To address this storage redundancy, CuSPC introduces a bit-packing mechanism. Prior to data transmission, the host end abandons the `int8` array and packages consecutive frozen bit indicators into 32-bit unsigned integers. To ensure continuous access to memory, these packed vectors are concatenated in the order of the cover segmentation and copied to global memory. When the GPU performs hard-decision verification in parallel, threads extract specific key bits in situ through simple masking operations. Compared to traditional array storage, this bit-level compression strategy significantly enhances the system's overall arithmetic intensity, ensuring that the encoder maintains extremely high computational efficiency even under stringent bandwidth constraints.

5 Experimental Results and Comprehensive Performance Evaluation

This section presents the experimental validation and system-level performance evaluation of the proposed GPU-based steganographic polar coding architecture (CuSPC). The evaluation focuses on embedding efficiency, encoding throughput, and anti-steganalysis performance. Through comparison with mainstream baselines, including SPSC and STC, we assess whether CuSPC improves throughput while preserving competitive security.

In this section, numerical simulations are conducted on randomly generated covers $\mathbf{x} \in \{0, 1, \ldots, 255\}^N$ and binary messages $\mathbf{m}$ to evaluate the performance of the implemented CuSPC on the GPU. Steganographic security is evaluated by the embedding efficiency e averaged over 10 runs, expressed as follows:

$$e = \frac{|\mathbf{m}|}{D(\mathbf{x}, \mathbf{y})} \tag{20}$$

Furthermore, the steganographic coding throughput τ is evaluated by measuring the average encoding time t_{enc} over 10 runs.

$$\tau = \frac{|\mathbf{x}|}{t_{\text{enc}}} \tag{21}$$

Note that the encoding latency is not evaluated, as there is no practical requirement for concurrent calls in this steganographic coding context.

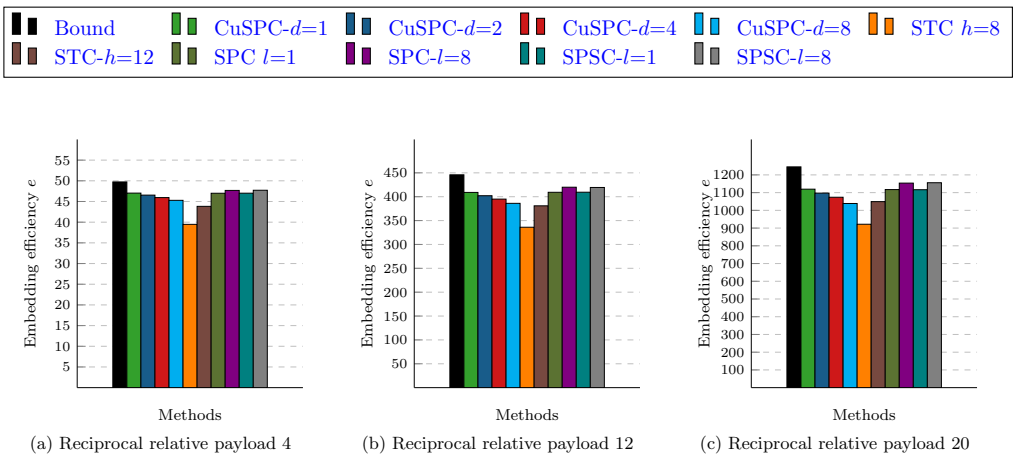
For comparison, the performance of STC [14] with a sub-matrix height of $h \in \{8, 12\}$, SPC [19], and SPSC [18] with a list size of $l \in \{1, 8\}$ is also evaluated. The distortion profile, denoted as $\boldsymbol{\varrho} = (\varrho_1, \varrho_2, \ldots, \varrho_N)$, is defined by a non-decreasing function $\varrho(x)$ where $\varrho_i = \varrho(i/N)$ for all $i \in \{1, 2, \ldots, N\}$ [14]. In this section, we use $\varrho(x) = x$ and $\varrho(x) = x^2$ for linear and square profiles, respectively, to simulate multi-level distortions in the real world.

Unless otherwise specified, all evaluations in this section are conducted on **Device1**, as detailed in Tab. 2. Two NVIDIA devices, a GTX 980M for laptops and a Tesla V100 for servers, are selected. All polar code-based steganographic codes are implemented in C/C++ with the same structure, where CPU-based calculations of modification probabilities and LLRs are accelerated by the SIMD programming model. Meanwhile, the proposed CuSPC is compiled with CUDA Toolkit 12.3. The open-source executable version of STC is fetched from the website of the DDE laboratory at Binghamton University[1]. Note that only the encoding procedure of STC is accelerated by streaming SIMD extension (SSE) instructions.

5.1 Evaluation on Embedding Efficiency

The embedding efficiencies of the proposed CuSPC are simulated under linear and square distortion profiles and compared with state-of-the-art steganographic codes

[1] https://dde.binghamton.edu/download/syndrome/



(a) Reciprocal relative payload 4 (b) Reciprocal relative payload 12 (c) Reciprocal relative payload 20

Fig. 8 Comparison results on the embedding efficiencies of the state-of-the-art steganographic codes. Evaluations are conducted under the square distortion profile with relative payloads of 1/4, 1/12, and 1/20.

across reciprocal relative payload ranges from 1 to 20. Binary covers with a fixed length of $N = 2^{20}$, as well as hidden messages, are randomly generated for the evaluation.

5.1.1 Performance Under Different Parameters

The security performance of the proposed CuSPC is first investigated under different cover segmentation parameters. As discussed previously, with cover segmentation, the embedding efficiency of steganographic coding decreases due to the reduction in channel polarization levels. However, the security performance level of the proposed CuSPC remains the same as that of SPC or SPSC since the SSC decoding structure is adopted. Fig. 7 confirms this analysis. Under both linear and square distortion profiles, the security performance of CuSPC slightly drops with the increase in segments but still approaches the rate-distortion bound.

5.1.2 Comparison with Other Steganographic Codes

Fig. 8 illustrates the embedding efficiencies under the square distortion profile for covers with a length of 2^{20}. Because the list coding strategy is not enabled, at different relative payloads, the proposed CuSPC without inter-cover parallelism achieves near-optimal performance identical to that of SPC and SPSC with a list size of 1. When d increases to 8, the security performance gradually decreases but maintains the same level as STC with a sub-matrix height of 12.

5.2 Evaluation on Embedding Throughput

In this subsection, we further investigate the steganographic coding throughput under different cover lengths and relative payloads. For comparison, STC with sub-matrix

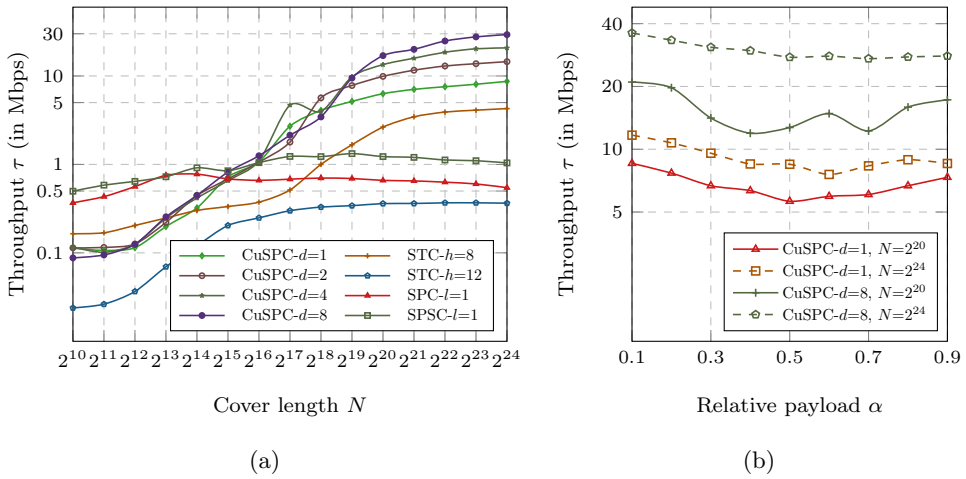


Fig. 9 Throughput performance on (a) various cover length N with a relative payload of 0.4 and (b) various relative payload α. Evaluations are conducted under the square distortion profile.

heights of 8 and 12 is introduced, while SPC and SPSC with a list size of 1 are used, as their security performance is already sufficiently high.

5.2.1 Throughput Performance Under Various Cover Lengths

Fig. 10 depicts the throughput variation with cover length for representative steganographic codes. Logarithmic coordinates are used to show the performance gap clearly. In general, the throughput of most schemes increases with cover length because the fixed cost of code construction is amortized as N grows. The exception is SPC, whose throughput changes more smoothly with cover length. For STC with $h = 8$, the throughput increases once the cover length reaches approximately 2^{16}, and more than 4 Mb/s is achieved for longer covers on **Device1**. However, as h increases, the encoding cost grows exponentially, so the measured throughput increases only from about 100 Kb/s to 360 Kb/s over the tested cover lengths. Because SSC-based codes operate node-wise, their throughput varies more noticeably with cover length. The CPU-based SPSC with list size 1 reaches about 1 Mb/s from a cover length of 2^{14}. In contrast, the proposed CuSPC benefits more from parallel execution as N increases. With segmentation parameter $d = 8$, it reaches 30 Mb/s on **Device1**. Even without inter-cover parallelism, CuSPC still improves throughput by about 8× over SPSC at the same embedding-efficiency level.

5.2.2 The Impact of Relative Payload on Throughput

Based on the partial order of channel polarization [30, 31], as the number of hidden message bits increases, the number of *Rate-0* and *DC* nodes in our architecture reduces. The total number of nodes grows and theoretically reaches a peak when the ratio of information bits is 0.5. Therefore, as the relative payload increases from 0 to 1, the throughput of steganographic codes with node-wise encoding generally decreases

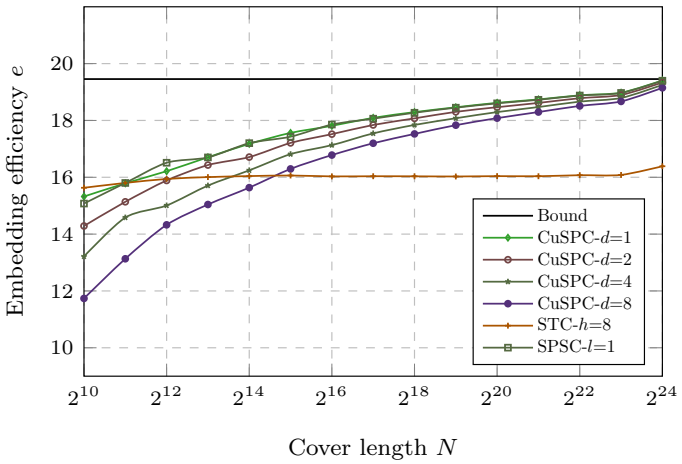


Fig. 10 Embedding efficiency on various cover length N from 2^{10} to 2^{24}. Evaluations are conducted under the square distortion profile with a relative payload of 0.4.

first and then increases, though it is also affected by code construction. Fig. 9 illustrates how the encoding throughput in logarithmic coordinates varies with the relative payload. The encoding speed of the proposed steganographic coding architecture is generally lowest at a relative payload of 0.5. Compared to a relative payload of 0.1, the throughput performance drops by up to 30%.

5.3 Effect of Cover Segmentation

The cover segmentation strategy significantly improves steganographic coding throughput, especially for larger covers, but it negatively affects the embedding efficiency. In this subsection, we focus on the impact of the cover segmentation strategy on the proposed encoding architecture.

5.3.1 Impact on Embedding Efficiency

Fig. 10 illustrates how cover segmentation affects the security performance of CuSPC. Through cover segmentation, the length of the constructed code in CuSPC for steganographic coding becomes shorter.

This results in a decrease in embedding efficiency as the performance of channel polarization drops. With the increase in cover length, the rate of convergence in channel polarization approaches the optimum, and the negative impact on security with segmented covers gradually becomes negligible. When the cover length is 2^{24}, the steganographic coding performance almost reaches the upper bound. Steganographic coding using the proposed CuSPC with $d = 8$ causes only a 2% performance degradation in embedding efficiency.

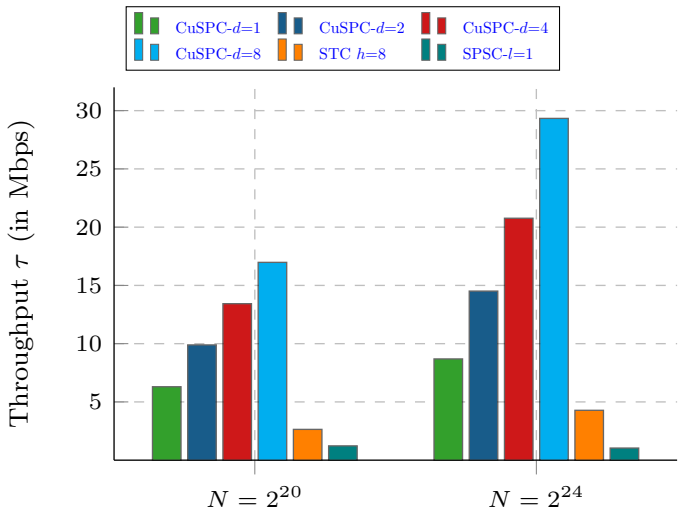


Fig. 11 Throughput performance of CuSPC on covers with lengths of 2^{20} and 2^{24} to investigate the improvements brought by the cover segmentation strategy.

5.3.2 Impact on Throughput Performance

As shown in Fig. 10, when the cover length is less than 2^{20}, the introduction of inter-cover parallelism does not contribute to the improvement of encoding throughput because code construction and memory copy operations are time-consuming. Therefore, we reevaluate the throughput performance of CuSPC on covers with lengths of 2^{20} and 2^{24} to further investigate the improvements brought by the cover segmentation strategy, as illustrated in Fig. 11. For large covers, the throughput of CuSPC increases by a fixed amount for a given cover length as the parameter d increases exponentially, which is determined by the computing architecture of the GPU. Segments of different covers are assigned to different kernels for steganographic coding, where the calculations are not strictly parallelized but performed by SMs. Despite this, the throughput performance improves by up to 30 times when the cover length is 2^{24} (4096×4096) compared to SPSC, which relies solely on CPUs.

5.4 Throughput on Different Devices

Tab. 4 presents the performance of the proposed steganographic coding architecture evaluated across different devices. Steganographic coding is successfully performed on different GPU devices, as the embedding efficiencies remain unchanged. However, for **Device2**, which has a lower-frequency CPU and a higher-performance GPU, the encoding throughput is poor with short covers.

Still, performance improves as the cover length increases, because the parallel calculation of node-wise estimation gradually accounts for an increasing proportion of the overall steganographic coding computation.

Table 3 The Detection Error Rate (in %) of Stego Images Generated with Relative Payloads (0.1 to 0.4 bpnzAC) by STC, SPC, SPSC, and CuSPC with J-UNIWARD (JUNI) and UERD. Samples with Different Quality Factors (QF) are Also Evaluated.

Method	QF=75				QF=90			
	0.1	0.2	0.3	0.4	0.1	0.2	0.3	0.4
JUNI-Simulator	0.4181	0.2972	0.1752	0.0826	0.4632	0.3901	0.2855	0.1737
JUNI-CuSPC-d=1	**0.4057**	**0.2771**	**0.1580**	**0.0708**	**0.4590**	**0.3801**	**0.2715**	**0.1607**
JUNI-CuSPC-d=2	0.4035	0.2759	0.1548	0.0687	0.4576	0.3767	0.2686	0.1567
JUNI-CuSPC-d=4	0.4024	0.2705	0.1505	0.0651	0.4577	0.3749	0.2652	0.1521
JUNI-CuSPC-d=8	**0.3992**	**0.2655**	**0.1470**	**0.0619**	**0.4567**	**0.3725**	**0.2605**	**0.1473**
JUNI-STC-h=8	**0.4022**	**0.2667**	**0.1421**	**0.0588**	**0.4577**	**0.3700**	**0.2542**	**0.2542**
JUNI-STC-h=12	**0.4103**	**0.2792**	**0.1592**	**0.0701**	**0.4602**	**0.3796**	**0.2698**	**0.1569**
JUNI-SPC-l=1	0.4053	0.2783	0.1588	0.0713	0.4581	0.3786	0.2736	0.1599
JUNI-SPC-l=8	0.4102	0.2851	0.1673	0.0752	0.4603	0.3839	0.2755	0.1670
JUNI-SPSC-l=1	0.4067	0.2784	0.1588	0.0707	0.4596	0.3793	0.2726	0.1599
JUNI-SPSC-l=8	0.4113	0.2856	0.1675	0.0754	0.4621	0.3851	0.2748	0.1678
UERD-Simulator	0.4109	0.2895	0.1749	0.0869	0.4553	0.3755	0.2767	0.1688
UERD-CuSPC-d=1	**0.3987**	**0.2707**	**0.1569**	**0.0770**	**0.4513**	**0.3656**	**0.2615**	**0.1558**
UERD-CuSPC-d=2	0.3984	0.2675	0.1541	0.0753	0.4495	0.3624	0.2586	0.1530
UERD-CuSPC-d=4	0.3932	0.2644	0.1502	0.0722	0.4487	0.3609	0.2548	0.1499
UERD-CuSPC-d=8	**0.3908**	**0.2597**	**0.1458**	**0.0694**	**0.4483**	**0.3573**	**0.2510**	**0.1471**
UERD-STC-h=8	**0.3947**	**0.2589**	**0.1444**	**0.0648**	**0.4473**	**0.3554**	**0.2440**	**0.1362**
UERD-STC-h=12	**0.4016**	**0.2723**	**0.1574**	**0.0761**	**0.4519**	**0.3641**	**0.2586**	**0.1532**
UERD-SPC-l=1	0.3974	0.2727	0.1571	0.0769	0.4509	0.3639	0.2621	0.1552
UERD-SPC-l=8	0.4025	0.2768	0.1653	0.0817	0.4523	0.3700	0.2685	0.1624
UERD-SPSC-l=1	0.3995	0.2715	0.1580	0.0762	0.4511	0.3658	0.2621	0.1577
UERD-SPSC-l=8	0.4039	0.2760	0.1656	0.0817	0.4522	0.3719	0.2680	0.1628

5.5 Anti-Steganalysis Performance

Having verified the computational performance, this section further evaluates the security of the proposed CuSPC architecture in real-world scenarios. The security evaluation against image steganalysis is conducted using the DCTR feature extractor [32] and an ensemble classifier [33]. Steganography resistance is rigorously measured by the average detection error rate P_E, which is mathematically defined as follows:

$$P_E = \frac{1}{2}(P_{FA} + P_{MD}), \tag{22}$$

Here, P_{FA} represents the false alarm rate and P_{MD} represents the missed detection rate. In covert communication adversarial scenarios, a higher P_E value (capped at 0.5) indicates that the classifier finds it more difficult to distinguish the stego image from the original cover. Consequently, this metric effectively evaluates the anti-detection capability of the encoding algorithm.

To conduct the experiments, a total of 10,000 JPEG covers of size 512×512 with quality factors of 75 and 90 are randomly selected and converted from the ALASKA#2 dataset in RAW format [34]. Stego images are generated using the J-UNIWARD [5] and UERD [6] distortion functions. The numerical simulations are performed on **Device2** (detailed in Tab. 2) using MATLAB R2023a. To ensure execution efficiency, the core steganographic coding algorithms are implemented in C/C++ and compiled into MEX files for direct invocation within the MATLAB environment.

As shown in Tab. 3, the anti-steganalysis performance of CuSPC decreases as the segmentation parameter d increases, but it remains higher than that of STC with sub-matrix height $h = 8$ in our experiments. Because list decoding is not implemented in

Table 4 Throughput Performance of the Proposed CuSPC on Different Devices, where Simulations are Conducted Under the Square Distortion Profile with Parameter d Fixed to 8.

Device	Cover length	Payload (bpp)	T/P τ (Mbps)	Embedding efficiency e
Device1	2^{16}	0.2	4.8516	66.7389
		0.4	4.2413	16.7808
	2^{20}	0.2	16.8649	69.9127
		0.4	15.0617	18.0915
	2^{24}	0.2	32.6545	75.3021
		0.4	29.6877	19.1486
Device2	2^{16}	0.2	0.6641	68.3735
		0.4	0.6758	17.8214
	2^{20}	0.2	8.1607	69.9389
		0.4	8.7017	18.0873
	2^{24}	0.2	39.1239	75.2896
		0.4	33.8062	19.1489

CuSPC, the detection error rate is slightly lower than that of SPC or SPSC when the list size is 1.

Because a higher detection error rate implies better security, the proposed architecture maintains steganalysis performance comparable to STC while delivering higher throughput. Specifically, without inter-cover parallelism, CuSPC (marked in red in Tab. 3) reaches a security level close to that of STC with $h = 12$. Under 8-segment acceleration, CuSPC still remains close to STC with $h = 8$ (denoted by blue values in the table). It is worth noting that embedding efficiency cannot fully represent practical resistance to steganalysis [35]. For example, although CuSPC with $d = 8$ has nearly the same embedding efficiency as STC with $h = 12$, their measured detection error rates still differ slightly.

5.6 Ablation and Design Analysis

As the final part of this chapter, we summarize the component-wise evidence behind the observed end-to-end speedup. Table 5 reports the main ablation results, and Fig. 12 shows the measured effect of enabling Ord-8 on the number of scheduled nodes.

The ablation results support three main conclusions. First, the Ord node directly reduces scheduled work: at payload 0.4 and $N = 2^{24}$, enabling Ord-8 lowers the node count by about 24.9%. Second, inter-cover parallelism is the main source of coarse-grained acceleration: at $N = 2^{24}$, the throughput rises from 8.69 Mb/s for $d = 1$ to 29.33 Mb/s for $d = 8$. Third, the corresponding embedding-efficiency drop remains small, only about 1.34% under the same setting. Interleaved memory layout and bit-packing are therefore retained in the ablation table as implementation-level enablers: they do not change the embedding rule, but they are required for the measured parallel speedup and the linear-space implementation of CuSPC.

Table 5 Ablation summary of the main CuSPC design choices. Numerical entries are taken from the existing measurements reported in Figs. 11 and 12.

Component	Key evidence	Main effect
Ord node	Ord-8 reduces scheduled nodes from 18881 to 14189 at $N = 2^{24}$ and payload 0.4 (24.9%).	Lowers traversal and scheduling overhead on short residual subtrees.
Inter-cover parallelism	Throughput increases from 8.69 to 29.33 Mb/s from $d = 1$ to $d = 8$ at $N = 2^{24}$ (3.38×).	Supplies the main coarse-grained throughput gain on large covers.
Segmentation trade-off	Embedding efficiency changes from 19.41 to 19.15 from $d = 1$ to $d = 8$ at $N = 2^{24}$ (1.34% loss).	Shows that the throughput gain is obtained at a small security-related cost.
Interleaved memory	Reflected in the final speedup of CuSPC at large cover lengths.	Improves global-memory coalescing so exposed parallelism translates into measured throughput.
Bit-packing	Reflected in the linear-space implementation of CuSPC.	Reduces control-data traffic and supports runtime and memory scaling.

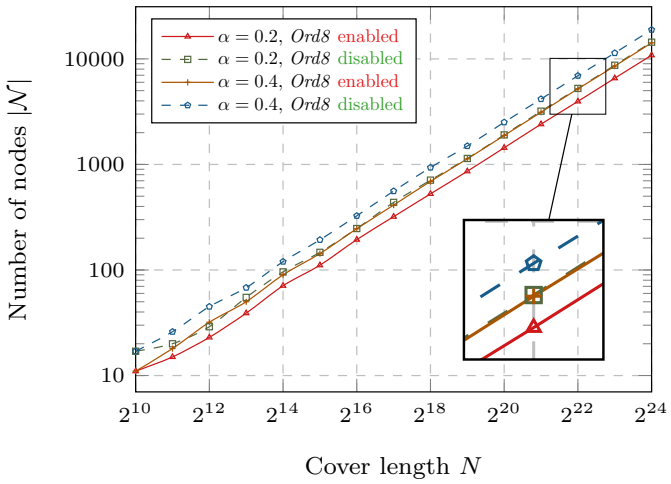


Fig. 12 The relationship between the number of nodes $|\mathcal{N}|$ and the cover length N with and without *Ord8* nodes enabled.

Table 6 Comparison of Complexity and Hardware Affinity for Steganographic Coding Algorithms

Algorithm	Core Mechanism	Time Complexity	Space Complexity	Hardware Affinity
STC [14]	Viterbi path search	$\mathcal{O}(2^h N)$	$\mathcal{O}(2^h N)$	Low
SPSC [18]	SC / SCL tree traversal	$\mathcal{O}(lN \log N)$	$\mathcal{O}(lN \log N)$	Medium
CuSPC	SSC pruning + GPU parallel mapping	$\mathcal{O}(lN^{0.75} \log N)$	$\mathcal{O}(N)$	High

Note: Here, h represents the sub-matrix height of STC, l is the list size of polar codes, and N denotes the total length of the cover sequence. The space complexity of CuSPC is compressed within the linear boundary of $\mathcal{O}(N)$ due to the introduction of bit-packing technology.

6 Complexity Analysis

The preceding sections demonstrate the throughput advantage of CuSPC. This section discusses the proposed architecture from the perspectives of runtime scaling and memory usage.

For a given cover of length N, traditional STC require the Viterbi algorithm to perform a global path search. When processing a single cover element, there are 2^h state transitions in its trellis diagram, where h is the height of the constraint submatrix. Consequently, the time and space complexities of STC are strictly bounded by $\mathcal{O}(2^h N)$. Although increasing h can improve embedding efficiency, the resulting computational complexity grows exponentially, which severely limits the practical application of STC in scenarios involving large covers or high security requirements.

In contrast, by introducing polar codes and the SCL decoding structure, and neglecting the constant-scale overhead of channel construction, the time and space complexities of the steganographic coding are both reduced to $\mathcal{O}(lN \log N)$, where l is the list size. Compared to STC, which typically requires setting $h \geq 8$ to ensure security against steganalysis, this method not only has lower steganographic coding complexity but also exhibits a much more gradual increase in computational time even when the list size l is moderately increased to enhance security.

Building on this, CuSPC further reduces the amount of scheduled computation by combining SSC pruning with GPU-oriented execution mapping. Its main change is to replace bit-wise estimation with node-wise estimation, which removes many redundant branches in the polarization tree without increasing memory usage asymptotically.

Specifically, the statistical fit in Fig. 12 shows that when Ord-8 optimization is enabled and the relative payload is 0.4 bpp, the measured number of scheduled nodes $|\mathcal{N}|$ follows an approximately logarithmic-linear trend with respect to the cover length N:

$$\log_{10} |\mathcal{N}| \approx 0.22218 \cdot \log_2 N - 1.1804 \tag{23}$$

Through base conversion and algebraic simplification, this fit can be rewritten as

$$|\mathcal{N}| \approx 0.066 \cdot N^{0.73806} < 0.07 \cdot N^{0.75} \tag{24}$$

This fitted node-count law should be interpreted as an empirical scaling observation rather than a formal asymptotic complexity proof. Under the tested payload and Ord-8 configuration, the measured runtime grows sublinearly relative to the standard $lN \log N$ reference and is well approximated by a term proportional to $lN^{0.75} \log N$. Table 6 therefore serves as a qualitative comparison between classical theoretical complexity results and the observed runtime behavior of CuSPC.

7 Conclusion

In practical covert communication systems designed for high-capacity and real-time interaction, the throughput of steganographic coding is as important as its resistance to steganalysis. To address the serial bottleneck of conventional polar steganographic codes, this paper proposes CuSPC, a GPU-oriented steganographic coding architecture built on SSC-based scheduling, ordinary-node processing, and interleaved inter-cover

execution. Experiments show that CuSPC improves throughput by about 30× over the CPU serial baseline while keeping the embedding-efficiency loss below 2% and maintaining competitive steganalysis performance.

As mobile edge computing and heterogeneous architectures continue to evolve, the application of covert streaming communication is rapidly extending to portable devices. In future work, we plan to adapt our hierarchical parallel architecture for resource-constrained mobile platforms, with a primary focus on optimizing energy efficiency. These efforts will facilitate the practical deployment of high-throughput covert systems in edge environments.

References

[1] Fridrich J. Steganography in digital media: principles, algorithms, and applications. Cambridge University Press; 2009.

[2] Pevnỳ T, Fridrich J. Benchmarking for steganography. In: international workshop on information hiding. Springer; 2008. p. 251–267.

[3] Guo L, Ni J, Shi YQ.: An efficient JPEG steganographic scheme using uniform embedding.

[4] Tang W, Tan S, Li B, Huang J. Automatic steganographic distortion learning using a generative adversarial network. IEEE Signal Processing Letters. 2017;24(10):1547–1551.

[5] Holub V, Fridrich J, Denemark T. Universal distortion function for steganography in an arbitrary domain. EURASIP Journal on Information Security. 2014;2014(1):1.

[6] Guo L, Ni J, Su W, Tang C, Shi YQ. Using statistical image model for JPEG steganography: uniform embedding revisited. IEEE Transactions on Information Forensics and Security. 2015;10(12):2669–2680.

[7] Li B, Wang M, Huang J, Li X.: A new cost function for spatial image steganography.

[8] Yang J, Ruan D, Huang J, Kang X, Shi YQ. An embedding cost learning framework using GAN. IEEE Transactions on Information Forensics and Security. 2019;15:839–851.

[9] Crandall R. Some notes on steganography. Posted on steganography mailing list. 1998;1998:1–6.

[10] Westfeld A.: High Capacity Despite Better Steganalysis (F5—A steganographic algorithm).

[11] Fridrich J, Goljan M, Lisonek P, Soukal D. Writing on wet paper. IEEE Transactions on signal processing. 2005;53(10):3923–3935.

[12] Fridrich J, Goljan M, Soukal D.: Efficient wet paper codes.

[13] Filler T, Judas J, Fridrich J.: Minimizing embedding impact in steganography using trellis-coded quantization.

[14] Filler T, Judas J, Fridrich J. Minimizing additive distortion in steganography using syndrome-trellis codes. IEEE Transactions on Information Forensics and Security. 2011;6(3):920–935.

[15] Arikan E. Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels. IEEE Transactions on information Theory. 2009;55(7):3051–3073.

[16] Diouf B, Diop I, Fall PA, Dolo B, Diop AK, Diouf M, et al.: JPEG Steganography Based on Successive Cancellation Decoding of Polar Codes.

[17] Diouf B, Diop I, Keita KW, Diouf M, Farsi SM, Tall K, et al.: Polar coding steganographic embedding using successive cancellation. Springer.

[18] Fu H, Zhao X, He X.: High-Performance Steganographic Coding Based on Sub-Polarized Channel.

[19] Li W, Zhang W, Li L, Zhou H, Yu N. Designing near-optimal steganographic codes in practice based on polar codes. IEEE Transactions on Communications. 2020;68(7):3948–3962.

[20] Su A, Ma S, Zhao X. Fast and secure steganography based on J-UNIWARD. IEEE Signal Processing Letters. 2020;27:221–225.

[21] Alamdar-Yazdi A, Kschischang FR. A simplified successive-cancellation decoder for polar codes. IEEE communications letters. 2011;15(12):1378–1380.

[22] Sarkis G, Giard P, Vardy A, Thibeault C, Gross WJ. Fast polar decoders: Algorithm and implementation. IEEE Journal on Selected Areas in Communications. 2014;32(5):946–957.

[23] Cammerer S, Leible B, Stahl M, Hoydis J, ten Brink S. Combining belief propagation and successive cancellation list decoding of polar codes on a GPU platform. In: 2017 IEEE international conference on acoustics, speech and signal processing (ICASSP). IEEE; 2017. p. 3664–3668.

[24] Han X, Liu R, Liu Z, Zhao L. Successive-cancellation list decoder of polar codes based on GPU. In: 2017 3rd IEEE International Conference on Computer and Communications (ICCC). IEEE; 2017. p. 2065–2070.

[25] Liu Z, Liu R, Zhang H. High-throughput adaptive list decoding architecture for polar codes on gpu. IEEE Transactions on Signal Processing. 2022;70:878–889.

[26] Filler T, Fridrich J. Gibbs construction in steganography. IEEE Transactions on Information Forensics and Security. 2010;5(4):705–720.

[27] Fridrich J, Filler T.: Practical methods for minimizing embedding impact in steganography.

[28] Leroux C, Tal I, Vardy A, Gross WJ.: Hardware architectures for successive cancellation decoding of polar codes.

[29] Tal I, Vardy A. List decoding of polar codes. IEEE Transactions on Information Theory. 2015;61(5):2213–2226.

[30] Schürch C. A partial order for the synthesized channels of a polar code. In: 2016 IEEE International Symposium on Information Theory (ISIT). IEEE; 2016. p. 220–224.

[31] He G, Belfiore JC, Land I, Yang G, Liu X, Chen Y, et al. Beta-expansion: A theoretical framework for fast and recursive construction of polar codes. In: GLOBECOM 2017-2017 IEEE Global Communications Conference. IEEE; 2017. p. 1–6.

[32] Holub V, Fridrich J. Low-complexity features for JPEG steganalysis using undecimated DCT. IEEE Transactions on Information forensics and security. 2014;10(2):219–228.

[33] Kodovsky J, Fridrich J, Holub V. Ensemble classifiers for steganalysis of digital media. IEEE Transactions on information forensics and security. 2011;7(2):432–444.

[34] Cogranne R, Giboulot Q, Bas P.: ALASKA# 2: Challenging academic research on steganalysis with realistic images.

[35] Ker AD, Bas P, Böhme R, Cogranne R, Craver S, Filler T, et al. Moving steganography and steganalysis from the laboratory into the real world. In: Proceedings of the first ACM workshop on Information hiding and multimedia security; 2013. p. 45–58.

Declarations

- **Ethics Approval:** Not applicable. This research did not involve human participants, animal subjects, or any data requiring ethical approval.
- **Funding:** This work was supported by the National Natural Science Foundation of China (NSFC) under Grant No. 62272456.

- **Conflict of Interest:** The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.
- **Consent to Participate:** Not applicable. This study did not involve human subjects.
- **Consent for Publication:** Not applicable. This manuscript does not contain any individual person's data in any form.
- **Data Availability:** The datasets generated and analyzed during the current study are available from the corresponding author on reasonable request.
- **Author Contributions:** X.H. and H.F. conceived and designed the research. X.H. and L.W. implemented the CuSPC architecture and performed the numerical simulations. H.F. and Y.C. developed the theoretical formulation of steganographic polar coding. X.H. and L.W. conducted the steganalysis experiments. M.Z. contributed to the GPU optimization and memory scheduling design. X.H. wrote the main manuscript text. H.F. and Y.C. reviewed and revised the manuscript. All authors approved the final version of the manuscript.
- **Acknowledgements:** Not applicable.