Window Processing of Binary Polarization Kernels

Grigorii Trofimiuk, *Student Member, IEEE,* Peter Trifonov, *Member, IEEE*

***Abstract*—A decoding algorithm for polar (sub)codes with binary $2^t \times 2^t$ polarization kernels is presented. It is based on the window processing (WP) method, which exploits the linear relationship of the polarization kernels and the Arikan matrix. This relationship enables one to compute the kernel input symbols probabilities by computing the probabilities of several paths in Arikan successive cancellation (SC) decoder.**

In this paper we propose an improved version of WP, which has significantly lower arithmetic complexity and operates in log-likelihood ratios (LLRs) domain. The algorithm identifies and reuses common subexpressions arising in computation of Arikan SC path scores.

The proposed algorithm is applied to kernels of size 16 and 32 with improved polarization properties. It enables polar (sub)codes with the considered kernels to simultaneously provide better performance and lower decoding complexity compared with polar (sub)codes with Arikan kernel.

***Index Terms*—Polar codes, polarization kernels, fast decoding.**

I. Introduction

Polar codes are a novel class of error-correcting codes, which achieves the symmetric capacity of a binary-input discrete memoryless channel W. They have low complexity construction, encoding, and decoding algorithms [1]. However, the performance of polar codes of practical length is quite poor. The reasons for this are their low minimum distance and the existence of imperfectly polarized bit subchannels, which are used for transmission of data. This causes the SC algorithm to be highly suboptimal. These problems can be alleviated by employing the successive cancellation list (SCL) algorithm [2], as well as improved code constructions, such as polar subcodes and polar codes with CRC [3], [4], [5].

Polarization is a general phenomenon and is not restricted to Arikan matrix [6]. One can replace it by a larger matrix, called *polarization kernel*, which has better polarization properties obtaining thus better finite length performance. Polar codes with large kernels were shown to provide an asymptotically optimal scaling exponent [7]. Various polarization kernels together with simplified processing algorithms were recently proposed, including l-formula method for medium length (up to 16) kernels [8], BCH kernels with reduced trellis processing complexity [9], approximate processing based on Box and Match algorithm [10], and convolutional polar kernels [11]. However, the complexity of the existing processing algorithms for the kernels with improved polarization properties (compared with Arikan kernel) is still too high for practical use.

Window processing algorithm was introduced in [12] and applied to Reed-Solomon kernels. Independently, WP algorithm was suggested for processing of permuted Arikan matrix [13]. WP exploits the relationship of the considered kernels and the Arikan matrix, which enables one to compute the kernel input symbols probabilities by computing the probabilities of several paths in Arikan SC decoding. In this paper we propose an improved version of WP, which has significantly lower arithmetic complexity and operates in LLRs domain. The proposed algorithm identifies and reuses common subexpressions arising in recursive computation of Arikan SC path scores. Further complexity reduction is achieved by exploiting some properties of maximization of these scores.

The proposed approach allows very simple processing for some specially constructed kernels. We discuss the application of the proposed algorithm to polar codes with 16×16 and 32×32 kernels with improved polarization properties. Namely, we consider two 16×16 kernels both having rate of polarization 0.51828, but different scaling exponents: 3.346 and 3.45. We present also a 32×32 kernel with rate of polarization 0.521936 and scaling exponent 3.41706. For comparison, Arikan matrix has rate of polarization 0.5 (the optimal value is 1) and scaling exponent 3.627 (the optimal value is 2).

The proposed algorithm can be used with SCL decoding. Simulation results show that polar codes with the considered large kernels provide significant performance gain compared with polar codes with Arikan kernels under decoding with the same list size. Furthermore, the proposed approach results in lower decoding complexity compared with polar codes with Arikan kernel with the same performance.

The paper is organized as follows. The background on polar codes and polarization kernels is presented in Section II. The window processing (WP) algorithm is introduced in Section III. Section IV presents a common subexpression identification and reusing method, which provides significant complexity reduction of WP. Some further improvements of WP are discussed in Section V. The processing algorithms for the considered kernels are described in Section VI. Simulation results are presented in Section VII.

II. Background

In this section we give a brief introduction to polarization kernels, including polarization properties, and polar codes. We also define the problem of kernel processing and describe the processing of Arikan matrix in the approximate LLR domain.

A. Notations

For a positive integer n, we denote by $[n]$ the set of n integers $\{0, 1, \ldots, n-1\}$. The vector u_a^b is a subvector

G. Trofimiuk, and P. Trifonov are with ITMO University, Saint-Petersburg, Russia. E-mail: {gtrofimiuk,pvtrifonov}@itmo.ru

This work was supported by Government of Russian Federation (grant 08-08).

This work was partially presented at the Information Theory Workshop'2018 and International Symposium on Information Theory'2019.

The source code is available at https://github.com/gtrofimiuk/SCLKernelDecoder.

$(u_a, u_{a+1}, \dots, u_b)$ of some vector u. For vectors a and b we denote their concatenation by $a.b$. $M[i]$ is an i-th row of the matrix M. By $M[i,j]$ we denote j-th element of $M[i]$.

B. Polarizing transformation

Polarization kernel K is an $l \times l$ invertible matrix, which is not upper triangular under any column permutation [6]. Consider a binary-input memoryless channel with transition probabilities $W(y|c), c \in \mathbb{F}_2, y \in \mathcal{Y}$, where $\mathcal{Y}$ is the output alphabet. An $(n = l^m, k)$ polar code is a linear block code generated by k rows of matrix $G_m = M^{(m)}K^{\otimes m}$, where $\otimes m$ is m-fold Kronecker product of matrix with itself, and $M^{(m)}$ is a digit-reversal permutation matrix, corresponding to mapping $\sum_{i=0}^{m-1} j_i l^i \to \sum_{i=0}^{m-1} j_{m-1-i} l^i$, $j_i \in [l]$. The encoding scheme is given by $c_0^{n-1} = u_0^{n-1} G_m$, where $u_i, i \in \mathcal{F}$, are set to some pre-defined values, e.g. zero (frozen symbols), $|\mathcal{F}| = n - k$, and the remaining values u_i are set to the payload data.

It is possible to show that a binary input memoryless channel W together with matrix G_m gives rise to bit subchannels $W_{m,K}^{(i)}(y_0^{n-1}, u_0^{i-1}|u_i)$ with capacities approaching 0 or 1. The fraction of noiseless subchannels approaches $I(W)$ [6]. Selecting $\mathcal{F}$ as the set of indices of low-capacity subchannels enables almost error-free communication.

It is convenient to define probabilities

$$W_{m,K}^{(i)}(u_0^i|y_0^{n-1}) = \frac{W_{m,K}^{(i)}(y_0^{n-1}, u_0^{i-1}|u_i)}{2W(y_0^{n-1})} =$$

$$\sum_{u_{i+1}^{n-1}} W_{m,K}^{(n-1)}(u_0^{n-1}|y_0^{n-1}) = \sum_{u_{i+1}^{n-1}} \prod_{i=0}^{n-1} W((u_0^{n-1}G_m)_i|y_i).$$

We further define $\mathbf{W}_m^{(i)}(u_0^j|y_0^{n-1}) = W_{m,K}^{(i)}(u_0^i|y_0^{n-1})$, where K should be clear from the context.

Due to the recursive structure of G_m, one has

$$\mathbf{W}_m^{(sl+\phi)}(u_0^{sl+\phi}|y_0^{n-1}) = \sum_{u_{sl+\phi+1}^{l(s+1)-1}} \prod_{j=0}^{l-1} \mathbf{W}_{m-1}^{(s)}(\theta_K[u_0^{l(s+1)-1}, j]|y_{j\frac{n}{l}}^{(j+1)\frac{n}{l}-1}), \quad (1)$$

where $\theta_K[u_0^{(s+1)l-1}, j]_r = (u_{lr}^{l(r+1)-1}K)_j, r \in [s+1], \phi \in [l]$.

At the receiver side, the successive cancellation (SC) decoding algorithm makes estimates

$$\widehat{u}_i = \begin{cases} \arg\max_{u_i \in \mathbb{F}_2} \mathbf{W}_m^{(i)}(\widehat{u}_0^{i-1}.u_i|y_0^{n-1}), & i \notin \mathcal{F}, \\ \text{the frozen value of } u_i & i \in \mathcal{F}. \end{cases}$$

C. Kernel processing

Decoding algorithms for polar codes require one to compute the probabilities $\mathbf{W}_m^{(i)}(u_0^i|y_0^{n-1})$ for a given polarization transform G_m. Since these probabilities are computed recursively (1), we assume for simplicity that $m = 1$. The corresponding task is referred to as *kernel processing*[1]. The probabilities for one layer of the polarization transform are given by

$$\mathbf{W}_1^{(\phi)}(u_0^\phi|y_0^{l-1}) = \sum_{u_{\phi+1}^{l-1} \in \mathbb{F}_2^{l-\phi-1}} \mathbf{W}_1^{(l-1)}(u_0^{l-1}|y_0^{l-1}). \quad (2)$$

[1]Sometimes it is also referred to as kernel marginalization [14].

The value of ϕ is referred to as processing *phase*, while vector u_0^ϕ is referred to as a *path*. Computing these probabilities reduces to soft-output decoding of nonsystematically encoded codes generated by the last $l-\phi-1$ rows of K. This problem was considered in [15].

We introduce approximate probabilities

$$\widetilde{\mathbf{W}}_1^{(\phi)}(u_0^\phi|\mathbf{y}_0^{l-1}) = \max_{u_{\phi+1}^{l-1} \in \mathbb{F}_2^{l-\phi-1}} \mathbf{W}_1^{(l-1)}(u_0^{l-1}|y_0^{l-1}) \quad (3)$$

They were shown to provide a substantial reduction of the complexity of polar codes sequential decoding [10], [16].

Decoding can be implemented using the LLRs of the approximate probabilities (3)

$$\mathbf{S}_{1,\phi} = \ln \frac{\widetilde{\mathbf{W}}_1^{(\phi)}(u_0^{\phi-1}.0|y_0^{l-1})}{\widetilde{\mathbf{W}}_1^{(\phi)}(u_0^{\phi-1}.1|y_0^{l-1})} = R(0) - R(1), \quad (4)$$

where $R(a) = \max_{u_{\phi+1}^{l-1}} \ln \mathbf{W}_1^{(l-1)}(u_0^{\phi-1}.a.u_{\phi+1}^{l-1}|y_0^{l-1})$.

The above expression means that $\mathbf{S}_{1,\phi}$ can be computed by performing the maximum likelihood (ML) decoding of the coset of a code generated by last $l-\phi-1$ rows of the kernel K, assuming that all $u_j, \phi < j < l$, are equiprobable. The particular coset representative is given by the product of $u_0^{\phi-1}$ and matrix consisting of top ϕ rows of kernel K.

D. Processing of Arikan matrix

Straightforward evaluation of (4) for an arbitrary kernel has complexity $O(l2^l)$. However, there is a simple recursive procedure for computing (4) for Arikan matrix $F_t = \begin{pmatrix} 1 & 0 \\ 1 & 1 \end{pmatrix}^{\otimes t}$.

Let $l = 2^t$. Consider encoding scheme

$$c_0^{l-1} = v_0^{l-1} F_t.$$

Similarly to (3), define approximate probabilities

$$\widetilde{W}_t^{(i)}(v_0^i|y_0^{l-1}) = \max_{v_{i+1}^{l-1} \in \mathbb{F}_2^{l-i-1}} W_t^{(l-1)}(v_0^{l-1}|y_0^{l-1})$$

and modified log-likelihood ratios

$$S_t^{(i)}(v_0^{i-1}, y_0^{l-1}) = \log \frac{\widetilde{W}_t^{(i)}(v_0^{i-1}.0|y_0^{l-1})}{\widetilde{W}_t^{(i)}(v_0^{i-1}.1|y_0^{l-1})}.$$

It can be seen [17] that

$$S_\lambda^{(2i)}(v_0^{2i-1}, y_0^{N-1}) = Q(a,b) = \operatorname{sign}(a)\operatorname{sign}(b)\min(|a|,|b|) \quad (5)$$

$$S_\lambda^{(2i+1)}(v_0^{2i}, y_0^{N-1}) = P(a,b,v_{2i}) = (-1)^{v_{2i}}a + b, \quad (6)$$

$$a = S_{\lambda-1}^{(i)}(v_{0,e}^{2i-1} \oplus v_{0,o}^{2i-1}, y_{0,e}^{N-1}), \quad (7)$$

$$b = S_{\lambda-1}^{(i)}(v_{0,o}^{2i-1}, y_{0,o}^{N-1}), \quad (8)$$

where $N = 2^\lambda$. The initial values for this recursion are given by $S_0^{(0)}(y_i) = \log \frac{W(0|y_i)}{W(1|y_i)}$. These expressions can be immediately recognized as the min-sum approximation of the SCL decoding [18]. However, these are also the exact values, which reflect the probability of the most likely valid continuation of a given path v_0^{i-1}.

Observe that one does not need to compute all values given by the recursion (5)–(6) at each phase. It is possible

to reuse intermediate LLRs $S_\lambda^{(i)} = S_\lambda^{(i)}(v_0^{i-1}|y_0^{N-1})$, obtained in previous phases. At phase $i > 0$ one needs to compute only values $S_{t-j}^{(\lfloor i/2^{s-j} \rfloor)}$, $0 \le j \le s$, where s is the largest integer such that 2^s divides i. Thus, the complexity of computing $S_t^{(i)}$ is $\sum_{j=0}^{s} 2^j$ operations, $i > 0$. See [16] or [19] for details.

The log-likelihood of a path v_0^i can be obtained [17] as

$$R_y(v_0^i) = R(v_0^i|y_0^{l-1}) = \log \widetilde{W}_t^{(i)}(v_0^i|y_0^{l-1}) = R_y(v_0^{i-1}) + \tau\left(S_t^{(i)}(v_0^{i-1}, y_0^{l-1}), v_i\right), \quad (9)$$

where $R_y(\epsilon)$ can be set to 0, ϵ is an empty sequence, and $\tau(S, v) = \begin{cases} 0, & \text{sign}(S) = (-1)^v \\ -|S|, & \text{otherwise.} \end{cases}$

E. Fundamental parameters of polar codes

1) Rate of polarization: The rate of polarization $E(K)$ shows how fast bit subchannels of $K^{\otimes m}$ approach either almost noiseless or pure noise channel with $n = l^m$. Suppose we construct (n, k) polar code $\mathcal{C}$ with kernel K. Let $P_e(n)$ be a block error probability of SC decoding of $\mathcal{C}$ under transmission over W. It was shown [6], that if $k/n < I(W)$ and $\beta < E(K)$, then for sufficiently large n the probability $P_e(n)$ can be bounded as $P_e(n) \le 2^{-n^\beta}$.

The rate of polarization is independent of channel W. The method of its computation is proposed in [6]. It is possible to obtain $l \times l$ kernels with the rate of polarization arbitrarily close to 1 by increasing the kernel dimension l to infinity [6]. Explicit constructions of kernels with high rate of polarization are provided in [20], [21].

2) Scaling exponent: Another crucial property of polarization kernels is the *scaling exponent*. Let us fix a binary discrete memoryless channel W of capacity $I(W)$ and a desired block error probability P_e. Given W and P_e, suppose we wish to communicate at rate $I(W) - \Delta$ using a family of (n, k) polar codes with kernel K. The value of n scales as $O(\Delta^{-\mu(K)})$, where the constant $\mu(K)$ is known as the scaling exponent [22]. The scaling exponent depends on the channel. Unfortunately, the algorithm of its computation is only known for the case of binary erasure channel [23], [22]. It is possible to show [24], [25] that there exist $l \times l$ kernels K_l, such that $\lim_{l\to\infty} \mu(K_l) = 2$, i.e. the corresponding polar codes provide optimal scaling behaviour. Constructions of the kernels with good scaling exponent are provided in [26], [27].

III. Window Processing

In this section we present a detailed description of the *window processing* (WP) algorithm for large polarization kernels. The basic idea of WP is to calculate the input symbols probabilities of the desired kernel K via probabilities of Arikan matrix F_t, which are easy to compute. This idea was originally proposed in [12] and reinvented in [13].

We also present an implementation of WP in LLR domain of approximated probabilities (4).

A. Relation between the input vectors of polarizing transforms

WP algorithm calculates the probabilities $\mathbf{W}_1^{(\phi)}(u_0^\phi|y_0^{l-1})$ of input symbols u_0^ϕ for $l \times l, l = 2^t$, kernel K using probabilities $W_t^{(i)}(v_0^i|y_0^{l-1}), i \ge \phi$ for Arikan matrix F_t. This operation requires establishing the relation between the input vectors u and v of polarizing transforms K and F_t.

Indeed, since K and F_t are invertible, we can write $TK = F_t$, where T is referred to as the *transition matrix*. Hence, any vector can be represented as $c_0^{l-1} = v_0^{l-1}F_t = u_0^{l-1}K$. This implies that $u_0^{l-1} = v_0^{l-1}T$, or

$$u_\phi = \sum_{s=0}^{l-1} v_s T[s, \phi] = \sum_{\substack{s=0 \\ T_{[s,\phi]}=1}}^{\tau_\phi} v_s, \quad (10)$$

where τ_ϕ denotes the position of the last non-zero symbol in the ϕ-th column of T. We rewrite (10) to obtain

$$v_{\tau_\phi} = u_\phi + \sum_{\substack{s=0 \\ T_{[s,\phi]}=1}}^{\tau_\phi - 1} v_s \quad (11)$$

However, some τ_ϕ might be equal. This means that some $v_i, i \in [l]$, are not explicitly defined by (11). Nevertheless, it is possible to use linear operations to transform (11) to derive the equations for all $v_i, i \in [l]$.

Indeed, the vectors u_0^{l-1} and v_0^{l-1} satisfy the equation

$$\Theta'(u_{l-1}, \dots, u_1, u_0, v_0, v_1, \dots, v_{l-1})^T = 0,$$

where $\Theta' = (\mathbb{T} \quad I)$, and $l \times l$ matrix $\mathbb{T}$ is obtained by transposing T^{-1} and reversing the order of columns in the obtained matrix. By applying elementary row operations, the matrix Θ' can be transformed into a minimum-span form Θ, such that the first and last non-zero elements of the ϕ-th row are located in columns ϕ and z_ϕ, respectively, where all z_ϕ are distinct. This enables one to obtain the symbols of the vector u as

$$u_\phi = \sum_{s=0}^{\phi-1} u_s \Theta_{l-1-\phi, l-1-s} + \sum_{j=0}^{\omega_\phi} v_j \Theta_{l-1-\phi, l+j}, \quad (12)$$

where $\omega_\phi = z_{l-1-\phi} - l$. Using (12), we write the equations for the vector v_0^{l-1}:

$$v_{\omega_\phi} = \sum_{s=0}^{\phi} u_s \Theta_{l-1-\phi, l-1-s} + \sum_{j=0}^{\omega_\phi - 1} v_j \Theta_{l-1-\phi, l+j}. \quad (13)$$

In this paper we consider kernels with all $\tau_\phi, \phi \in [l]$, distinct. In this case, we do not need to compute the matrix Θ to obtain (12). For such kernels we have $\omega_\phi = \tau_\phi, \phi \in [l]$, and can immediately use (10) to evaluate u from v.

Fig. 1 presents 16×16 kernels K'_{16} and K_{16}, which were constructed by algorithm given in [26]. They have improved polarization properties and admit low complexity WP. The expressions (10) for K'_{16} and K_{16} are given in Table I.

B. Decoding window

In the previous section we discussed the relation between input vectors u and v of polarizing transforms K and F_t. In

$K'_{16}, E = 0.51828, \mu = 3.346$

$$\begin{pmatrix}
1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
1&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0\\
1&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0\\
1&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0\\
1&1&0&0&0&0&0&0&1&1&0&0&0&0&0&0\\
1&0&1&0&0&0&0&0&1&0&1&0&0&0&0&0\\
1&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0\\
1&0&0&0&1&0&0&0&1&0&0&0&1&0&0&0\\
0&1&1&0&1&1&0&0&1&0&1&0&0&0&0&0\\
1&1&0&0&1&0&1&0&0&1&1&0&0&0&0&0\\
1&1&1&1&1&1&1&1&0&0&0&0&0&0&0&0\\
1&1&1&1&0&0&0&0&1&1&1&1&0&0&0&0\\
1&1&0&0&1&1&0&0&1&1&0&0&1&1&0&0\\
1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0\\
1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1
\end{pmatrix}$$

$K_{16}, E = 0.51828, \mu = 3.45$

$$\begin{pmatrix}
1&0&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
1&1&0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
1&0&1&0&0&0&0&0&0&0&0&0&0&0&0&0\\
1&1&1&1&0&0&0&0&0&0&0&0&0&0&0&0\\
1&0&0&0&1&0&0&0&0&0&0&0&0&0&0&0\\
1&0&0&0&0&0&0&0&1&0&0&0&0&0&0&0\\
1&1&0&0&0&0&0&0&1&1&0&0&0&0&0&0\\
1&0&1&0&0&0&0&0&1&0&1&0&0&0&0&0\\
0&1&1&0&1&1&0&0&1&0&1&0&0&0&0&0\\
1&1&0&0&1&0&1&0&0&1&1&0&0&0&0&0\\
1&1&1&1&1&1&1&1&1&0&0&0&0&0&0&0\\
1&1&1&1&0&0&0&0&1&1&1&0&0&0&0&0\\
1&0&0&0&1&0&0&0&1&0&0&0&1&0&0&0\\
1&1&0&0&1&1&0&0&1&1&0&0&1&1&0&0\\
1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0\\
1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1
\end{pmatrix}$$

Fig. 1: Large polarization kernels with improved polarization

TABLE I: Transition matrices of K'_{16}, K_{16} kernels

ϕ	K'_{16}			K_{16}		
	u_ϕ	$\mathcal{D}_\phi$	Cost	u_ϕ	$\mathcal{D}_\phi$	Cost
0	v_0	$\{\}$	15	v_0	$\{\}$	15
1	v_1	$\{\}$	1	v_1	$\{\}$	1
2	v_2	$\{\}$	3	v_2	$\{\}$	3
3	v_4	$\{3\}$	21	v_3	$\{\}$	1
4	v_8	$\{3,5,6,7\}$	127	v_4	$\{\}$	7
5	$v_6 \oplus v_9$	$\{3,5,6,7\}$	48	v_8	$\{5,6,7\}$	67
6	$v_5 \oplus v_6 \oplus v_{10}$	$\{3,5,6,7\}$	95	$v_6 \oplus v_9$	$\{5,6,7\}$	24
7	v_3	$\{5,6,7\}$	1	$v_5 \oplus v_6 \oplus v_{10}$	$\{5,6,7\}$	47
8	v_{12}	$\{5,6,7,11\}$	127	v_5	$\{6,7\}$	1
9	v_5	$\{6,7,11\}$	1	v_6	$\{7\}$	1
10	v_6	$\{7,11\}$	1	v_7	$\{\}$	1
11	v_7	$\{11\}$	1	v_{11}	$\{\}$	1
12	v_{11}	$\{\}$	1	v_{12}	$\{\}$	7
13	v_{13}	$\{\}$	1	v_{13}	$\{\}$	1
14	v_{14}	$\{\}$	3	v_{14}	$\{\}$	3
15	v_{15}	$\{\}$	1	v_{15}	$\{\}$	1

this section we use this relation to compute the probabilities of K via the probabilities of F_t.

Using (12) we can reconstruct u_0^ϕ from $v_0^{h_\phi}$, where $h_\phi = \max_{0 \le \phi' \le \phi} \tau_{\phi'}$. Using this fact, we want to express the probability $\mathbf{W}_1^{(\phi)}(u_0^\phi | y_0^{l-1})$ via $W_t^{(h_\phi)}(v_0^{h_\phi} | y_0^{l-1})$. However, if $h_\phi > \phi$, then some values of $v_0^{h_\phi}$ are independent from u_0^ϕ and, therefore, unknown. This suggests that to compute $\mathbf{W}_1^{(\phi)}(u_0^\phi | y_0^{l-1})$ we need to consider the probabilities $W_t^{(h_\phi)}(v_0^{h_\phi} | y_0^{l-1})$ for vectors $v_0^{h_\phi}$ with all possible values of unknown bits.

Let

$$\mathcal{D}_\phi = [h_\phi + 1] \setminus \{\omega_0, \omega_1, \dots, \omega_\phi\}$$

be a *decoding window*, i.e. the set of indices of independent (from u_0^ϕ) components of $v_0^{h_\phi}$. Note that

$$|\mathcal{D}_\phi| = |[h_\phi + 1]| - |\{\omega_0, \omega_1, \dots, \omega_\phi\}| = h_\phi - \phi,$$

since all ω_ϕ are distinct and $\{\omega_0, \omega_1, \dots, \omega_\phi\} \subseteq [h_\phi + 1]$.

Theorem 1. *Let K be an $l \times l, l = 2^t$, polarization kernel. Then, the probability of the input vector u_0^ϕ of K is given by*[2]

$$\mathbf{W}_1^{(\phi)}(u_0^\phi | y_0^{l-1}) = \sum_{v_0^{h_\phi} \in \mathcal{Z}_\phi^{(u_\phi)}} W_t^{(h_\phi)}(v_0^{h_\phi} | y_0^{l-1}), \qquad (14)$$

where $\mathcal{Z}_\phi^{(b)}$ is the set of vectors $v_0^{h_\phi}$, such as $v_s \in \mathbb{F}_2, s \in \mathcal{D}_\phi$, the values of $v_t, t \in [h_\phi + 1] \setminus \mathcal{D}_\phi$, are obtained according to expression (13) *under condition that $u_\phi = b$.*

[2]The method proposed in [13] is a special case of this approach.

Proof. By definition (2), we have

$$\mathbf{W}_1^{(\phi)}(u_0^\phi | y_0^{l-1}) = \sum_{u_{\phi+1}^{l-1} \in \mathbb{F}_2^{l-\phi-1}} \mathbf{W}_1^{(l-1)}(u_0^{l-1} | y_0^{l-1}) =$$

$$\sum_{u_{\phi+1}^{l-1} \in \mathbb{F}_2^{l-\phi-1}} \prod_{i=0}^{l-1} W((u_0^{l-1}K)_i | y_i) \stackrel{a}{=}$$

$$\sum_{v_0^{h_\phi} \in \mathcal{Z}_\phi^{(u_\phi)}, v_{h_\phi+1}^{l-1} \in \mathbb{F}_2^{l-h_\phi-1}} \prod_{i=0}^{l-1} W(((v_0^{l-1}T)K)_i | y_i) \stackrel{b}{=}$$

$$\sum_{v_0^{h_\phi} \in \mathcal{Z}_\phi^{(u_\phi)}} \sum_{v_{h_\phi+1}^{l-1} \in \mathbb{F}_2^{l-h_\phi-1}} \prod_{i=0}^{l-1} W((v_0^{l-1}F_t)_i | y_i) \stackrel{c}{=}$$

$$\sum_{v_0^{h_\phi} \in \mathcal{Z}_\phi^{(u_\phi)}} \sum_{v_{h_\phi+1}^{l-1} \in \mathbb{F}_2^{l-h_\phi-1}} W_t^{(l-1)}(v_0^{l-1} | y_0^{l-1}) =$$

$$\sum_{v_0^{h_\phi} \in \mathcal{Z}_\phi^{(u_\phi)}} W_t^{(h_\phi)}(v_0^{h_\phi} | y_0^{l-1}).$$

Equality (a) follows from (12) and definition of $\mathcal{Z}_\phi^{(u_\phi)}$. Note that $|\mathcal{Z}_\phi^{(u_\phi)}| \cdot |\mathbb{F}_2^{l-h_\phi-1}| = |\mathbb{F}_2^{l-\phi-1}|$. Equality (b) follows from the definition $TK = F_t$. Equality (c) is due to (2). □

The expression (14) is the WP algorithm in the probabilistic domain. It means that computation of the probability $\mathbf{W}_1^{(\phi)}(u_0^\phi | y_0^{l-1})$ requires considering $2^{|\mathcal{D}_\phi|+1}$ paths $v_0^{h_\phi} \in \mathcal{Z}_\phi^{(u_\phi)}$ in context of Arikan SC decoding and computing $2^{|\mathcal{D}_\phi|+1}$ probabilities $W_t^{(h_\phi)}(v_0^{h_\phi} | y_0^{l-1})$ for each path.

Let $\mathcal{M}(K)$ denote the $\max_{i \in [l]} |\mathcal{D}_i|$ for kernel K. In general, one has $\mathcal{M}(K) = O(l)$ for an arbitrary K. In this case, the complexity of computing (14) is given by $O(2^l)$ arithmetic operations. Fortunately, there are some methods [26], [28] to construct the polarization kernels with the reduced $\mathcal{M}(K)$, which allow low complexity WP.

Example 1. *Consider computing $\mathbf{W}_1^{(6)}(u_0^6 | y_0^{15})$ for K'_{16}. By definition* (2) *one should compute*

$$\mathbf{W}_1^{(6)}(u_0^6 | y_0^{15}) = \sum_{u_7^{15} \in \mathbb{F}_2^9} \mathbf{W}_1^{(15)}(u_0^{15} | y_0^{15}). \qquad (15)$$

The value of $h_6 = 10$ indicates the largest index of the dependent (from u_0^6) element of v. It implies that we should consider the probabilities $W_4^{(10)}(v_0^{10} | y_0^{15})$ of paths v_0^{10}. However, in v_0^{10} we have 4 independent (unknown) elements: v_3 and v_5^7. Indices of these elements form the decoding window $\mathcal{D}_6 = \{3, 5, 6, 7\}$. The expressions for each v_i are given by (11). *This enables us to construct the set $\mathcal{Z}_6^{(u_6)}$ of paths v_0^{10}.*

In this example we have $\mathcal{Z}_6^{(u_6)} = \{v_0^{10} | v_i \in \mathbb{F}_2, i \in \mathcal{D}_6, v_j = u_j, j \in [3], v_4 = u_3, v_8 = u_4, v_9 = u_6 \oplus v_5, v_{10} = v_5 \oplus v_6 \oplus u_6\}$ and $|\mathcal{Z}_6| = 16$. Thus, according to Theorem 1,

$$\mathbf{W}_1^{(6)}(u_0^6 | y_0^{15}) = \sum_{v_0^{10} \in \mathcal{Z}_6^{(u_6)}} W_4^{(10)}(v_0^{10} | y_0^{15}).$$

The decoding windows of K'_{16}, K_{16} are presented in Table I together with WP complexity of each phase. The detailed description of K_{16} processing is presented in Section VI-B. The computation of WP complexity of K'_{16} is given in [29].

C. Log-likelihood domain

Similarly to (3), we can rewrite (14) for the case of approximate probabilities

$$\widetilde{\mathbf{W}}_1^{(\phi)}(u_0^{\phi}|y_0^{l-1}) = \max_{v_0^{h_\phi} \in \mathcal{Z}_\phi^{(u_\phi)}} \widetilde{W}_t^{(h_\phi)}(v_0^{h_\phi}|y_0^{l-1}) = \max_{v_0^{h_\phi} \in \mathcal{Z}_\phi^{(u_\phi)}} \max_{v_{h_\phi+1}^{l-1}} W_t^{(l-1)}(v_0^{l-1}|y_0^{l-1}). \quad (16)$$

Using (9), we can obtain the LLRs

$$\mathbf{S}_{1,\phi} = \max_{v_0^{h_\phi} \in \mathcal{Z}_\phi^{(0)}} R_y(v_0^{h_\phi}) - \max_{v_0^{h_\phi} \in \mathcal{Z}_\phi^{(1)}} R_y(v_0^{h_\phi}). \quad (17)$$

Calculation of LLRs $\mathbf{S}_{1,\phi}$ via (17) is referred to as the WP algorithm in LLR domain. Let $\mathcal{Z}_\phi = \mathcal{Z}_\phi^{(0)} \cup \mathcal{Z}_\phi^{(1)}$. The number of path scores to be computed in (17) is equal to $|\mathcal{Z}_\phi| = 2^{|\mathcal{D}_\phi|+1}$. It is also possible to reuse the intermediate LLRs for each path $v_0^{h_\phi}$ (see Section II-D).

Although the number of path scores $R_y(v_0^{h_\phi})$, $v_0^{h_\phi} \in \mathcal{Z}_\phi$, is $2^{|\mathcal{D}_\phi|+1}$, they can be computed with $2^{|\mathcal{D}_\phi|}$ operations. Namely, during the calculation of each $R_y(v_0^{h_\phi})$, both terms $\tau(S_t^{(h_\phi)}(v_0^{h_\phi-1}, y_0^{l-1}), b)$, $b \in \mathbb{F}_2$, arise. Due to the definition of τ, there is such value of $a \in \mathbb{F}_2$, that $\text{sign}(S_t^{(h_\phi)}(v_0^{h_\phi-1}, y_0^{l-1})) = (-1)^a$, thus, $\tau(S_t^{(h_\phi)}(v_0^{h_\phi-1}, y_0^{l-1}), a) = 0$. It remains to compute only $\tau(S_t^{(h_\phi)}(v_0^{h_\phi-1}, y_0^{l-1}), a \oplus 1)$. The number of such nonzero terms is $2^{|\mathcal{D}_\phi|}$.

The equations (17) and (5)–(9) can be immediately used to perform kernel processing. However, in the next sections we show how to substantially reduce the complexity of WP.

IV. Common Subexpressions Identification

In this section we propose a simplified version of WP algorithm. Conventional WP considers several paths in the context of Arikan SC decoding. For each path, the algorithm recursively computes its LLR (which is used to obtain the path score). It turns out that the intermediate LLRs induced by this recursion can be equal for different paths. Thus, one can compute only the unique intermediate LLRs, and, therefore, substantially reduce the processing complexity.

A. Motivation

Suppose we want to compute ϕ-th input symbol LLR $\mathbf{S}_{1,\phi}$ of the $l \times l, l = 2^t$, polarization kernel K by WP method. It computes several scores $R_y(v_0^{h_\phi})$ of paths $v_0^{h_\phi}$, considered in the context of Arikan SC decoding. The number of such paths is determined by the size of the ϕ-th decoding window $\mathcal{D}_\phi$. The values ϕ and h_ϕ are referred to as *external phase* and *internal phase*.

According to (9), to obtain a single path score $R_y(v_0^{h_\phi})$ one needs to compute the LLR $S_t^{(h_\phi)} = S_t^{(h_\phi)}(v_0^{h_\phi-1}, y_0^{l-1})$ and the path score $R_y(v_0^{h_\phi-1})$ from the previous phase $h_\phi - 1$. The value $S_t^{(h_\phi)}$ is computed recursively via (5)–(8). On each layer $\lambda \in [t+1]$ of this recursion $M = 2^{t-\lambda}$ intermediate LLRs $S_\lambda^{(j)} = S_\lambda^{(j)}(\bar{v}_0^{j-1}, \bar{y}_0^{N-1}), N = 2^\lambda$, are arising. The value $\bar{v}_0^{j-1}$ denotes a partial sum of some elements from $v_0^{h_\phi-1}$ and $\bar{y}_0^{N-1}$ is a particular subvector of y_0^{l-1}.

It is possible to write $\bar{v}_0^{j-1}$ and $\bar{y}_0^{N-1}$ explicitly for all $S_\lambda^{(j)}$. We introduce an array $C_\lambda^{(j)}[\beta], \beta \in [M]$, of partial sums $\bar{v}_0^{j-1}$ at layer λ

$$C_\lambda^{(j)}[\beta] = \begin{cases} (c_\beta^{(0)}, c_\beta^{(1)}, \dots, c_\beta^{(j-1)}), & j > 0, \\ \epsilon, & \text{otherwise}, \end{cases} \quad (18)$$

where $c^{(i)} = v_{M\cdot i}^{M\cdot i+M-1} F_{t-\lambda}, i \in [j]$. We also define an array $S_\lambda^{(j)}[\beta], \beta \in [M]$, of the intermediate LLRs $S_\lambda^{(j)}(\bar{v}_0^{j-1}, \bar{y}_0^{N-1})$

$$S_\lambda^{(j)}[\beta] = S_\lambda^{(j)}(C_\lambda^{(j)}[\beta], (y_\beta, y_{\beta+M}, \dots, y_{\beta+M(N-1)})). \quad (19)$$

Similarly to [16], these LLRs can be computed as

$$S_\lambda^{(j)}[\beta] = \begin{cases} Q(S_{\lambda-1}^{(\hat{j})}[\beta], S_{\lambda-1}^{(\hat{j})}[\beta+N]), j \text{ even}, \\ P(S_{\lambda-1}^{(\hat{j})}[\beta], S_{\lambda-1}^{(\hat{j})}[\beta+N], (C_\lambda^{(j)}[\beta])_{j-1}), j \text{ odd}, \end{cases}$$

where $\hat{j} = \lfloor j/2 \rfloor$.

WP method requires calculation of $2^{|\mathcal{D}_\phi|}$ LLRs $S_t^{(h_\phi)}$ given by $v_0^{h_\phi-1} \in \mathcal{Z}_\phi$. For convenience, we define a set

$$\mathbb{S}_\phi^{(h)} = \left\{ S_t^{(h)}(v_0^{h-1}, y_0^{l-1}) | v_0^{h_\phi-1} \in \mathcal{Z}_\phi \right\}, h \le h_\phi.$$

In a straightforward implementation, all intermediate LLRs $S_\lambda^{(j)}[\beta]$ should be separately computed for each $S_t^{(h_\phi)} \in \mathbb{S}_\phi^{(h_\phi)}$. It turns out that $S_\lambda^{(j)}[\beta]$ induced by recursive calculation of $S_t^{(h_\phi)}$ for different $v_0^{h_\phi-1} \in \mathcal{Z}_\phi$ may be the same. For example, the number of distinct vectors $\bar{v}_0^{j-1}$ and length-N subvectors $\bar{y}_0^{N-1}$ of y_0^{l-1} arising in (5)–(6) is at most 2^j and $2^{t-\lambda}$, respectively. Thus, the number of distinct pairs $(\bar{v}_0^j, \bar{y}_0^{N-1}) \le 2^{j+t-\lambda}$, which can be even less than $2^{|\mathcal{D}_\phi|}$.

We propose to compute $S_\lambda^{(j)}[\beta] = S_\lambda^{(j)}(\bar{v}_0^j, \bar{y}_0^{N-1})$ only for distinct pairs $(\bar{v}_0^j, y_0^{N-1})$ arising at each layer of the recursion (5)–(8). We denote by $X_\lambda = X_\lambda^{(h_\phi)}$, $\lambda \in [t+1]$, the set of these distinct pairs $(\bar{v}_0^j, \bar{y}_0^{N-1})$ and refer it to as a set of *common subexpressions* (CSE). In the case when $h_\phi - 1$ is greater than $h_{\phi-1}$, one needs to compute several sets $\mathbb{S}_\phi^{(h)}$ using CSE for each phase h, where $h_{\phi-1} < h < h_\phi$.

B. The CSE identification algorithm

The sets X_λ of CSE can be identified offline for a given polarization kernel K of size $l = 2^t$, external phase ϕ and phase h, $h_{\phi-1} < h < h_{\phi+1}$. To do that, one should run the procedure *GetCSEPairs*(h, ϕ) illustrated in Alg. 1. This procedure iteratively enumerates the pairs $(\bar{v}_0^{j-1}, \bar{y}_0^{N-1})$ according to the recursion in (5)–(8), and stores the unique ones only. It starts with pairs $(v_0^{h-1}|y_0^{l-1})$ in symbolic form, where $v_0^{h-1} \in \widehat{\mathcal{Z}}_{\phi,h}$, $\widehat{\mathcal{Z}}_{\phi,h} = \left\{ v_0^{h-1} | v_i \in \{0,1\}, i \in \mathcal{D}_\phi, u_j = \widehat{u}_j, j \in [\phi] \right\}$. Recall that $\widehat{u}_i$ is already estimated by SC decoding. Observe that $X_0 = \left\{ S_0^{(0)}(\epsilon, y_i), i \in [l] \right\}$ for any h.

Algorithm 1: GetCSEPairs(h, ϕ)

1 $\widehat{\mathcal{Z}}_{\phi,h} = \left\{v_0^{h-1} | v_i \in \{0,1\}, i \in \mathcal{D}_\phi, u_j = \widehat{u}_j, j \in [\phi]\right\}$;
2 $y_0^{l-1} \leftarrow (y_0, y_1, \ldots, y_{l-1})$ in symbolic form;
3 $X_t \leftarrow \left\{(v_0^{h-1}, y_0^{l-1}) | v_0^{h-1} \in \widehat{\mathcal{Z}}_{\phi,h}\right\}$;
4 $I \leftarrow h$;
5 **for** $\lambda \leftarrow t$ ***downto*** 1 **do**
6 $X_{\lambda-1} \leftarrow \emptyset$; $N \leftarrow 2^\lambda$;
7 $I' \leftarrow I - (I \mod 2)$;
8 **for** ***each*** $(\bar{v}_0^{I-1}, \bar{y}_0^{N-1})$ ***in*** X_λ **do**
9 $X_{\lambda-1} \leftarrow X_{\lambda-1} \cup \left\{(v_{0,e}^{I'-1} \oplus v_{0,o}^{I'-1}, y_{0,e}^{N-1})\right\}$;
10 $X_{\lambda-1} \leftarrow X_{\lambda-1} \cup \left\{(v_{0,o}^{I'-1}, y_{0,o}^{N-1})\right\}$;
11 $I \leftarrow \lfloor I/2 \rfloor$;
12 **return** $\{X_\lambda | \lambda \in [t+1]\}$

Algorithm 2: ComputeLLRs(X, h, ϕ)

1 $S_0^{(0)}(\epsilon, y_i) \leftarrow \log \frac{W(0|y_i)}{W(1|y_i)}, i \in [l]$;
2 **for** $\lambda \leftarrow 1$ ***to*** t **do**
3 $I \leftarrow \lfloor h/2^{t-\lambda} \rfloor$;
4 $I' \leftarrow I - (I \mod 2)$;
5 $N \leftarrow 2^{t-\lambda}$;
6 **for** ***each*** $(\bar{v}_0^{I-1}, \bar{y}_0^{N-1})$ ***in*** X_λ **do**
7 $a \leftarrow S_{\lambda-1}^{(I'/2)}(v_{0,e}^{I'-1} \oplus v_{0,o}^{I'-1}, y_{0,e}^{N-1})$;
8 $b \leftarrow S_{\lambda-1}^{(I'/2)}(v_{0,o}^{2i-1}, y_{0,o}^{N-1})$;
9 **if** I ***is even*** **then**
10 $S_\lambda^{(I)}(\bar{v}_0^{I-1}, \bar{y}_0^{N-1}) \leftarrow$ $\operatorname{sign}(a)\operatorname{sign}(b)\min(|a|, |b|)$;
11 **else**
12 evaluate $\bar{v}_I$ using $\widehat{u}_0^{\phi-1}$ as $\bar{c}$;
13 $S_\lambda^{(I)}(\bar{v}_0^{I-1}, \bar{y}_0^{N-1}) \leftarrow (-1)^{\bar{c}} a + b$;
14 **return** $\mathbb{S}_\phi^{(h)}$

C. LLR computation

Finally, to obtain the LLRs $\mathbb{S}_\phi^{(h)}$ at the external phase ϕ of the kernel processing, one should run *ComputeLLRs*(X, h, ϕ), presented in Alg. 2. It uses CSE pairs $(\bar{v}_0^{j-1}, \bar{y}_0^{N-1}) \in X_\lambda, \lambda \in \{1, \ldots, t\}$, and iteratively computes all intermediate LLRs $S_\lambda^{(j)}(\bar{v}_0^{j-1}, \bar{y}_0^{N-1})$ corresponding to these pairs.

We would like to emphasise that CSE pairs $(\bar{v}_0^{j-1}, \bar{y}_0^{N-1}) \in X_\lambda$ are used in symbolic form, while the corresponding LLRs $S_\lambda^{(j)}(\bar{v}_0^{j-1}, \bar{y}_0^{N-1})$ are obtained as evaluated numbers in Alg. 2. The pairs $(\bar{v}_0^{j-1}, \bar{y}_0^{N-1})$ are used as indices of $S_\lambda^{(j)}$. We assume that one can access any value $S_\lambda^{(j)}$ in constant time, since no sorting is required in Alg. 2. In a software implementation, one can replace the pairs $(\bar{v}_0^{j-1}, \bar{y}_0^{N-1})$ by integer indices and consider arrays of $S_\lambda^{(j)}$.

Example 2. *Consider the identification of CSE for kernel K_{16} at external phase 5 and internal phase $h_5 = 8$. The decoding window is $\mathcal{D}_5 = \{5, 6, 7\}$. Thus, under condition $u_0^4 = \widehat{u}_0^4$ we have $\widehat{\mathcal{Z}}_{5,8} = \left\{v_0^7 | \widehat{u}_0^4 . v_5^7, v_i = \{0,1\}, i \in \mathcal{D}_5\right\}$, which is initialized at line 1 of Alg. 1.*

Computation of X_λ, $0 \le \lambda \le 4$, by GetCSEPairs$(8, 5)$ is illustrated in Fig. 2. The set X_4 is initialized as $\left\{(v_0^7 | y_0^{15}) | v_0^7 \in \widehat{\mathcal{Z}}_5^{(8)}\right\}$ at line 3. The elements of X_3 are obtained from elements of X_4 at lines 7-10. Each pair $(v_0^7 | y_0^{15}) \in X_4$ is connected with two distinct pairs $(\bar{v}_0^3 | \bar{y}_0^7) \in X_3$. One of them is given by (7) *and connected by a black solid line, while another one is given by* (8) *and connected by a red dashed line. This process is performed at lines 9–10. The same process is performed to obtain the sets X_2, X_1, X_0.*

In a straightforward implementation of WP, for each value of $S_4^{(8)} \in \mathbb{S}_5^{(8)}$ one should compute 2 values of $S_3^{(4)}$, 4 values of $S_2^{(2)}$, and 8 values of $S_1^{(1)}$. This results in $8 \cdot (1+2+4+8) = 120$ operations. According to Fig. 2, we have $|X_4| + |X_3| + |X_2| + |X_1| = 8 \cdot 3 + 16 = 40$, which implies that we can compute $\mathbb{S}_5^{(8)}$ only with 40 operations using Alg. 2.

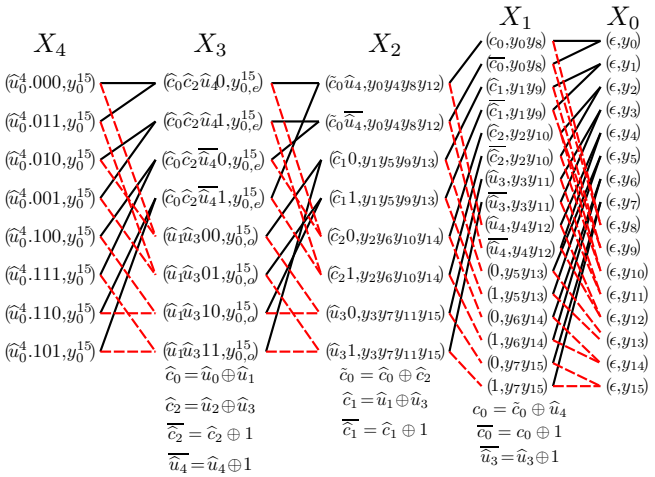


Fig. 2: CSE of phase 5 of K_{16} kernel

D. Reusing of CSE at different phases

Arikan SC decoding allows one to reuse intermediate LLRs to compute LLR for the next phases as we mentioned in Section II-D. Similar approach can be used in WP with CSE.

Recall that in Arikan SC decoding at phase $i > 0$ one needs to recompute only the intermediate LLRs $S_{t-k}^{(\lfloor i/2^{s-k} \rfloor)}$, $0 \le k \le s$, where $s = \psi(i)$ is the largest integer such that 2^s divides i. Observe that $S_{t-s}^{(\lfloor i/2^s \rfloor)}$ is calculated via P function (6) using partial sums $C_{t-s}^{(\lfloor i/2^s \rfloor)}[\beta]$, while remaining $S_{t-k}^{(\lfloor i/2^{s-k} \rfloor)}$, $0 \le k < s$, are computed via Q function (5).

The above principle is applicable to the case of WP with CSE as well. Consider processing of the ϕ-th phase of kernel K. Suppose we compute all CSE at the internal phase h_ϕ. Let $s' = \psi(h+1)$, and $|\mathcal{D}_\phi| \ge |\mathcal{D}_{\phi+1}|$. In this case, $X_\lambda^{(h_\phi+1)} \subseteq X_\lambda^{(h_\phi)}$, $\lambda \in [s']$. Thus, at phase $h_\phi + 1$ we need to recompute only CSE LLRs $S_\lambda^{(\lfloor h_\phi / 2^{t-\lambda} \rfloor)}$ at layers $s' \le \lambda \le t$. In other words, one can start Alg. 2 from layer s'. In contrast, in case of $|\mathcal{D}_\phi| < |\mathcal{D}_{\phi+1}|$ we have $X_\lambda^{(h_\phi)} \subseteq X_\lambda^{(h_\phi+1)}$, $\lambda \in [t+1]$, and need to update all X_λ. However, even in this case we can reuse $|X_\lambda^{(h_\phi+1)}| / |X_\lambda^{(h_\phi)}|$ LLRs at layers $\lambda \in [s']$.

Example 3. *At external phase $\phi = 5$ of kernel K'_{16} we have $u_5 = v_6 \oplus v_9$, $h_5 = 9$, and $\mathcal{D}_5 = \mathcal{D}_4$. In this case, $X_\lambda^{(5)} = X_\lambda^{(4)}, \lambda \in [4]$. Thus, one can start Alg. 2 from layer 4. The set $\mathbb{S}_4^{(9)}$ is computed at lines 12–13 with v_8 evaluated as $\widehat{u}_4$.*

As a result, arithmetic complexity (number of summation and comparison operations) of computing $\mathbb{S}_\phi^{(h)}$ is given by $\sum_{\lambda=s'}^{t} |X_\lambda|$, while straightforward implementation of the WP requires $|\mathcal{D}_\phi| \cdot \sum_{\lambda=\lambda'}^{t} 2^{t-\lambda}$ operations. It can be seen that

$$\sum_{\lambda=\lambda'}^{t} |X_\lambda| \le |\mathcal{D}_\phi| \cdot \sum_{\lambda=\lambda'}^{t} 2^{t-\lambda}$$

by definition of the CSE.

V. Further improvements of window processing

In this section we further simplify window processing by exploiting some properties of maximization of path scores.

A. Path score computation

We consider WP of $l \times l$ polarization kernel K. Suppose for some external phase ϕ we have $h_\phi > h_{\phi-1} + 1$, or, in other words, $|\mathcal{D}_\phi| > |\mathcal{D}_{\phi-1}|$. As we discussed in Section IV-A, in this case we need to compute several input symbols LLRs $S_t^{(h)}$, where $h_{\phi-1} < h < h_\phi$.

Essentially, in this case WP requires to compute path scores

$$R_y(v_0^{h_\phi}) = R_y(v_0^{h_\phi - 1}) + \tau\left(S_t^{(h_\phi)}, v_{h_\phi}\right) = R_y(v_0^{h_{\phi-1}}) + \sum_{h=h_{\phi-1}+1}^{h_\phi} \tau\left(S_t^{(h)}, v_h\right). \quad (20)$$

There is a simple way to compute the last term in (20). Let

$$E(c_0^{n-1}, s_0^{n-1}) = \sum_{i=0}^{n-1} \tau(s_i, c_i)$$

be an ellipsoidal weight (also known as correlation discrepancy) of vector $c_0^{n-1} \in \mathbb{F}_2^n$ with respect to s_0^{n-1} [30], [31].

Theorem 2 ([19])**.** *The ellipsoidal weight of the vector $v_0^{2^t-1}F_t$ with respect to the input LLRs s is equal to the score of the path $v_0^{2^t-1}$ in the SC decoder, i.e.*

$$E(v_0^{2^t-1}F_t, s) = \sum_{i=0}^{2^t-1} \tau(S_t^{(i)}(v_0^{i-1}, y_0^{2^t-1}), v_i),$$

where $s = (S_0^{(0)}(y_0), \dots, S_0^{(0)}(y_{2^m-1}))$.

Theorem 2 implies the following:

Corollary 1. *Consider Arikan SC decoding of $(n = 2^t, k)$ polar code at phase $\phi = i + 2^q - 1$, where q is the largest integer such that 2^q divides i. Then,*

$$\sum_{\beta=i}^{i+2^q-1} \tau(S_t^{(\beta)}(v_0^{\beta-1}, y_0^{n-1}), v_j) = \sum_{\beta=0}^{2^q-1} \tau(S_{\lambda-q}^{(\lfloor \phi/2^q \rfloor)}[\beta], c_\beta),$$

where $c_0^{2^q-1} = v_i^{i+2^q-1}F_q$.

Corollary 1 allows one to avoid the computation of some $S_t^{(h)}$ in (20) and use $S_{t-q}^{(\lfloor h/2^q \rfloor)}[\beta]$ instead. In some cases, one can obtain $R(v_0^{h_\phi}|y_0^{n-1})$ in a lower number of operations.

Example 4. *Consider computing the LLR for u_4 of K'_{16} kernel. The decoding window $\mathcal{D}_4 = \{3, 5, 6, 7\}$. According to* (9)*, one has*

$$R_y(v_0^8) = R_y(v_0^7) + \tau\left(S_4^{(8)}(v_0^7, y_0^{15}), v_8\right).$$

By definition, we have

$$R_y(v_0^7) = \sum_{j=0}^{7} \tau(S_4^{(j)}(v_0^{j-1}, y_0^{15}), v_j). \quad (21)$$

Thus, to compute all $R_y(v_0^7)$, $v_0^7 \in \mathcal{Z}_4$, one needs to obtain intermediate LLRs $\mathbb{S}_4^{(5)}, \mathbb{S}_4^{(6)}$ and $\mathbb{S}_4^{(7)}$ and sum corresponding $\tau(S_4^{(h)}, v_h)$. Using CSE and properties of τ function, this can be done with 36 operations.

Using Corollary 1, one can rewrite (21) *in the following way:*

$$R_y(v_0^7) = R_y(v_0^3) + \sum_{\beta=0}^{3} \tau(S_2^{(1)}[\beta], (v_4^7 F_2)_\beta). \quad (22)$$

According to (19)*,*

$$S_2^{(1)}[\beta] = S_2^{(1)}((v_0^3 F_2)_\beta, (y_\beta, y_{\beta+4}, y_{\beta+8}, y_{\beta+12})).$$

Note that $R_y(v_0^3)$ was computed at the previous phase. Unlike (20)*, one does not need to compute $\mathbb{S}_4^{(5)}, \mathbb{S}_4^{(6)}$ and $\mathbb{S}_4^{(7)}$ in* (22)*. To compute the last term of* (22) *for all $v_0^7 \in \mathcal{Z}_4$ we first observe that $\bar{\mathcal{Z}}_b = \{v_4^7 F_2 | v_4 = b, v_5^7 \in \mathbb{F}_2^3\}$ is a coset of Reed-Muller code $RM(1,2) \oplus b(F_2[0])$. Furthermore,*

$$\sum_{\beta=0}^{3} \tau(S_2^{(1)}[\beta], c_\beta) = \frac{1}{2}\left(\sum_{\beta=0}^{3} (-1)^{c_\beta} S_2^{(1)}[\beta] - \sum_{\beta=0}^{3} |S_2^{(1)}[\beta]|\right), \quad (23)$$

where $c_0^3 = v_4^7 F_2$. Assume for simplicity that $v_4 = 0$. Then, using the fast Hadamard transform (FHT) [32], one can obtain the term $\sum_{\beta=0}^{3} (-1)^{c_\beta} S_2^{(1)}[\beta]$ for all $v_4^7 \in \bar{\mathcal{Z}}_0$ simultaneously in 8 operations. The term $\sum_{\beta=0}^{3} |S_2^{(1)}[\beta]|$ does not need to be computed, since it is constant and cancels in (17)*.*

Recall that we have the index $3 \in \mathcal{D}_4$, so, we have two arrays $S_2^{(1)}[\beta]$ of intermediate LLRs computed for $v_3 \in \{0, 1\}$. Fortunately, these LLRs were already obtained at phase 3, since $\mathcal{D}_3 = \{3\}$. Thus, to compute $R_y(v_0^7)$ for all $v_0^7 \in \mathcal{Z}_4$ we need to evaluate (23) *twice for different $S_2^{(1)}[\beta]$ and add the results to $R_y(v_0^3)$. We can also set $R_y(v_0^2) = 0$, since v_0^2 is fixed, so, only 8 values of $R_y(v_0^3)$ is nonzero, see Section III-C. As a result, Corollary 1 and FHT allow us to reduce the path score computation from 36 to 24 operations.*

B. Recursive maximum computation

Consider the case when the decoding window is monotonically decreasing at the next phases of window processing. In this case, the complexity of (17) can be drastically reduced.

Let $h_\phi = h_{\phi+1} = \dots = h_{\phi+q}$ at some phase ϕ. It implies that $|\mathcal{D}_{\phi+i}| = |\mathcal{D}_\phi| - i, i \in [q+1]$, and, therefore $\mathcal{Z}_{\phi+q} \subset$

$\mathcal{Z}_{\phi+q-1} \subset \cdots \subset \mathcal{Z}_{\phi}$. Thus, each path score $R_y(v_0^{h_{\phi+i}})$ for $v_0^{h_{\phi+i}} \in \mathcal{Z}_{\phi+i}$ in fact is computed at the phase ϕ. Recall that WP (17) for phase $\phi+i$ requires the maximization of $R_y(v_0^{h_{\phi+i}})$ over $v_0^{h_{\phi+i}} \in \mathcal{Z}_{\phi+1}^{(b)}$, $b \in \mathbb{F}_2$. Since $h_{\phi+i} = h_{\phi}$ and $\mathcal{Z}_{\phi+i} \subset \mathcal{Z}_{\phi}$, we propose to perform this maximization during the computation of $\mathbf{S}_{1,\phi}$. To do this, it is convenient to define the array

$$M_i[\bar{u}_{\phi}^{\phi+i-1}.b] = \max_{v_0^{h_{\phi+i}} \in \mathcal{Z}_{\phi+i}^{(b)}, \quad u_{\phi}^{\phi+i-1} = \bar{u}_{\phi}^{\phi+i-1}} R_y(v_0^{h_{\phi}}), \quad (24)$$

where $i \in [q+1]$. This array can be defined recursively as

$$M_i[\bar{u}_{\phi}^{\phi+i-1}.b] = \max(M_{i+1}[\bar{u}_{\phi}^{\phi+i-1}.b.0], M_{i+1}[\bar{u}_{\phi}^{\phi+i-1}.b.1]). \quad (25)$$

The base of this recursion is given by $M_q[\bar{u}_{\phi}^{\phi+q-1}.b], b \in \mathbb{F}_2$, which should be computed by (24). As a result, after recursive computation of $M_0[b], b \in \mathbb{F}_2$, according to (17), the $(\phi+i)$-th LLR $\mathbf{S}_{1,\phi+i}$ can be obtained in one subtraction as $M_i[\widehat{u}_{\phi}^{\phi+i-1}.0] - M_i[\widehat{u}_{\phi}^{\phi+i-1}.1]$, where symbols $\widehat{u}_{\phi}^{\phi+i-1}$ are estimated by the decoder. Observe that the number of comparisons to obtain $M_0[b]$ by (25) is the same as for straightforward maximization $\max_{v_0^{h_{\phi}} \in \mathcal{Z}_{\phi}^{(b)}} R_y(v_0^{h_{\phi}})$ in (17), we just propose to find this maximum in such way that intermediate comparisons can be used at next q phases.

Example 5. *Consider external phase 7 of K_{16} kernel processing. For this phase, $h_7 = h_8 = h_9 = h_{10} = 10$, $\mathcal{D}_7 = \{5,6,7\}$, thus, $\mathcal{D}_8 = \mathcal{D}_7 \backslash \{5\} = \{6,7\}$, $\mathcal{D}_9 = \mathcal{D}_8 \backslash \{6\} = \{7\}$, and $\mathcal{D}_{10} = \mathcal{D}_9 \setminus \{7\} = \{\}$. It implies that the recursive maximum computation can be used for phases 8–10.*

Recall that $u_7 = v_5 \oplus v_6 \oplus v_{10}$, $u_8 = v_5$, $u_9 = v_6$, and $u_{10} = v_7$. Here $q = 3$, so, to compute $M_0[b], b \in \mathbb{F}_2$, we start from $M_3[(v_5 \oplus v_6 \oplus v_{10}).v_5.v_6.v_7] = R_y(v_0^{10})$, for $v_0^{10} \in \mathcal{Z}_7$. One step of (25) *results in computing $M_2[(v_5 \oplus v_6 \oplus v_{10}).v_5.v_6]$, which are equal to $\max_{v_7 \in \mathbb{F}_2} R_0(v_0^{10})$ for some fixed v_5, v_6, v_{10}. Note that the term $\max_{v_7 \in \mathbb{F}_2} R_0(v_0^{10})$ is a part of in WP* (17) *for phase 9. After computing $M_0[b]$ one immediately obtain $\mathbf{S}_{1,7} = M_0[0] - M_0[1]$. At the next phase, using value $\widehat{u}_7$, provided by the decoder, we have $\mathbf{S}_{1,8} = M_1[\widehat{u}_7.0] - M_1[\widehat{u}_7.1]$.*

C. Reusing of maximums from the previous phase

Expression (17) requires computing of two maximums over the path scores $R_y(v_0^{h_{\phi}})$ for $v_0^{h_{\phi}} \in \mathcal{Z}_{\phi}^{(b)}$, $b \in \mathbb{F}_2$. It is possible to reduce the complexity of this step. Namely, consider the case $h_{\phi+1} = h_{\phi} + 1$, which implies that $\mathcal{D}_{\phi+1} = \mathcal{D}_{\phi}$. According to (9), the path score

$$R_y(v_0^{h_{\phi}+1}) = R_y(v_0^{h_{\phi}}) + \tau(S_t^{(h_{\phi}+1)}(v_0^{h_{\phi}}, y_0^{l-1}), v_{h_{\phi}+1}).$$

Recall that $\tau(S,c)$ function is zero for one of $c \in \mathbb{F}_2$, thus, the path score $R_y(v_0^{h_{\phi}+1}) = R_y(v_0^{h_{\phi}})$ for some $v_{h_{\phi}+1}$.

Suppose the decoder makes an estimate of the symbol $\widehat{u}_{\phi}$. Let $\bar{v}_0^{h_{\phi}} = \arg\max_{v_0^{h_{\phi}} \in \mathcal{Z}_{\phi}^{(\widehat{u}_{\phi})}} R_y(v_0^{h_{\phi}})$. Let $b \in \mathbb{F}_2$ be such that $\tau(S_t^{(h_{\phi}+1)}(\bar{v}_0^{h_{\phi}}, y_0^{l-1}), b) = 0$. Using (13) for $v_{h_{\phi}+1}$ with $u_{\phi+1} = b$ we can obtain such value $a \in \mathbb{F}_2$ so that vector $\bar{v}_0^{h_{\phi}}.a \in \mathcal{Z}_{\phi+1}^{(b)}$. Therefore, $R_y(\bar{v}_0^{h_{\phi}}.a) = R_y(\bar{v}_0^{h_{\phi}})$ and $\bar{v}_0^{h_{\phi}}.a = \arg\max_{v_0^{h_{\phi}+1} \in \mathcal{Z}_{\phi+1}^{(b)}} R_y(v_0^{h_{\phi}+1})$. It means that by tracking the value of b we can directly obtain $\max_{v_0^{h_{\phi}+1} \in \mathcal{Z}_{\phi+1}^{(b)}} R_y(v_0^{h_{\phi}+1})$ in (17). It remains to compute only $\max_{v_0^{h_{\phi}+1} \in \mathcal{Z}_{\phi+1}^{(b \oplus 1)}} R_y(v_0^{h_{\phi}+1})$.

This approach can be employed in the case of reduced decoding window at the next phases.

VI. Efficient processing of some kernels

In this section we briefly describe the window processing of some polarization kernels with good polarization properties and small size of decoding windows. We carefully compute the arithmetic complexity, and point out all simplifications used. Observe that the proposed algorithm involves addition and comparison operations only.

A. Overall processing algorithm

Algorithm 3: Improved window processing algorithm for phase ϕ of $2^t \times 2^t$ kernel K

Step 1 : If $h_{\phi} > h_{\phi-1} + 1$, then use Corollary 1 to obtain $R_y(v_0^{h_{\phi}-1}), v_0^{h_{\phi}-1} \in \mathcal{Z}_{\phi}$.

Step 2 : Compute $S_t^{(h_{\phi})}(v_0^{h_{\phi}-1}, y_0^{l-1}), v_0^{h_{\phi}-1} \in \mathcal{Z}_{\phi}$ using CSE via Alg. 2. Reuse the intermediate LLRs from the previous phases if possible (see Section IV-D).

Step 3 : Compute $R_y(v_0^{h_{\phi}})$ for $v_0^{h_{\phi}} \in \mathcal{Z}_{\phi}$ with $2^{|\mathcal{D}_{\phi}|}$ operations (see Section III-C).

Step 4 : Compute $\mathcal{R}_b = \max_{v_0^{h_{\phi}} \in \mathcal{Z}_{\phi}^{(b)}} R_y(v_0^{h_{\phi}})$ for $b \in [2]$ with $2^{|\mathcal{D}_{\phi}|+1} - 2$ operations. Some modifications are possible:

a) If $h_{\phi} = h_{\phi-1} + 1$, then reuse computations from previous phase to obtain $\mathcal{R}_b$ and compute only $\mathcal{R}_{b \oplus 1}$ with $2^{|\mathcal{D}_{\phi}|} - 1$ operations (see Section V-C).

b) If $h_{\phi} = h_{\phi+1} = \cdots = h_{\phi+q}$, then compute $\mathcal{R}_b$ as $M_0[b]$ using (25) with $2^{|\mathcal{D}_{\phi}|+1} - 2$ operations.

Step 5 : $\mathbf{S}_{1,\phi} \leftarrow \mathcal{R}_0 - \mathcal{R}_1$.

We consider the processing of a $2^t \times 2^t$ polarization kernel K. Alg. 3 illustrates all stages of WP of phase ϕ with improvements proposed in the previous sections. Observe that the transition matrix T, equation (13), and CSE pairs (see Section IV-B) should be computed offline.

Observe that Alg. 3 can be skipped for some phases ϕ. If $\mathcal{D}_{\phi} = \emptyset$, then WP reduces to Arikan SC decoding, i.e. $\mathbf{S}_{1,\phi} = S_t^{h_{\phi}}$. In the case $h_{\phi-p} = \cdots = h_{\phi} = \cdots = h_{\phi+q}$ we assume that line 4b of Alg. 3 is performed at phase $h_{\phi-p}$, hence $\mathbf{S}_{1,\phi} = M_p[\widehat{u}_{\phi-p}^{\phi-1}0] - M_p[\widehat{u}_{\phi-p}^{\phi-1}1]$ (Section V-B).

The memory requirements for this algorithm are following. The total number of CSE stored is given by $C_s = \sum_{\lambda=0}^{t} \max_{\phi \in [l]} |X_{\lambda}^{(h_{\phi})}|$ values. We also need to store the path scores, which requires $C_r = \max_{\phi \in [l]} 2^{|\mathcal{D}_{\phi}|+1}$ values in the worst case. If recursive maximum computation (Section V-B) is applicable, then we additionally need $C_m =$

TABLE II: Complexity of Alg. 3 for K_{16} kernel

ϕ	h_ϕ	$\lvert\mathcal{D}_\phi\rvert$	Complexity of Alg. 3 steps								Total complexity
			1	2				3	4	5	
				X_1	X_2	X_3	X_4				
5	8	3	8	16	8	8	8	8	10	1	67
6	9	3	-	-	-	-	8	8	7 (4a)	1	24
7	10	3	-	-	-	16	8	8	14 (4b)	1	47

$\max_{\phi\in[l]:h_\phi=\cdots=h_{\phi+q}}(2^{q+1}-2)$ LLR entries. Therefore, the overall memory requirements are $C_s+C_r+C_m$ LLR entries.

B. K_{16} kernel

At phases 0–4, 11–15, $\mathcal{D}_\phi=\emptyset$, thus, WP is equal to Arikan SC decoding. The processing complexity for these phases is given in Table I. Observe that all intermediate LLRs for phases 11–15 can be retrieved from CSE LLRs computed at the previous phases.

For phases 8–10 one can use the recursive maximum computation, as shown in Example 5.

Table II demonstrates the complexity of all Alg. 3 steps for phases 5–7, including CSE sizes. The complexity figures in this table take into account reusing of CSE. We also indicate the type of simplification implemented at step 4.

At step 1 of phase 5, the path scores $R_y(v_0^7)$ for $v_0^7\in\mathcal{Z}_5$ can be computed with 8 operations (see Section V-A). Moreover, using the properties of FHT, one can find $\arg\max_{v_0^7\in\mathcal{Z}_5} R_y(v_0^7)$ with 3 operations and obtain one of $\mathcal{R}_b$. The remaining $\mathcal{R}_{b\oplus1}$ is computed with 7 operations.

The total processing complexity of K_{16} is 181 operations, including 95 additions and 86 comparisons.

C. K_{32} kernel

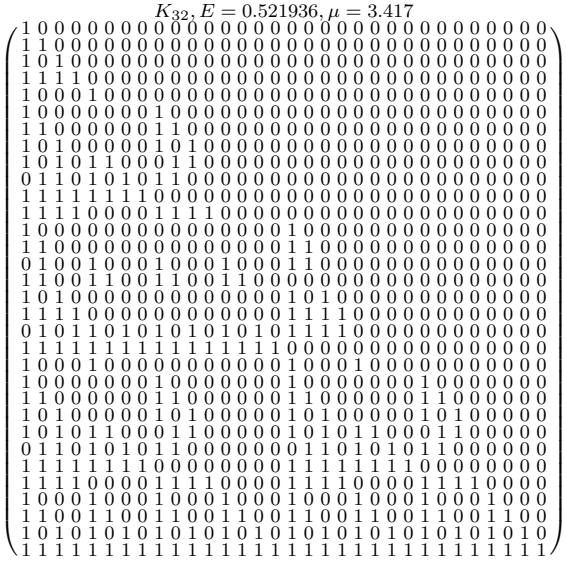


Fig. 3: 32×32 kernel with improved polarization

We present a new 32×32 polarization kernel K_{32}, shown in Fig. 3. This kernel has rate of polarization $E(K_{32})=0.521936$ and scaling exponent $\mu(K_{32})=3.417$. We constructed this kernel via the algorithm presented in [26]. Table III demonstrates the transition matrix, decoding windows and processing complexity for each phase.

TABLE III: Transition matrix of K_{32} kernel

ϕ	u_ϕ	$\mathcal{D}_\phi$	Cost	ϕ	u_ϕ	$\mathcal{D}_\phi$	Cost
0	v_0	$\{\}$	31	16	v_{18}	$\{14,15\}$	16
1	v_1	$\{\}$	1	17	$v_{14}\oplus v_{19}$	$\{14,15\}$	15
2	v_2	$\{\}$	3	18	v_{14}	$\{15\}$	1
3	v_3	$\{\}$	1	19	v_{15}	$\{\}$	1
4	v_4	$\{\}$	7	20	v_{20}	$\{\}$	7
5	v_8	$\{5,6,7\}$	67	21	v_{24}	$\{21,22,23\}$	67
6	$v_5\oplus v_6\oplus v_9$	$\{5,6,7\}$	24	22	$v_{21}\oplus v_{22}\oplus v_{25}$	$\{21,22,23\}$	24
7	$v_5\oplus v_{10}$	$\{5,6,7\}$	47	23	$v_{21}\oplus v_{26}$	$\{21,22,23\}$	47
8	v_5	$\{6,7\}$	1	24	v_{21}	$\{22,23\}$	1
9	v_6	$\{7\}$	1	25	v_{22}	$\{23\}$	1
10	v_7	$\{\}$	1	26	v_{23}	$\{\}$	1
11	v_{11}	$\{\}$	1	27	v_{27}	$\{\}$	1
12	v_{16}	$\{12,13,14,15\}$	127	28	v_{28}	$\{\}$	7
13	$v_{12}\oplus v_{17}$	$\{12,13,14,15\}$	63	29	v_{29}	$\{\}$	1
14	v_{12}	$\{13,14,15\}$	1	30	v_{30}	$\{\}$	3
15	v_{13}	$\{14,15\}$	1	31	v_{31}	$\{\}$	1

TABLE IV: Complexity of Algorithm 3 for K_{32} kernel

ϕ	h_ϕ	$\lvert\mathcal{D}_\phi\rvert$	Complexity of Algorithm 3 steps									Total complexity
			1	2					3	4	5	
				X_1	X_2	X_3	X_4	X_5				
12	16	4	15	32	16	8	8	16	16	15	1	127
13	17	4	-	-	-	-	-	16	16	30 (4b)	1	63
16	18	2	-	-	-	-	4	4	4	3 (4a)	1	16
17	19	2	-	-	-	-	-	4	4	6 (4b)	1	15

At phases 0–4, 11, 20, 27–31 $\mathcal{D}_\phi=\emptyset$, the processing complexity is equal to that of Arikan SC decoding. Surprisingly, phases 5-10 and 21-26 of K_{32} have absolutely the same structure and processing complexity as phases 5-10 of K_{16} (see Section VI-B). The only difference is in the expressions for v_h and in the specific CSE pairs. However, the sizes of CSE sets are the same.

Table IV demonstrates the complexity of all steps of Alg. 3 for phases 12–13 and 16–17, including the sizes of CSE. At phase 12, we have

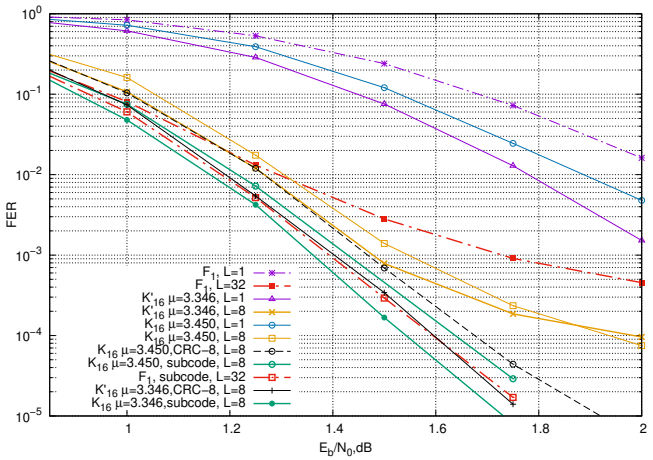
$$R_y(v_0^{15})=R_y(v_0^{12})+\sum_{\beta=0}^{3}\tau(S_3^{(3)}[\beta],c_\beta), \tag{26}$$

where $c_0^3=v_{12}^{15}F_2$ (see Corollary 1). The structure of $\mathcal{D}_{12}$ implies that $c_0^3\in\mathbb{F}_2^4$. The equation (26) requires 4 values $S_3^{(3)}[\beta]$. After that, using the properties of τ function, the last part of (26) can be computed with 11 operations, while $R_y(v_0^{12})$ can be omitted. Moreover, since $c_0^3=v_{12}^{15}F_2$, there is $\bar{c}_0^3$ such that $\sum_{\beta=0}^{3}\tau(S_4^{(3)}[\beta],\bar{c}_\beta)=0$, thus, $\arg\max_{v_0^{15}\in\mathcal{Z}_{15}} R_y(v_0^{12})$ can be immediately obtained with 0 operations. Thus, the total processing complexity for K_{32} is 571 operations, including 297 additions and 274 comparisons.

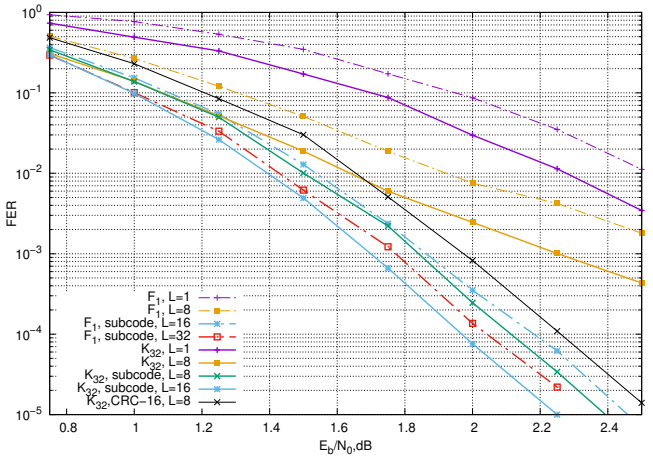
VII. Numeric Results

The performance of all codes was investigated for the case of AWGN channel with BPSK modulation. The sets of frozen symbols were obtained by the method [33].

Observe that the processing techniques described in the previous sections can be also used to implement SCL decoder for polar codes with the considered kernels, using a straightforward generalization of the algorithm and data structures presented in [2]. The SCL algorithm was implemented using

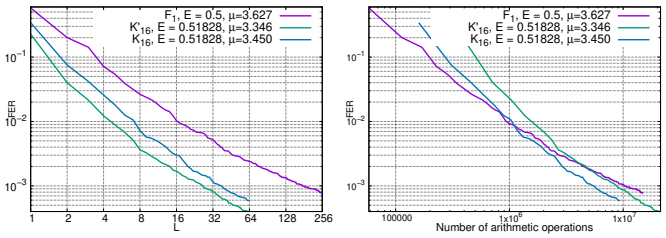


(a) $(4096, 2048)$ polar codes

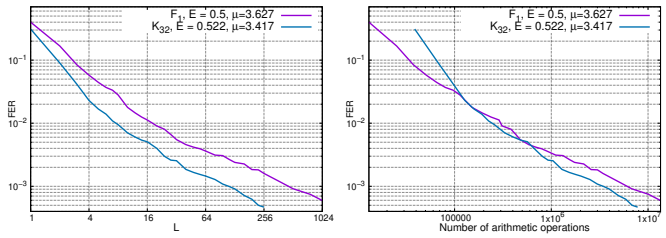


(b) $(1024, 512)$ polar codes

Fig. 4: Performance of polar codes with the proposed kernels



(a) Performance, K_{16}, K'_{16} (b) Complexity, K_{16}, K'_{16}



(c) Performance, K_{32} (d) Complexity, K_{32}

Fig. 5: SCL decoding of polar subcodes with large kernels

the randomized order statistic algorithm for the selection of the paths to be killed at each phase, which has the complexity $O(L)$, where L is a list size.

A. Comparison with Arikan kernel

We constructed $(4096, 2048)$ and $(1024, 512)$ polar codes with kernels K'_{16}, K_{16} and K_{32} respectively. Fig. 4a illustrates the performance of $(4096, 2048)$ plain polar codes, polar codes with CRC and polar subcodes (PSCs) [4]. The same polar code constructions were used for $(1024, 512)$ polar codes, whose performance is demonstrated in Fig. 4b. It can be seen that the codes based on kernels K'_{16}, K_{16} and K_{32} with improved polarization rate ($E(K'_{16}) = E(K_{16}) = 0.51828$, $E(K_{32}) = 0.521936$) provide significant performance gain compared with polar codes with Arikan kernel F_1. Observe also that randomized PSCs provide better performance compared with polar codes with CRC. PSCs with kernels K'_{16}, K_{16} under SCL with $L = 8$ have almost the same performance as PSCs with F_1 kernel under SCL with $L = 32$. Observe that the codes based on kernels with lower scaling exponent exhibit better performance. In the case of PSCs with kernel K_{32} their performance under SCL with $L = 8$ is slightly better than the performance of PSC with F_1 kernel under SCL with $L = 16$.

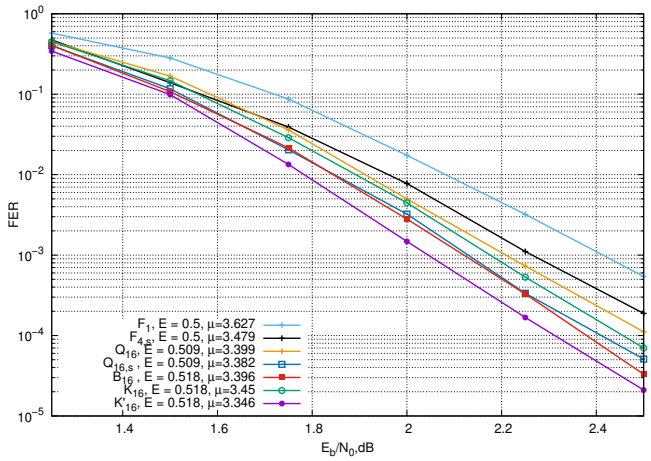
Fig. 5a presents the simulation results for $(4096, 2048)$ PSCs with different kernels under SCL with varied L at $E_b/N_0 = 1.25$ dB. It can be seen that the kernels with rate of polarization 0.51828 require significantly lower list size L to achieve the same performance as the code with F_1 kernel, which has the rate of polarization 0.5. Moreover, this gap grows with L. This is due to the improved rate of polarization, which results in a smaller number of unfrozen imperfectly polarized bit subchannels. The size of the list needed to correct possible errors in these subchannels grows exponentially with their number (at least for the genie-aided decoder considered in [34]). On the other hand, lower scaling exponent enables better performance with the same list size L, but the slope of the curve remains the same for both kernels K'_{16}, K_{16}.

Fig. 5c presents the simulation results for $(1024, 512)$ PSC with different kernels under SCL with varied L at $E_b/N_0 = 1.5$ dB. Similarly to the case of 16×16 kernels, 32×32 kernel K_{32} with improved polarization properties ($E(K_{32}) = 0.521936, \mu(K_{32}) = 3.417$) provides significant performance gain compared with polar subcode with F_1.

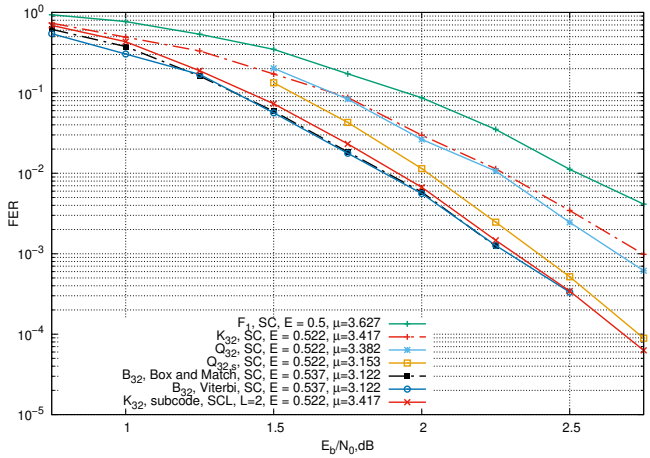
Fig. 5b presents the performance of SCL decoding of $(4096, 2048)$ PSCs with K'_{16} and K_{16} kernels in terms of the decoding complexity. Observe that the PSC based on kernel K_{16} can provide better performance with the same decoding complexity for FER $\leq 7 \cdot 10^{-3}$ ($L \geq 8$ for K_{16} and $L \geq 22$ for K'_{16}). This is due to the higher slope of the corresponding curve in Fig. 5a, which eventually enables one to compensate the relatively high complexity of the LLR computation algorithm presented in Section VI-B.

Unfortunately, K'_{16} kernel with lower scaling exponent has greater processing complexity than K_{16}, so that its curve intersects the one for the F_1 kernel only at FER$= 2 \cdot 10^{-3}$.

Fig. 5d presents the results of SCL decoding of $(1024, 512)$ PSC with K_{32} in terms of the decoding complexity. Similarly to K_{16}, the PSC with K_{32} can provide better performance



(a) $(4096, 2048)$ codes, 16×16 kernels



(b) $(1024, 512)$ codes, 32×32 kernels

Fig. 6: Performance of polar codes with various polarization kernels

with the same decoding complexity. Starting from FER $\leq 2 \cdot 10^{-2}$ PSC with K_{32} ($L \geq 4$) maintain approximately the same performance-complexity tradeoff as PSC with F_1 ($L \geq 9$). Observe that K_{32} enables one to reduce required list size by more than two times. Starting with FER $\leq 4 \cdot 10^{-3}$ PSC with K_{32} ($L \geq 20$) provides better performance with the same decoding complexity compared with PSC with F_1 ($L \geq 48$).

B. Comparison with other large kernels

TABLE V: SC decoding complexity of (4096,2048) polar codes with various kernels

$F_{4,s}$	Q_{16}	K_{16}	$Q_{16,s}$	B_{16}	K'_{16}
$1.7 \cdot 10^5$	$2.1 \cdot 10^5$	$1.4 \cdot 10^5$	$3.9 \cdot 10^5$	$1.3 \cdot 10^6$	$3.4 \cdot 10^5$

In this section we compare the performance and complexity of the recently proposed kernels with kernels K'_{16}, K_{16} and K_{32}. We consider the convolutional polar kernels Q_{16} and Q_{32} [11] processed by the algorithm introduced in [35], as well as sorted convolutional polar kernels [11] $Q_{16,s}$ and $Q_{32,s}$. Authors in [9] minimized the complexity of trellis-based Viterbi processing algorithm for 16×16 and 32×32 BCH kernels B_{16} and B_{32}. We also include the results for the Box and Match based approximate processing algorithm [10] applied to B_{32} and the results for sorted Arikan kernel $F_{4,s}$ [13] processed by the WP algorithm.

TABLE VI: Decoding complexity of (1024,512) polar codes with various kernels

SC					SCL, L=2
K_{32}	Q_{32}	$Q_{32,s}$	B_{32}, approx. [10]	B_{32}, Viterbi	K_{32}
$3.6 \cdot 10^4$	$6.2 \cdot 10^4$	$1.1 \cdot 10^6$	$\approx 1.5 \cdot 10^5$	$1.9 \cdot 10^7$	$6.9 \cdot 10^4$

Fig. 6a compares the performance of $(4096, 2048)$ polar codes with different kernels, Table V depicts the decoding complexity of these codes. It can be observed that polar code with K_{16} outperforms polar codes with Q_{16} and $F_{4,s}$, while having lower processing complexity and better polarization properties (shown in the plot). Polar code with K'_{16} kernel also outperforms polar codes with $Q_{16,s}$ and B_{16}, while having lower processing complexity and better polarization properties.

Fig. 6b compares the performance of $(1024, 512)$ polar codes with different kernels, while Table VI depicts the decoding complexity of these codes. It can be seen that due to better polarization properties (shown in the plot) these kernels also provide better SC decoding performance. However, by employing SCL decoding of with $L = 2$, K_{32} kernel can compete with $Q_{32,s}$ and B_{32} in terms of performance, while having significantly lower arithmetic complexity.

VIII. Conclusions

In this paper a reduced complexity decoding algorithm for polar codes with $2^t \times 2^t$ polarization kernels was proposed. Application of this approach to kernels of size 16 and 32 was considered. The algorithm computes the kernel input symbols LLRs via the ones for the Arikan kernel, and exploits the structure of recursive calculations induced by the Arikan kernel to identify and reuse the values of some common subexpressions. It was shown that in the case of SCL decoding with sufficiently large list size, the proposed approach results in lower decoding complexity compared with the case of polar (sub)codes with Arikan kernel with the same performance.

Acknowledgments

We thank Fariba Abbasi Aghdam Meinagh for many comments and stimulating discussions. We also thank the editor and the reviewers for their beneficial comments.

References

[1] E. Arıkan, "Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels," *IEEE Transactions on Information Theory*, vol. 55, no. 7, pp. 3051–3073, July 2009.

[2] I. Tal and A. Vardy, "List decoding of polar codes," *IEEE Transactions On Information Theory*, vol. 61, no. 5, pp. 2213–2226, May 2015.

[3] P. Trifonov and V. Miloslavskaya, "Polar subcodes," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 2, pp. 254–266, February 2016.

[4] P. Trifonov and G. Trofimiuk, "A randomized construction of polar subcodes," in *Proceedings of IEEE International Symposium on Information Theory*. Aachen, Germany: IEEE, 2017, pp. 1863–1867.
[5] T. Wang, D. Qu, and T. Jiang, "Parity-check-concatenated polar codes," *IEEE Communications Letters*, vol. 20, no. 12, December 2016.
[6] S. B. Korada, E. Sasoglu, and R. Urbanke, "Polar codes: Characterization of exponent, bounds, and constructions," *IEEE Transactions on Information Theory*, vol. 56, no. 12, pp. 6253–6264, December 2010.
[7] A. Fazeli, S. H. Hassani, M. Mondelli, and A. Vardy, "Binary linear codes with optimal scaling: Polar codes with large kernels," in *Proceedings of IEEE Information Theory Workshop*, 2018.
[8] Z. Huang, S. Zhang, F. Zhang, C. Duanmu, F. Zhong, and M. Chen, "Simplified successive cancellation decoding of polar codes with medium-dimensional binary kernels," *IEEE Access*, vol. 6, pp. 26 707–26 717, 2018.
[9] E. Moskovskaya and P. Trifonov, "Design of BCH polarization kernels with reduced processing complexity," *IEEE Communications Letters*, vol. 24, no. 7, pp. 1383–1386, July 2020.
[10] V. Miloslavskaya and P. Trifonov, "Sequential decoding of polar codes with arbitrary binary kernel," in *Proceedings of IEEE Information Theory Workshop*. Hobart, Australia: IEEE, 2014, pp. 377–381.
[11] R. A. Morozov, "Convolutional polar kernels," *IEEE Transactions on Communications*, vol. 68, no. 12, pp. 7352–7361, 2020.
[12] P. Trifonov, "Binary successive cancellation decoding of polar codes with Reed-Solomon kernel," in *Proceedings of IEEE International Symposium on Information Theory*. Honolulu, USA: IEEE, 2014, pp. 2972 – 2976.
[13] S. Buzaglo, A. Fazeli, P. H. Siegel, V. Taranalli, and A. Vardy, "Permuted successive cancellation decoding for polar codes," in *2017 IEEE International Symposium on Information Theory (ISIT)*, 2017, pp. 2618–2622.
[14] V. Bioglio and I. Land, "On the marginalization of polarizing kernels," in *Proceedings of International Symposium on Turbo Codes and Iterative Information Processing*, 2018.
[15] H. Griesser and V. R. Sidorenko, "A posteriory probability decoding of nonsystematically encoded block codes," *Problems of Information Transmission*, vol. 38, no. 3, 2002.
[16] V. Miloslavskaya and P. Trifonov, "Sequential decoding of polar codes," *IEEE Communications Letters*, vol. 18, no. 7, pp. 1127–1130, 2014.
[17] P. Trifonov, "A score function for sequential decoding of polar codes," in *Proceedings of IEEE International Symposium on Information Theory*, Vail, USA, 2018.
[18] A. Balatsoukas-Stimming, M. B. Parizi, and A. Burg, "LLR-based successive cancellation list decoding of polar codes," *IEEE Transactions On Signal Processing*, vol. 63, no. 19, pp. 5165–5179, October 2015.
[19] G. Trofimiuk, N. Iakuba, S. Rets, K. Ivanov, and P. V. Trifonov, "Fast block sequential decoding of polar codes," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 10, pp. 10 988–10 999, October 2020.
[20] N. Presman, O. Shapira, S. Litsyn, T. Etzion, and A. Vardy, "Binary polarization kernels from code decompositions," *IEEE Transactions On Information Theory*, vol. 61, no. 5, May 2015.
[21] H.-P. Lin, S. Lin, and K. A. Abdel-Ghaffar, "Linear and nonlinear binary kernels of polar codes of small dimensions with maximum exponents," *IEEE Transactions On Information Theory*, vol. 61, no. 10, October 2015.
[22] A. Fazeli and A. Vardy, "On the scaling exponent of binary polarization kernels," in *Proceedings of 52nd Annual Allerton Conference on Communication, Control and Computing*, 2014, pp. 797 – 804.
[23] S. H. Hassani, K. Alishahi, and R. Urbanke, "Finite-length scaling for polar codes," *IEEE Transactions On Information Theory*, vol. 60, no. 10, October 2014.
[28] F. Abbasi and E. Viterbo, "Large kernel polar codes with efficient window decoding," *IEEE Transactions On Vehicular Technology*, vol. 69, no. 11, November 2020.
[24] H. Pfister and R. Urbanke, "Near-optimal finite-length scaling for polar codes over large alphabets," in *Proceedings of IEEE International Symposium on Information Theory*, 2016.
[25] A. Fazeli, H. Hassani, M. Mondelli, and A. Vardy, "Binary linear codes with optimal scaling: Polar codes with large kernels," *IEEE Transactions on Information Theory*, 2020.
[26] G. Trofimiuk and P. Trifonov, "Construction of binary polarization kernels for low complexity window processing," in *Proceedings of IEEE Information Theory Workshop*, 2019.
[27] H. Yao, A. Fazeli, and A. Vardy, "Explicit polar codes with small scaling exponent," in *Proceedings of IEEE International Symposium on Information Theory*, 2019.
[29] G. Trofimiuk and P. Trifonov, "Efficient decoding of polar codes with some 16×16 kernels," ArXiV:2001.03921, 2020.
[30] A. Valembois and M. Fossorier, "Box and match techniques applied to soft-decision decoding," *IEEE Transactions on Information Theory*, vol. 50, no. 5, pp. 796–810, 5 2004.
[31] H. T. Moorthy, S. Lin, and T. Kasami, "Soft-decision decoding of binary linear block codes based on an iterative search algorithm," *IEEE Transactions On Information Theory*, vol. 43, no. 3, pp. 1030–1040, May 1997.
[32] Y. Beery and J. Snyders, "Optimal soft decision block decoders based on fast Hadamard transform," *IEEE Transactions on Information Theory*, vol. 32, no. 3, May 1986.
[33] P. Trifonov, "On construction of polar subcodes with large kernels," in *Proceedings of IEEE International Symposium on Information Theory*, Paris, France, 2019.
[34] M. Mondelli, S. H. Hassani, and R. Urbanke, "Scaling exponent of list decoders with applications to polar codes," *IEEE Transactions On Information Theory*, vol. 61, no. 9, September 2015.
[35] R. Morozov, "Efficient list decoding of convolutional polar codes," arXiv:2007.05811, 2020. [Online]. Available: https://arxiv.org/abs/2007.05811

Grigorii Trofimiuk (S'15) was born in Boksitogorsk, Russia in 1994. He received the B.Sc. and M.Sc. degrees from St.Petersburg Polytechnic University in 2016 and 2018, respectively, all in computer science. He is currently working toward the Ph.D. degree at the ITMO University in St.Petersburg, Russia. His research interests include coding theory and its applications in telecommunications.

Peter Trifonov (S'02,M'05) was born in St.Petersburg, USSR in 1980. He received the MSc and PhD (Candidate of Science) degrees from Saint Petersburg Polytechnic University in 2003 and 2005, and Dr.Sc degree from the Institute for Information Transmission Problems in 2018. His research interests include coding theory and its applications in telecommunications and storage systems. Currently he is a professor at the ITMO University in Saint Petersburg, Russia. He is an editor at IEEE Transaction on Communications.