A Practical Indexing Scheme for Noisy Shuffling Channels Using Cosets of Polar Codes

Javad Haghighat and Tolga M. Duman, *Fellow, IEEE*

***Abstract*—The noisy shuffling channel models the conditions encountered in DNA storage systems, where transmitted data segments experience random permutation and substitution errors. Reliable communication over this channel requires effective indexing and channel coding strategies for segment order restoration and error correction. This paper introduces a concatenated coding approach for communication over the noisy shuffling channel, using Reed-Solomon (RS) codes as outer codes and polar codes as inner codes. A coset-based indexing method, derived from polar codes, is proposed. A joint decoder is designed to detect the permutation pattern and perform polar decoding simultaneously. An approximate analysis of the frame error rate (FER) using random coding is conducted. Additionally, a mapping between the cosets of the polar code and subsets of its frozen bits is established to design cosets achieving lower FERs compared to a commonly used explicit indexing method. Furthermore, a list decoding approach is devised, providing a trade-off between the computational complexity of the joint decoder and its performance.**

***Index Terms*—Noisy shuffling channel, Polar code, Coset, Indexing, DNA storage.**

I. INTRODUCTION

DNA storage systems are receiving significant attention from the research community, thanks to their longevity and their impressive storage density [1]-[7]. The basic idea in DNA storage is to employ a synthesizer that takes information bits as input and maps them to synthetic DNA strands. However, due to technical limitations in current synthesizing technologies, synthetic strands are limited to a few hundreds of nucleotides in length. Therefore, data has to be divided into short segments that are then written on short strands and are stored in a solution known as the DNA pool.

The DNA pool has a fundamental disadvantage compared to other storage environments such as disks and magnetic tapes; that is, DNA pool is not capable of maintaining the order of the stored strands (since the strands are floating in a solution and their physical positions cannot be fixed). Consequently, when the information is being read from the pool, there is no guarantee that the strands are sequenced (read) in the same order as they are synthesized (written). In short, the output of the sequencer is a shuffled version of the strands generated by the synthesizer. Furthermore, substitution errors are likely to occur during the synthesis, during the storage, and while sequencing the strands. For this reason, the end-to-end channel between the original data and the output of the sequencer may be modeled as a noisy shuffling channel [8]. The noisy shuffling channel model may be further modified by noting that the strands are amplified (copied several times) inside the DNA pool, via the Polymerase Chain Reaction (PCR) process. Hence, the sequencing process of a randomly selected subset of the stored strands is more accurately represented by a noisy shuffling-sampling channel model, in which each strand is sampled a random number of times. These channel models and their variations are studied in a number of recent papers including [8]-[18].

Since the ordering of the segments is not maintained by the noisy shuffling channel, a mechanism has to be implemented at the transmitter (i.e., during synthesis) to enable the receiver to restore the correct order. The simplest mechanism is to explicitly assign an index of length $\lceil \log_2 M \rceil$ bits to each segment, where M denotes the number of segments. However, if the indexes are corrupted by channel noise, the order may be lost. This observation motivates several works including [14]-[18] to focus on more robust indexing methods. In [14], each index is appended by a vector referred to as the anchor of that index. Anchors are selected such that the Hamming distance between two arbitrarily selected index-anchor vectors is greater than or equal to a threshold value (by an index-anchor vector, we refer to a vector generated by concatenating an index and its corresponding anchor). Also, each set of anchors belongs to a maximum distance separable (MDS) code. The receiver may take advantage of these properties to first decode the noisy anchors, and then employ the retrieved anchors to restore the transmission order [14]. In [15], indexes are encoded by an error correcting code with a minimum distance of $2K+1$, where K is the maximum number of substitution errors imposed by the channel. This approach enables the receiver to recover the indexes using a minimum distance decoder. In [16] indexes are protected by aid of specially designed short-length and low-rate codes. The codes are defined in $GF(4)$ and have a rate of $\frac{1}{3}$. In [17], transmission over a noisy shuffling-sampling channel is considered and the indexes are designed such that if the distance between two particular segments is small, the distance between their corresponding indexes is large. This property enables the receiver to correctly cluster the channel reads, i.e., to correctly

---

J. Haghighat is with the Department of Electrical and Electronics Engineering, TED University, 06420 Ankara, Turkey. T. M. Duman is with the Department of Electrical and Electronics Engineering, Bilkent University, 06800 Ankara, Turkey (e-mail: javad.haghighat@tedu.edu.tr, duman@ee.bilkent.edu.tr)

This work was funded by the European Union through the ERC Advanced Grant 101054904: TRANCIDS. Views and opinions expressed are however those of the authors only and do not necessarily reflect those of the European Union or the European Research Council Executive Agency. Neither the European Union nor the granting authority can be held responsible for them.

Part of this paper is presented in IEEE GLOBECOM 2023 [1].

identify the noisy segments that correspond to the same original segment. In [18], a concatenated coding scheme is proposed for transmission over noisy shuffling-sampling channels, where the inner code is partitioned into disjoint sub-codes and each data segment is encoded using a separate sub-code. The decoder may identify the position of each noisy segment, given the decoder is capable of finding the sub-code by which that noisy segment is encoded. Therefore, [18] does not employ explicit indexes to maintain the order of the segments; instead, the scheme relies on the capability of the decoder to correctly identify the sub-codes and restore the order.

In this paper, we focus on the scenario where a sequence of information bits is sliced into a finite number of short-length segments, and the segments are transmitted over a noisy shuffling channel. We implement a concatenated Reed-Solomon and polar coding scheme, where each data segment is encoded by a separate coset of a polar code. We design a joint decoder that decodes the received noisy segments and also detects the permutation pattern and restores their order. Then, we establish a one-to-one mapping between the cosets of the polar code and subsets of its frozen bits. By employing this mapping, we argue that explicit indexing is a special case of the proposed coset-based indexing; hence, the designed joint decoder may also be employed to decode explicitly indexed segments. Furthermore, we introduce a process to select the cosets such that the probability of error in detecting the permutation pattern is reduced. We refer to this selection process as the coset design process. We demonstrate that our designed cosets outperform explicit indexing, as well as randomly selected ones. We also propose an approach to reduce the complexity of the joint decoder. This approach is applicable to the designed cosets, but not to randomly selected cosets. Therefore, the designed cosets are more suitable choices for trading off performance for complexity. It is worth mentioning that the idea of encoding different segments by different cosets of the polar code is similar to the approach presented in [18], where the inner code is partitioned into disjoint sub-codes and each segment is encoded by a separate sub-code. However, [18] does not propose a practical encoding or decoding scheme to implement this idea. Instead, it assumes that a proper decoder implementation exists; then, performs asymptotic analysis, where the number of segments and the segment length grow arbitrarily large.

Our contributions in this paper are as follows:

- We propose an implicit indexing approach based on the cosets of the inner code, and derive bounds on its performance through a random coding analysis.
- We derive analytical approximations for the FER of a concatenated RS-polar coding scheme employed to communicate over a noisy shuffling channel.
- We design a decoder that jointly detects the permutation pattern imposed by the channel and performs polar decoding.
- We define a one-to-one mapping between the cosets of the polar code and subsets of its frozen bits. Then, we employ this mapping to search for cosets that offer FERs lower than those achieved by explicit indexing, as well as a benchmark scheme which is based on randomly selected cosets. We refer to this search process as the *coset design* process.
- In order to reduce the computational complexity of our originally proposed joint decoder, referred to by the *full decoder*, we propose an additional low-complexity decoder that works by a one-shot list decoding. We compare the performance and the complexity of this low-complexity decoder with those of the full decoder.

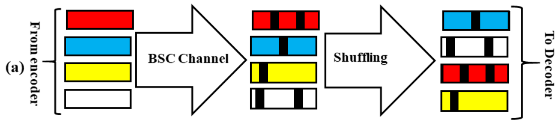


Figure 1. Schematic of the noisy shuffling channel model.

The paper is organized as follows. The system model is given in Section II. In Section III we present our proposed concatenated coding and joint decoding scheme. Section IV provides a performance analysis for the proposed scheme through an approximation derived on the FER. Section V is dedicated to the proposed coset design process and the low-complexity decoding approach. Section VI includes numerical results and discussions. Finally, Section VII concludes the paper.

II. SYSTEM MODEL

Schematics of the employed noisy permutation channel model is shown in Fig. 1. We assume that M codewords of length N bits, which are generated by a channel code, are input to the noisy shuffling channel. We refer to these codewords as the segments. First, the M segments are passed through a noisy channel. Although in practice, DNA storage schemes may be affected by several types of errors, including substitution, deletion and insertion errors, different works on current DNA storage technologies confirm that substitution errors are dominant [3]-[7]. For this reason, in this paper, we focus on noisy channels with substitution errors. Specifically, we consider a binary symmetric channel (BSC) with crossover probability α. The segments at the output of the BSC are shuffled before being processed by the channel decoder.

For convenience, a list of frequently used symbols employed in this paper, along with their definition is given in Table I.

III. PROPOSED CODING SCHEME

Due to the shuffling process, the order of segments is not preserved at the output of a noisy shuffling channel. Therefore, the segments must be indexed before transmission to enable restoration of their order at the receiver. Inspired by the work of [18], in this section, we propose a concatenated encoding scheme along with an implicit indexing method that employs different cosets of a polar code to encode distinct segments. Then, we design a joint decoder that decodes the segments and restores their order. The output of this decoder is delivered to the outer decoder that corrects the remaining substitution errors.

Table I
List of frequently used symbols and their definitions.

Symbol	Definition
α	BSC crossover probability
N_o	Codeword length of RS code
K_o	Data length of RS code
q	Number of bits per symbol of RS codewords
N	Codeword length of polar code
K	Data length of polar code
$\boldsymbol{\rho}$	Channel-reliability sequence
M	Number of cosets
$\mathcal{C}^m$	The m-th coset of the polar code
$\boldsymbol{s}^m$	The m-th segment of data
$\boldsymbol{\pi}$	Permutation pattern
L	List size of the list decoder
$\boldsymbol{\mathcal{L}}(\boldsymbol{a};\epsilon)$	Artificial likelihood ratios
n_F	Number of frozen bits

The block diagram of the proposed scheme is shown in Fig. 2. Let q, K_o be positive integers. The binary input sequence, $\boldsymbol{t} = (t_0, \ldots, t_{qK_o-1})$consisting of qK_o bits, is partitioned into q-bit symbols, and then encoded using an (N_o, K_o) RS code, where $N_o = 2^q - 1$. The RS codeword length is N_o symbols, or equivalently qN_o bits. This codeword is then zero-padded by $KM - qN_o$ bits to form a binary vector $\boldsymbol{s}$ with length KM bits, where $K = \left\lceil \frac{qN_o}{M} \right\rceil$ and $\lceil . \rceil$ denotes the ceiling function. Then, $\boldsymbol{s}$ is partitioned into M segments of length K bits, denoted by $\boldsymbol{s}^0$ through $\boldsymbol{s}^{M-1}$. These M segments are encoded by M cosets of a polar code.

Encoding the M segments by M different cosets of the polar code provides the segments with *implicit indexes*; i.e., for each received noisy segment, if the decoder is able to detect the coset with which that segment is encoded, then it is able to detect the position of that segment in $\boldsymbol{s}$. Therefore, we refer to the above encoding approach by coset-based indexing, and explain it in detail in Section III-A.

A. Proposed Coset-based Indexing

In order to formulate the proposed coset-based indexing method, let us denote the (N, K) polar code by $\mathcal{C}^0$ and let $\mathcal{B}_K$ denote the set of all possible binary message vectors with length K. In order to encode a message vector $\boldsymbol{b} = (b_0, \ldots, b_{K-1})$ by the polar code, first a vector $\boldsymbol{u}(\boldsymbol{b}) = (u_0, \ldots, u_{N-1})$ is generated where:

$$u_{\rho_j} = \begin{cases} b_j; & j \in \{0, \ldots, K-1\}, \\ 0; & \text{otherwise.} \end{cases} \tag{1}$$

The vector $\boldsymbol{\rho} = (\rho_0, \ldots, \rho_{N-1})$ denotes the channel-reliability sequence [19] which, in this paper, is assumed to be sorted from the most reliable bit position to the least reliable bit position. Entries $u_{\rho_K}, u_{\rho_{K+1}}, \ldots, u_{\rho_{N-1}}$ are called frozen bits which are taken as zero. Then, the polar codeword corresponding to the message vector $\boldsymbol{b}$ is given by

$$\boldsymbol{x}^0(\boldsymbol{b}) = \boldsymbol{u}(\boldsymbol{b}) \times \boldsymbol{G}_N, \tag{2}$$

where $\boldsymbol{G}_N$ denotes the polar transform [19]. In the sequel, we drop the subscript, N, and denote the polar transform by $\boldsymbol{G}$.

Now, let $\mathcal{C}^1, \ldots, \mathcal{C}^{M-1}$ be $M-1$ cosets of $\mathcal{C}^0$. The members of $\mathcal{C}^m$ are expressed as:

$$\boldsymbol{x}^m(\boldsymbol{b}) = \boldsymbol{x}^0(\boldsymbol{b}) \oplus \boldsymbol{e}^m, \tag{3}$$

where $\boldsymbol{e}^m$ denotes the coset leader of $\mathcal{C}^m$. Clearly, $\boldsymbol{e}^0 = \boldsymbol{0}_N$, where $\boldsymbol{0}_N$ is an all-zero vector of length N.

In the proposed scheme, segment $\boldsymbol{s}^m$ is encoded as $\boldsymbol{x}^m(\boldsymbol{s}^m)$, for $m \in \{0, \ldots, M-1\}$, where $\boldsymbol{x}^m(.)$ is defined according to (3). Subsequently, as shown in Fig. 2, vectors $\boldsymbol{r}^0$ through $\boldsymbol{r}^{M-1}$ are received at the output of the noisy shuffling channel, where

$$\boldsymbol{r}^{\pi_m} = \boldsymbol{x}^m(\boldsymbol{s}^m) \oplus \boldsymbol{z}^m, \tag{4}$$

where $\boldsymbol{z}^m$ is a BSC noise vector, and $\boldsymbol{\pi} = (\pi_0, \ldots, \pi_{M-1})$ is the permutation pattern imposed by the channel.

B. Joint Decoder Design

In order to recover the transmitted sequence, $\boldsymbol{t}$, the receiver needs to accomplish two tasks, namely (i) detecting the permutation pattern, $\boldsymbol{\pi}$, and (ii) correcting the substitution errors imposed by the BSC noise vectors, $\boldsymbol{z}^m$. In the following, we design a joint decoder that simultaneously detects the permutation pattern and performs polar decoding. The output of this joint decoder is then delivered to the outer RS decoder that corrects the remaining substitution errors.

In order to jointly decode and sort the received segments, we propose to generate a *Coset-Segment Distance* (CSD) matrix at the decoder, where we define an *ideal* CSD matrix[1] as an $M \times M$ matrix $\boldsymbol{\Gamma} = [\gamma_{m,m'}]$ with entries:

$$\gamma_{m,m'} = \min_{\boldsymbol{b} \in \mathcal{B}_K} \mathrm{d}_H\left(\boldsymbol{x}^m(\boldsymbol{b}), \boldsymbol{r}^{m'}\right) \tag{5}$$

where $\mathrm{d}_H(.,.)$ denotes the Hamming distance function and $\gamma_{m,m'}$ represents the minimum Hamming distance between $\boldsymbol{r}^{m'}$ and members of $\mathcal{C}^m$. Note that in order to generate $\boldsymbol{\Gamma}$, one needs to run M^2 minimum distance decoders. Let us denote the M^2 corresponding decoded information sequences as:

$$\hat{\boldsymbol{s}}^{m,m'} = \underset{\boldsymbol{b} \in \mathcal{B}_K}{\operatorname{argmin}}\, d_H\left(\boldsymbol{x}^m(\boldsymbol{b}), \boldsymbol{r}^{m'}\right) \tag{6}$$

and their corresponding codewords in $\mathcal{C}^0$ as:

$$\hat{\boldsymbol{x}}^{m,m'} = \boldsymbol{u}\left(\hat{\boldsymbol{s}}^{m,m'}\right) \times \boldsymbol{G}. \tag{7}$$

Hence, the closest vector in $\mathcal{C}^m$ to $\boldsymbol{r}^{m'}$ is found as:

$$\boldsymbol{x}^m\left(\hat{\boldsymbol{s}}^{m,m'}\right) = \hat{\boldsymbol{x}}^{m,m'} \oplus \boldsymbol{e}^m. \tag{8}$$

---

[1] This ideal matrix is defined based on minimum distance decoding which is not feasible in general cases. A practical approximation of this matrix will be given based on list decoding in the sequel.

Let $\mathcal{P}_M$ denote the set of all $M!$ permutations of $0, 1, \dots, M-1$. Given a permutation pattern $\boldsymbol{\pi} \in \mathcal{P}_M$, the received vectors are reordered as $\boldsymbol{r}^{\pi_0}, \boldsymbol{r}^{\pi_1}, \dots, \boldsymbol{r}^{\pi_{M-1}}$, where $\boldsymbol{r}^{\pi_m}$ corresponds to the transmitted vector $\boldsymbol{x}^m(\boldsymbol{s}^m)$. Therefore, $\boldsymbol{r}^{\pi_m}$ needs to be decoded in the m-th coset, which gives the decoded vector $\hat{\boldsymbol{x}}^{m,\pi_m} \oplus \boldsymbol{e}^m$. Given this observation, we propose to detect the permutation pattern as follows:

$$\hat{\boldsymbol{\pi}} = \underset{\boldsymbol{\pi} \in \mathcal{P}_M}{\operatorname{argmin}} \sum_{m=0}^{M-1} \mathrm{d}_H\left(\hat{\boldsymbol{x}}^{m,\pi_m} \oplus \boldsymbol{e}^m, \boldsymbol{r}^{\pi_m}\right), \tag{9}$$

i.e., as the permutation pattern that minimizes the cumulative Hamming distance between the re-sorted received vectors and their closest neighbors in the cosets. Note that from (5), (6), (8), we have:

$$\mathrm{d}_H\left(\hat{\boldsymbol{x}}^{m,\pi_m} \oplus \boldsymbol{e}^m, \boldsymbol{r}^{\pi_m}\right) = \gamma_{m,\pi_m}, \tag{10}$$

therefore:

$$\hat{\boldsymbol{\pi}} = \underset{\boldsymbol{\pi} \in \mathcal{P}_M}{\operatorname{argmin}} \sum_{m=0}^{M-1} \gamma_{m,\pi_m}. \tag{11}$$

After detecting the permutation pattern as $\hat{\boldsymbol{\pi}}$, the vector $\hat{\boldsymbol{s}} = \left(\hat{\boldsymbol{s}}^{0,\hat{\pi}_0}, \dots, \hat{\boldsymbol{s}}^{M-1,\hat{\pi}_{M-1}}\right)$ is delivered to the RS decoder which generates the estimated bit stream $\hat{\boldsymbol{t}}$.

Since minimum distance decoding has a computational complexity that exponentially grows with K, the ideal CSD matrix defined in (5) cannot be generated with reasonable complexity, except for very small values of K. However, one may employ a list decoder of the polar code to generate an approximate CSD matrix. In order to construct such an approximate matrix, first note that by employing (3), (5) can be rewritten as:

$$\gamma_{m,m'} = \min_{\boldsymbol{b} \in \mathcal{B}_K} \mathrm{d}_H\left(\boldsymbol{x}^0(\boldsymbol{b}), \boldsymbol{e}^m \oplus \boldsymbol{r}^{m'}\right) \tag{12}$$

Therefore, $\gamma_{m,m'}$ may be found by running a decoder which finds a codeword in the polar code, $\mathcal{C}^0$, with a minimum distance from $\boldsymbol{e}^m \oplus \boldsymbol{r}^{m'}$ compared to any other codeword in $\mathcal{C}^0$. Now, we may replace such an optimal minimum distance decoder by a (suboptimal) list decoder by taking the following approach. For a binary vector $\boldsymbol{a} = (a_0, \dots, a_{N-1})$, and a small positive number, ϵ, define an artificial reliability vector $\boldsymbol{\mathcal{L}}(\boldsymbol{a}; \epsilon) = (\mathcal{L}_0, \dots, \mathcal{L}_{N-1})$ such that:

$$\mathcal{L}_j = \begin{cases} \epsilon; & a_j = 0, \\ 1-\epsilon; & a_j = 1. \end{cases} \tag{13}$$

The vector $\boldsymbol{\mathcal{L}}(\boldsymbol{a}; \epsilon)$ models the reliability values (likelihood ratios) at the output of a low-noise channel (i.e., a channel that adds a small amount of noise to its input vector, $\boldsymbol{a}$). Therefore, if one gives the vector $\boldsymbol{\mathcal{L}}\left(\boldsymbol{e}^m \oplus \boldsymbol{r}^{m'}; \epsilon\right)$ as an input to a list decoder with a list size of L, one expects that codewords with small Hamming distances from $\boldsymbol{e}^m \oplus \boldsymbol{r}^{m'}$ appear among the L candidate codewords produced by that list decoder. Let $\mathcal{X}^{m,m'}$ denote the set of the L codewords produced by the list decoder, and let $\mathcal{S}^{m,m'}$ denote the set of information sequences corresponding to these codewords. Then, we may define:

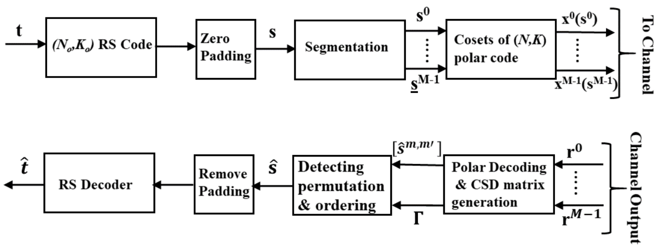


Figure 2. Block diagram of the proposed scheme.

$$\begin{cases} \tilde{\boldsymbol{s}}^{m,m'} = \underset{\boldsymbol{b} \in \mathcal{S}^{m,m'}}{\operatorname{argmin}} \mathrm{d}_H\left(\boldsymbol{x}^0(\boldsymbol{b}), \boldsymbol{e}^m \oplus \boldsymbol{r}^{m'}\right) \\ \tilde{\boldsymbol{x}}^{m,m'} = \boldsymbol{u}\left(\tilde{\boldsymbol{s}}^{m,m'}\right) \times \boldsymbol{G} \\ \tilde{\gamma}_{m,m'} = \mathrm{d}_H\left(\tilde{\boldsymbol{x}}^{m,m'}, \boldsymbol{e}^m \oplus \boldsymbol{r}^{m'}\right) \end{cases} \tag{14}$$

and employ them as approximations to $\hat{\boldsymbol{s}}^{m,m'}$, $\hat{\boldsymbol{x}}^{m,m'}$, and $\gamma_{m,m'}$, respectively.

Remark 1. We note that one may take an alternative indexing approach, which we refer to as *explicit indexing*, where an index of length $\lceil \log_2 M \rceil$ bits is appended to each segment and the indexed segment is encoded by an $(N, K + \lceil \log_2 M \rceil)$ polar code. In Section V-A we show that this explicit indexing is equivalent to a special case of the proposed coset-based indexing; hence, the above joint decoder is applicable to explicit indexing as well.

IV. PERFORMANCE ANALYSIS OF THE PROPOSED SCHEME

In this section, we analyze the FER of the proposed scheme for a scenario where the outer decoder is a minimum distance decoder. Since the RS code has a minimum distance of $N_o - K_o + 1$ symbols, a minimum distance decoder definitely corrects up to $\frac{N_o - K_o}{2}$ symbol errors in every RS codeword (selecting $N_o - K_o$ even). According to Fig. 2, if zero-padding is neglected, the sequence of symbols delivered to the RS decoder is $\hat{\mathbf{s}}$. Therefore, if a minimum distance decoder is applied to decode the RS codeword, the probability of a frame error event is less than or equal to the probability of observing more than $\frac{N_o - K_o}{2}$ symbol errors in $\hat{\mathbf{s}}$.

Let us assume that a permutation pattern $\boldsymbol{\pi} \in \mathcal{P}_M$ is imposed by the channel and a permutation pattern $\hat{\boldsymbol{\pi}}$ is detected at the receiver. Define a correct detection event, $\overline{\mathcal{E}_d}$, as the event that $\hat{\boldsymbol{\pi}} = \boldsymbol{\pi}$; i.e., $\hat{\pi}_m = \pi_m$ for all m; and denote a detection error event by $\mathcal{E}_d$. When $\overline{\mathcal{E}_d}$ occurs, $\boldsymbol{r}^{\pi_m}$ is assigned to its *matched* coset, i.e., $\mathcal{C}^m$, for all m. In this case, the bit error rate (BER) at the output of the inner decoder, which is defined as $E\left(\frac{\mathrm{d}_H(\boldsymbol{s}, \hat{\boldsymbol{s}})}{KM}\right)$, is equal to the BER of the polar code ($E(.)$ denotes the expectation operator). Therefore, since each symbol of an RS codeword consists of q consecutive bits, the symbol error rate at the input of the outer decoder can be evaluated as:

$$p_s = 1 - \{1 - p_b\}^q , \tag{15}$$

where p_b denotes the BER of the polar code.

Note that (15) is applicable when $\overline{\mathcal{E}_d}$ has occurred, in which case $\boldsymbol{r}^{\pi_m}$ is always assigned to its matched coset, $\mathcal{C}^m$. On the other hand, when $\mathcal{E}_d$ occurs, there exist values of m for which $\boldsymbol{r}^{\pi_m}$ is assigned to a *mismatched* coset, e.g., coset $\mathcal{C}^{m''}$ where $m'' \neq m$. This mismatched assignment is expected to increase the BER at the input of the outer decoder; therefore, it is likely that the outer decoder cannot correct these excessive bit errors and makes a frame decoding error; i.e., maps $\hat{\boldsymbol{s}}$ to a $\hat{\boldsymbol{t}} \neq \boldsymbol{t}$. Following the above discussion, we assume that given $\mathcal{E}_d$, a frame error event always occurs. This is equivalent to bounding the FER by 1 when $\mathcal{E}_d$ occurs. Therefore, the FER achieved by a minimum distance decoder can be upper bounded as:

$$\begin{aligned} P_e \le \;& p_d + (1-p_d) \\ & \times \left(1 - \sum_{j=0}^{\frac{N_o-K_o}{2}} \binom{N_o}{j} p_s^j \left(1-p_s\right)^{N_o-j}\right) \end{aligned} \tag{16}$$

where $p_d = Pr\left(\mathcal{E}_d\right)$ and p_s is expressed in (15) in terms of p_b, the BER of the polar code.

To evaluate the right hand side of (16), one needs the values (or estimates) of p_b and p_d. To estimate p_b, one may either employ existing bounds on the error probability of polar codes (e.g., the bounds provided in [20]-[22]), or employ Monte-Carlo simulations. Since the M cosets of the polar code are correlated, it is not clear how to derive analytical approximations for p_d in case of polar codes. Therefore, we obtain a random coding bound as follows. We replace the M cosets by M independently generated random codes. Then, we derive an upper bound on p_d for such a scenario, and take the derived bound as an approximation for p_d when the cosets belong to a polar code.

Before proceeding, let us prove the following lemma which is later employed to derive the bound on p_d.

Lemma 2. *Assume that the channel has imposed a permutation pattern* $\boldsymbol{\pi}$ *and the CSD matrix* $\boldsymbol{\Gamma} = [\gamma_{m,m'}]$ *is generated at the decoder. Define* $\overline{\mathcal{E}}_{\pi_m}$ *as the event that* $\gamma_{m,\pi_m} < \gamma_{m'',\pi_m}$ *for all* $m'' \neq m$*, i.e., the event that the smallest entry of column numbered* π_m *appears in the* m*-th row and all other entries in that column are greater than this minimum value. If* $\overline{\mathcal{E}}_{\pi_m}$*'s jointly occur for all* m*, then* $\overline{\mathcal{E}}_d$ *occurs, i.e., then the permutation pattern detected by (11) is equal to* $\boldsymbol{\pi}$.

Before proving Lemma 2, let us clarify its statement by providing an example. Assume that $M = 3$ cosets are employed, the permutation pattern $\boldsymbol{\pi} = (1, 0, 2)$ is imposed by the channel, and the following CSD matrix is generated by the joint decoder:

$$\Gamma = \begin{bmatrix} \gamma_{0,0} & \gamma_{0,1} & \gamma_{0,2} \\ \gamma_{1,0} & \gamma_{1,1} & \gamma_{1,2} \\ \gamma_{2,0} & \gamma_{2,1} & \gamma_{2,2} \end{bmatrix} = \begin{bmatrix} 5 & 3 & 6 \\ 2 & 7 & 8 \\ 8 & 5 & 2 \end{bmatrix} \tag{17}$$

The minimum entries in columns numbered $0, 1, 2$ of Γ are $\gamma_{1,0}, \gamma_{0,1}, \gamma_{2,2}$. Since $\boldsymbol{\pi} = (1, 0, 2)$, we have $\pi_0 = 1, \pi_1 = 0, \pi_2 = 2$; hence, we may rewrite the minimum entries as $\gamma_{1,\pi_1}, \gamma_{0,\pi_0}, \gamma_{2,\pi_2}$. Therefore, $\overline{\mathcal{E}}_{\pi_1}, \overline{\mathcal{E}}_{\pi_0}, \overline{\mathcal{E}}_{\pi_2}$ jointly occur. Also, since the summation on the right hand side of (11) is minimized by picking $\gamma_{0,1}, \gamma_{1,0}, \gamma_{2,2}$, it is clear that $\hat{\boldsymbol{\pi}} = (1, 0, 2)$ is detected; i.e., $\hat{\boldsymbol{\pi}} = \boldsymbol{\pi}$, hence $\overline{\mathcal{E}}_d$ occurs.

***Proof of Lemma 2*:** For clarity of notation, let us rewrite (11) as:

$$\hat{\boldsymbol{\pi}} = \underset{\boldsymbol{\sigma} \in \mathcal{P}_M}{\text{argmin}} \sum_{m=0}^{M-1} \gamma_{m,\sigma_m} . \tag{18}$$

Define $\boldsymbol{\mu} = \boldsymbol{\sigma}^{-1} \circ \boldsymbol{\pi}$ where -1 denotes inversion of a function and $\circ$ denotes composition of two functions. It is clear that $\boldsymbol{\mu} \in \mathcal{P}_M$ and also $\sigma_{\mu_m} = \pi_m$ for all $m \in \{0, 1, \ldots, M-1\}$. Hence, we may write:

$$\hat{\boldsymbol{\pi}} = \underset{\boldsymbol{\mu} \in \mathcal{P}_M}{\text{argmin}} \sum_{m=0}^{M-1} \gamma_{\mu_m,\pi_m} . \tag{19}$$

Since $\boldsymbol{\mu} \in \mathcal{P}_M$, the summation on the right hand side of (19) is constrained to pick only one entry from each row of $\boldsymbol{\Gamma}$. By removing this constraint, we may derive a lower bound for the summation as follows:

$$\sum_{m=0}^{M-1} \gamma_{\mu_m,\pi_m} \ge \sum_{m=0}^{M-1} \min_{m'' \in \{0,\ldots,M-1\}} \gamma_{m'',\pi_m} . \tag{20}$$

Since it is assumed in the lemma that $\overline{\mathcal{E}}_{\pi_m}$'s jointly occur, we have:

$$\min_{m'' \in \{0,\ldots,M-1\}} \gamma_{m'',\pi_m} = \gamma_{m,\pi_m} \tag{21}$$

for all m.

Replacing (21) in (20) gives $\sum_{m=0}^{M-1} \gamma_{\mu_m,\pi_m} \ge \sum_{m=0}^{M-1} \gamma_{m,\pi_m}$, where the lower bound is achieved by choosing $\mu_m = m$ for all m. Since according to the definition of $\overline{\mathcal{E}}_{\pi_m}$, all entries γ_{m'',π_m}, $m'' \neq m$, are strictly larger than γ_{m,π_m}, then for all other choices of $\boldsymbol{\mu}$, $\sum_{m=0}^{M-1} \gamma_{\mu_m,\pi_m}$ is strictly larger than the lower bound; i.e., the optimal choice of $\boldsymbol{\mu}$ is unique and is given by $\mu_m = m$ for all m. Since $\sigma_{\mu_m} = \pi_m$, we conclude that the optimal choice of $\boldsymbol{\sigma}$ is unique and is given by $\sigma_m = \pi_m$ for all m. In other words, the summation on the right hand side of (18) is minimized for $\boldsymbol{\sigma} = \boldsymbol{\pi}$, which gives the detected permutation pattern as $\hat{\boldsymbol{\pi}} = \boldsymbol{\pi}$. Therefore, $\overline{\mathcal{E}}_d$ occurs and the proof is complete.

Using Lemma 2, we observe that joint occurrence of $\overline{\mathcal{E}}_{\pi_m}$'s is a sufficient condition for occurrence of $\overline{\mathcal{E}}_d$. Therefore, we may write:

$$Pr\left(\overline{\mathcal{E}}_d\right) \ge Pr\left(\bigcap_{m=0}^{M-1} \overline{\mathcal{E}}_{\pi_m}\right) \tag{22}$$

Now, we may employ (22) along with the union bound to find:

$$Pr\left(\mathcal{E}_d\right) \le Pr\left(\bigcup_{m=0}^{M-1} \mathcal{E}_{\pi_m}\right) \le \sum_{m=0}^{M-1} Pr\left(\mathcal{E}_{\pi_m}\right) \tag{23}$$

where $\mathcal{E}_{\pi_m}$ is the complement of $\overline{\mathcal{E}}_{\pi_m}$. In the following, we derive an upper bound on $Pr\left(\mathcal{E}_{\pi_m}\right)$ when the cosets are random codes; then, we replace it in (23) to find an upper bound on $Pr\left(\mathcal{E}_d\right)$. For this purpose, we follow the approach taken in [23] to find the exact average error probability of a random code ensemble over a BSC.

A. Random Coding Analysis

Let $\mathcal{C} = \left\{\mathcal{C}^0, \ldots, \mathcal{C}^{M-1}\right\}$ be a set of (N, K) random codes where each code has 2^K codewords and each codeword is drawn uniformly at random (with replacement) from $\mathcal{B}_N$. Recall that $\boldsymbol{r}^{\pi_m} = \boldsymbol{x}^m\left(\boldsymbol{s}^m\right) \oplus \boldsymbol{z}^m$ where $\boldsymbol{s}^m \in \mathcal{B}_K$ is the m-th data segment. From (5) it is clear that:

$$\gamma_{m,\pi_m} \leq \mathrm{d}_H\left(\boldsymbol{x}^m\left(\boldsymbol{s}^m\right), \boldsymbol{r}^{\pi_m}\right) = \mathrm{W}_H\left(\boldsymbol{z}^m\right) \tag{24}$$

where $\mathrm{W}_H\left(\boldsymbol{z}^m\right)$ denotes the Hamming weight of $\boldsymbol{z}^m$. Therefore, given $\mathrm{W}_H\left(\boldsymbol{z}^m\right) = w$, a sufficient condition for $\overline{\mathcal{E}}_{\pi_m}$ to occur, is that $\mathrm{d}_H\left(\boldsymbol{x}^{m''}\left(\boldsymbol{b}\right), \boldsymbol{r}^{\pi_m}\right) > w$, for all $m'' \neq m$ and all $\boldsymbol{b} \in \mathcal{B}_K$. This condition is satisfied if for all codewords $\boldsymbol{x'} \in \mathcal{C} - \mathcal{C}^m$ it is true that $d_H\left(\boldsymbol{x'}, \boldsymbol{r}^{\pi_m}\right) > w$. Using the above discussion and by noting that codewords $\boldsymbol{x'}$ are selected uniformly at random from $\mathcal{B}_N$, we may write:

$$Pr\left(\overline{\mathcal{E}}_{\pi_m} | W_H\left(\boldsymbol{z}^m\right) = w\right) \geq \left(Pr\left(d_H\left(\boldsymbol{x'}, \boldsymbol{r}^{\pi_m}\right) > w\right)\right)^{|\mathcal{C}-\mathcal{C}^m|} \tag{25}$$

Now, let us assume that $\boldsymbol{x}^m\left(\boldsymbol{s}^m\right) = \boldsymbol{x}_0$ for a fixed $\boldsymbol{x}_0 \in \mathcal{B}_N$. Then, $\boldsymbol{r}^{\pi_m} = \boldsymbol{x}_0 \oplus \boldsymbol{z}^m$; and we may write:

$$\mathrm{d}_H\left(\boldsymbol{x'}, \boldsymbol{r}^{\pi_m}\right) = W_H\left(\boldsymbol{x'} \oplus \boldsymbol{x}_0 \oplus \boldsymbol{z}^m\right) \tag{26}$$

Let us define:

$$\mathcal{G}\left(\boldsymbol{z}^m\right) = \left\{\boldsymbol{b} \in \mathcal{B}_N \text{ s.t.} \, \mathrm{W}_H\left(\boldsymbol{b} \oplus \boldsymbol{z}^m\right) > w\right\}, \tag{27}$$

by noting that $\boldsymbol{x'} \oplus \boldsymbol{x}_0$ is uniformly distributed over $\mathcal{B}_N$, from (26) and (27) we will have:

$$Pr\left(d_H\left(\boldsymbol{x'}, \boldsymbol{r}^{\pi_m}\right) > w\right) = Pr\left(\boldsymbol{x'} \oplus \boldsymbol{x}_0 \in \mathcal{G}\left(\boldsymbol{z}^m\right)\right) = \frac{|\mathcal{G}\left(\boldsymbol{z}^m\right)|}{|\boldsymbol{\mathcal{B}}_N|} \tag{28}$$

To find $|\mathcal{G}\left(\boldsymbol{z}^m\right)|$, we show that there exists a one-to-one mapping between $|\mathcal{G}\left(\boldsymbol{z}^m\right)|$ and the set of vectors with a Hamming weight greater than w, defined as:

$$\mathcal{A}_w = \left\{\boldsymbol{b} \in \mathcal{B}_N \; s.t. \, \mathrm{W}_H\left(\boldsymbol{b}\right) > w\right\} \tag{29}$$

To show the one-to-one mapping, we note that:

(i) If $\boldsymbol{b} \in \mathcal{G}\left(\boldsymbol{z}^m\right)$, then from (27) it is known that $\mathrm{W}_H\left(\boldsymbol{b} \oplus \boldsymbol{z}^m\right) > w$; i.e., $\boldsymbol{b} \oplus \boldsymbol{z}^m \in \mathcal{A}_w$.

(ii) If $\boldsymbol{b} \in \mathcal{A}_w$, $\mathrm{W}_H\left(\boldsymbol{b}\right) = \mathrm{W}_H\left(\boldsymbol{z}^m \oplus \left(\boldsymbol{b} \oplus \boldsymbol{z}^m\right)\right) > w$; i.e., $\boldsymbol{b} \oplus \boldsymbol{z}^m \in \mathcal{G}\left(\boldsymbol{z}^m\right)$.

Using this one-to-one mapping, we write:

$$|\mathcal{G}\left(\boldsymbol{z}^m\right)| = |\mathcal{A}_w| = 2^N - \sum_{j=0}^{w} \binom{N}{j} \tag{30}$$

By replacing (30) in (28) and noting that $|\mathcal{B}_N| = 2^N$ we derive $Pr\left(d_H\left(\boldsymbol{x'}, \boldsymbol{r}^{\pi_m}\right) > w\right)$; then, we plug it in (25), and also replace $|\mathcal{C} - \mathcal{C}^m| = (M-1)\, 2^K$, to find:

$$Pr\left(\overline{\mathcal{E}}_{\pi_m} | \mathrm{W}_H\left(\boldsymbol{z}^m\right) = w\right) \geq \left(1 - 2^{-N} \sum_{j=0}^{w} \binom{N}{j}\right)^{(M-1)2^K} \tag{31}$$

Now, we may employ (31) along with the law of total probability and the distribution function of the Hamming weight of the BSC noise vector, i.e.,

$$Pr\left(W_H\left(\boldsymbol{z}^m\right) = w\right) = \binom{N}{w} \alpha^w \left(1-\alpha\right)^{N-w} \tag{32}$$

to obtain:

$$\begin{aligned} Pr\left(\overline{\mathcal{E}}_{\pi_m}\right) \; &\geq \textstyle\sum_{w=0}^{N} \binom{N}{w} \alpha^w \left(1-\alpha\right)^{N-w} \\ &\times \left(1 - 2^{-N} \textstyle\sum_{j=0}^{w} \binom{N}{j}\right)^{(M-1)2^K} \end{aligned} \tag{33}$$

To simplify the calculations, we employ the inequality:

$$\left(1-y\right)^n \geq \left(1-ny\right) \times u_{-1}\left(1-ny\right), \; -1 < y < 1, \, n \geq 1 \tag{34}$$

where $u_{-1}\left(.\right)$ is the unit step function. Using (33), (34), and by noting that $Pr\left(\mathcal{E}_{\pi_m}\right) = 1 - Pr\left(\overline{\mathcal{E}}_{\pi_m}\right)$, we obtain $Pr\left(\mathcal{E}_{\pi_m}\right) \leq 1 - \zeta\left(N, K, M, \alpha\right)$ where:

$$\begin{aligned} \zeta\left(N, K, M, \alpha\right) \; &= \textstyle\sum_{w=0}^{N} \binom{N}{w} \alpha^w \left(1-\alpha\right)^{N-w} \\ &\times \left(1 - 2^{-(N-K)} \textstyle\sum_{j=0}^{w} \binom{N}{j}\right)^{(M-1)} \\ &\times u_{-1}\left(1 - 2^{-(N-K)} \textstyle\sum_{j=0}^{w} \binom{N}{j}\right) \end{aligned} \tag{35}$$

Note that $\zeta\left(N, K, M, \alpha\right)$ does not depend on specific choices of $\boldsymbol{x}_0$ and m. By plugging this result into (23) and by noting that $p_d = Pr\left(\mathcal{E}_d\right)$, we conclude:

$$p_d \leq M \times \left(1 - \zeta\left(N, K, M, \alpha\right)\right), \tag{36}$$

The right hand side of (36) may be replaced in (16) as an approximation for the FER as follows:

$$\begin{aligned} P_e \; &\approx M\left(1 - \zeta\left(N, K, M, \alpha\right)\right) \\ &+ \left(1 - M + M \times \zeta\left(N, K, M, \alpha\right)\right) \\ &\times \left(1 - \textstyle\sum_{j=0}^{\frac{N_o-K_o}{2}} \binom{N_o}{j} p_s^j \left(1-p_s\right)^{N_o-j}\right) \end{aligned} \tag{37}$$

Note that one cannot claim that (37) gives an upper bound on the FER, since (36) is derived by random coding analysis; whereas, in the proposed scheme, polar codes and their corresponding cosets are employed. Nonetheless, this analysis provides insights on the performance of a concatenated coding scheme that employs the proposed coset-based indexing method.

V. COSET DESIGN AND COMPLEXITY REDUCTION

In this section, we show that there is a one-to-one mapping between the 2^{N-K}cosets of an (N, K) polar code and the

2^{N-K} subsets of its frozen bits. Using this one-to-one mapping, we define explicit indexing as a specific type of coset-based indexing. Then, we define the *minimum pairwise distance* (MPD) between two cosets and propose a search method to find a set of cosets that maximize the MPD. We refer to this search method as the coset design process. We show that our coset design approach has the potential to reduce the FER compared to random coset selection.

We also study solutions to reduce the computational complexity of the joint decoding method presented in Section III-B. As discussed, M^2 list decoders are required to generate the CSD matrix; furthermore, $M!$ permutations must be examined before selecting the most likely permutation pattern. This imposes a high computational complexity for large values of M. In this section, by employing the one-to-one mapping between the subsets of frozen bits and the cosets, we introduce a method that generates each column of the CSD matrix by running a single list decoder, such that the whole matrix can be generated by running M list decoders. Also, we propose a suboptimal iterative method to infer a permutation pattern from the CSD matrix, without the need to examine all the $M!$ patterns.

A. Relationship between Cosets and Frozen Bits

In (1) we defined a bit reliability pattern, $\boldsymbol{\rho} = (\rho_0, \ldots, \rho_{N-1})$, in descending reliability order, such that the bits numbered ρ_j, $K \leq j \leq N-1$, denote the frozen bits that are set to zero. To establish a one-to-one mapping between cosets of the polar code and the subsets of its frozen bits, let us define a set of 2^{N-K} vectors denoted by $\tilde{\boldsymbol{u}}(\boldsymbol{f}) = (\tilde{u}_0, \ldots, \tilde{u}_{N-1})$ for $\boldsymbol{f} \in \mathcal{B}_{N-K}$ such that:

$$\tilde{u}_{\rho_j} = \begin{cases} 0; & 0 \leq j \leq K-1, \\ f_{j-K}; & K \leq j \leq N-1, \end{cases} \tag{38}$$

i.e., $\tilde{\boldsymbol{u}}(\boldsymbol{f})$ is a vector for which all information bits are set to zero; whereas the frozen bits are set according to $\boldsymbol{f}$. It is clear that $\tilde{\boldsymbol{u}}(\boldsymbol{f}) \times \boldsymbol{G}$ is not a codeword except for the case $\boldsymbol{f} = \boldsymbol{0}_{N-K}$ which gives the all-zero codeword. Otherwise, $\tilde{\boldsymbol{u}}(\boldsymbol{f}) \times G$ belongs to a coset of the polar code other than $\mathcal{C}^0$. Also, for each pair $\boldsymbol{f}, \boldsymbol{f'} \in \mathcal{B}_{N-K}$, it is clear from (38) that $\tilde{\boldsymbol{u}}(\boldsymbol{f}) \oplus \tilde{\boldsymbol{u}}(\boldsymbol{f}') = \tilde{\boldsymbol{u}}(\boldsymbol{f} \oplus \boldsymbol{f}')$. Hence, we may write:

$$(\tilde{\boldsymbol{u}}(\boldsymbol{f}) \times \boldsymbol{G}) \oplus (\tilde{\boldsymbol{u}}(\boldsymbol{f}') \times \boldsymbol{G}) = \tilde{\boldsymbol{u}}(\boldsymbol{f} \oplus \boldsymbol{f}') \times \boldsymbol{G}, \tag{39}$$

for all $\boldsymbol{f}, \boldsymbol{f'} \in \mathcal{B}_{N-K}$. Therefore, $\tilde{\boldsymbol{u}}(\boldsymbol{f}) \times \boldsymbol{G}$ and $\tilde{\boldsymbol{u}}(\boldsymbol{f'}) \times \boldsymbol{G}$ belong to the same coset, if and only if $\boldsymbol{f} = \boldsymbol{f'}$ (since that is the only case where their addition leads to a codeword). In other words, for every $\boldsymbol{f} \neq \boldsymbol{f'}$, the vectors $\tilde{\boldsymbol{u}}(\boldsymbol{f}) \times \boldsymbol{G}$ and $\tilde{\boldsymbol{u}}(\boldsymbol{f}') \times \boldsymbol{G}$ belong to two different cosets. Since $|\mathcal{B}_{N-K}| = 2^{N-K}$ which is equal to the total number of cosets, we conclude that there is a one-to-one mapping between the 2^{N-K} vectors $\boldsymbol{f} \in \mathcal{B}_{N-K}$ and the 2^{N-K} cosets of the polar code. In summary, corresponding to each coset $\mathcal{C}^m$ with coset leader $\boldsymbol{e}^m$, there exists a unique vector in $\mathcal{B}_{N-K}$ denoted by $\boldsymbol{f}^m$, for which $\tilde{\boldsymbol{u}}(\boldsymbol{f}^m) \times \boldsymbol{G} \in \mathcal{C}^m$ and subsequently:

$$(\boldsymbol{e}^m \oplus \tilde{\boldsymbol{u}}(\boldsymbol{f}^m) \times \boldsymbol{G}) \in \mathcal{C}^0 \tag{40}$$

where $\mathcal{C}^0$ denotes the polar code. Hence, the set of M cosets $\mathcal{C}^0, \ldots, \mathcal{C}^{M-1}$ can be represented by their M corresponding vectors denoted by $\boldsymbol{f}^0, \ldots, \boldsymbol{f}^{M-1}$.

The above observation enables us to categorize explicit indexing as a special case of coset-based indexing, for which $\boldsymbol{f}^m$ is generated by setting the first $\lceil \log_2 M \rceil$ bits according to the binary representation of the decimal number m (and setting the remaining bits to zero). For example, for $M = 4$, explicit indexing is equivalent to a coset-based indexing with:

$$\begin{cases} \boldsymbol{f}^0 = (0, 0, 0, ..., 0) \\ \boldsymbol{f}^1 = (0, 1, 0, ..., 0) \\ \boldsymbol{f}^2 = (1, 0, 0, ..., 0) \\ \boldsymbol{f}^3 = (1, 1, 0, ..., 0) \end{cases} \tag{41}$$

Note that since $\tilde{\boldsymbol{u}}(\boldsymbol{f}^m) \times \boldsymbol{G} \in \mathcal{C}^m$, its summation with any codeword such as $\boldsymbol{x}^0(\boldsymbol{b})$ is also in $\mathcal{C}^m$. Therefore, by noting that $\boldsymbol{x}^0(\boldsymbol{b}) = \boldsymbol{u}(\boldsymbol{b}) \times \boldsymbol{G}$ (Eq. (2)), we may write:

$$\mathcal{C}^m = \{(\boldsymbol{u}(\boldsymbol{b}) \oplus \tilde{\boldsymbol{u}}(\boldsymbol{f}^m)) \times \boldsymbol{G} \text{ for } \boldsymbol{b} \in \mathcal{B}_K\} \tag{42}$$

Remark 3. From (1) and (38) and by noting that for explicit indexing only the first $\lceil \log_2 M \rceil$ bits can be set to non-zero values, we conclude that for explicit indexing case, $\boldsymbol{u}(\boldsymbol{b}) \oplus \tilde{\boldsymbol{u}}(\boldsymbol{f}^m)$ is a vector for which bits numbered $\rho_{K+\lceil \log_2 M \rceil}, \ldots, \rho_{N-1}$ are forced to zero. Hence, $(\boldsymbol{u}(\boldsymbol{b}) \oplus \tilde{\boldsymbol{u}}(\boldsymbol{f}^m)) \times \boldsymbol{G}$ can be regarded as a codeword of an $(N, K + \lceil \log_2 M \rceil)$ polar code, which we refer to as the *supercode*. From this conclusion and using (42), we observe that for the explicit indexing scheme, all cosets $\mathcal{C}^m$ are subsets of this supercode. This is also clear from the nature of explicit indexing, where the K information bits are appended by $\lceil \log_2 M \rceil$ index bits and then are encoded by an $(N, K + \lceil \log_2 M \rceil)$ polar code (see Remark 1).

Also, we note that a random selection of cosets is equivalent to selecting a set of vectors $\boldsymbol{f}^0$ through $\boldsymbol{f}^{M-1}$ uniformly at random out of the set $\mathcal{B}_{N-K}$. This observation is helpful when we study benchmark schemes consisting of randomly selected cosets, in Section VI.

1) Finding the Coset Leaders:

Although we have established the one-to-one mapping between $\mathcal{C}^m$'s and $\boldsymbol{f}^m$'s, it remains to explain how to find the coset leader corresponding to $\boldsymbol{f}^m$, which we denoted by $\boldsymbol{e}^m$. When K is small such that generating all 2^K codewords, $\boldsymbol{x}^0(\boldsymbol{b})$, is feasible; then $\boldsymbol{e}^m$ is found as:

$$\boldsymbol{e}^m = \boldsymbol{x}^0(\boldsymbol{b}') \oplus \tilde{\boldsymbol{u}}(\boldsymbol{f}^m) \times G \tag{43}$$

where:

$$\boldsymbol{b}' = \underset{\boldsymbol{b} \in \mathcal{B}_K}{\operatorname{argmin}}\, \mathrm{d}_H(\boldsymbol{x}^0(\boldsymbol{b}), \tilde{\boldsymbol{u}}(\boldsymbol{f}^m) \times G) \tag{44}$$

i.e., we find the codeword with the minimum Hamming distance from $\tilde{\boldsymbol{u}}(\boldsymbol{f}^m) \times \boldsymbol{G}$ and then add it to $\tilde{\boldsymbol{u}}(\boldsymbol{f}^m) \times \boldsymbol{G}$ to find the coset leader.

When K is such that generating all 2^K codewords is not feasible, we take an iterative approach to search for the coset leader. For this, we initialize $\boldsymbol{e}^m$ as $\tilde{\boldsymbol{u}}\left(\boldsymbol{f}^m\right)\times G$, then we repeat the following steps.

(i) We generate the artificial reliability vector $\boldsymbol{L}\left(\boldsymbol{e}^m,\epsilon\right)$ defined in (13).

(ii) $\boldsymbol{L}\left(\boldsymbol{e}^m,\epsilon\right)$ is given to a list decoder of the (N,K) polar code which provides a list of L codewords.

(iii) We search in the list for the codeword with a minimum Hamming distance from $\boldsymbol{e}^m$. Let us denote this codeword by $\boldsymbol{x}'$.

(iv) If $\mathrm{d}_H\left(\boldsymbol{e}^m\oplus\boldsymbol{x}'\right)<\mathrm{W}_H\left(\boldsymbol{e}^m\right)$, we update $\boldsymbol{e}^m\leftarrow\boldsymbol{e}^m\oplus\boldsymbol{x}'$ and return to Step (i).

(v) The final value of $\boldsymbol{e}^m$ is declared as the coset leader.

Note that by increasing the list size, L, and choosing a proper value for ϵ, one may improve the accuracy of the above search process.

B. Designing Good Cosets

In this section, we devise a systematic approach to search for sets of cosets that provide small detection error probabilities, denoted by $p_d=Pr\left(\mathcal{E}_d\right)$. In particular, we look for *good* cosets that reduce $Pr\left(\mathcal{E}_d\right)$ compared to a benchmark scheme, where for the benchmark scheme the cosets are randomly selected and are regenerated for transmitting each data sequence, $\boldsymbol{t}$ (but their realizations are known at the decoder). To search for such good cosets, we employ the MPD between the cosets $\mathcal{C}^m$ and $\mathcal{C}^{m'}$ which we define as:

$$\delta_{m,m'}=\min_{\boldsymbol{b},\boldsymbol{b}'\in\mathcal{B}_K}\mathrm{d}_H\left(\boldsymbol{x}^m\left(\boldsymbol{b}\right),\boldsymbol{x}^{m'}\left(\boldsymbol{b}'\right)\right)\tag{45}$$

Let $\Delta=\left[\delta_{m,m'}\right]$ denote the MPD matrix, which is a symmetric matrix with diagonal entries $\delta_{m,m}=0$. We refer to the minimum positive-valued entry in Δ as the minimum MPD (MMPD) and define it as:

$$\check{\delta}=\min_{m,m'\,:\,m'\neq m}\delta_{m,m'}.\tag{46}$$

By noting that $\boldsymbol{x}^m\left(\boldsymbol{b}\right)=\boldsymbol{e}^m\oplus\boldsymbol{x}^0\left(\boldsymbol{b}\right)$, and by straightforward manipulations, we may rewrite (45) as:

$$\mathrm{d}_H\left(\boldsymbol{x}^m\left(\boldsymbol{b}\right),\boldsymbol{x}^{m'}\left(\boldsymbol{b}'\right)\right)=\mathrm{d}_H\left(\boldsymbol{e}^m\oplus\boldsymbol{e}^{m'},\boldsymbol{x}^0\left(\boldsymbol{b}\right)\oplus\boldsymbol{x}^0\left(\boldsymbol{b}'\right)\right)\tag{47}$$

Also, since the polar code is a linear code (when the frozen bits are set to 0), we have $\boldsymbol{x}^0\left(\boldsymbol{b}\right)\oplus\boldsymbol{x}^0\left(\boldsymbol{b}'\right)=\boldsymbol{x}^0\left(\boldsymbol{b}''\right)$ for a $\boldsymbol{b}''\in\mathcal{B}_K$. Therefore:

$$\delta_{m,m'}=\min_{\boldsymbol{b}''\in\mathcal{B}_K}d_H\left(\boldsymbol{e}^m\oplus\boldsymbol{e}^{m'},\boldsymbol{x}^0\left(\boldsymbol{b}''\right)\right)\tag{48}$$

i.e., the MPD of $\mathcal{C}^m$ and $\mathcal{C}^{m'}$ can be found by employing a minimum distance decoder to find the closest codeword in $\mathcal{C}^0$ to the vector $\boldsymbol{e}^m\oplus\boldsymbol{e}^{m'}$.

Recall the definition of the CSD matrix, $\boldsymbol{\Gamma}=\left[\gamma_{m,m'}\right]$, according to (5), and let $\mathcal{E}_{m\to m'}$ denote the event that $\gamma_{m',\pi_m}<\gamma_{m,\pi_m}$; i.e. the event that $\boldsymbol{r}^{\pi_m}$ which is a noisy version of $\boldsymbol{x}^m\left(\boldsymbol{s}^m\right)\in\mathcal{C}^m$, is closer to a vector $\boldsymbol{x}^{m'}\left(\boldsymbol{b}'\right)\in\mathcal{C}^{m'}$ (compared to all vectors in $\mathcal{C}^m$, including $\boldsymbol{x}^m\left(\boldsymbol{s}^m\right)$). It is clear that in the worst-case scenario where $\mathrm{d}_H\left(\boldsymbol{x}^m\left(\boldsymbol{s}^m\right),\boldsymbol{x}^{m'}\left(\boldsymbol{b}'\right)\right)=\delta_{m,m'}$, $Pr\left(\mathcal{E}_{m\to m'}\right)$ is maximized. This maximum value can be bounded as follows:

$$\max\left(Pr\left(\mathcal{E}_{m\to m'}\right)\right)\leq 1-\sum_{j=0}^{\lfloor\frac{\delta_{m,m'}-1}{2}\rfloor}\binom{N}{j}\alpha^j\left(1-\alpha\right)^{N-j}\tag{49}$$

where α is the BSC crossover probability and $\lfloor.\rfloor$ denotes the floor function. Now, we may employ the union bound to obtain:

$$\begin{aligned}Pr\left(\mathcal{E}_d\right)&=Pr\left(\bigcup_{\substack{m,m'\\m'\neq m}}\mathcal{E}_{m\to m'}\right)\\&\leq\textstyle\sum_{\substack{m,m'\\m'\neq m}}Pr\left(\mathcal{E}_{m\to m'}\right)\\&\leq\textstyle\sum_{\substack{m,m'\\m'\neq m}}\max\left(Pr\left(\mathcal{E}_{m\to m'}\right)\right)\\&\leq\textstyle\sum_{\substack{m,m'\\m'\neq m}}\left(1-\sum_{j=0}^{\lfloor\frac{\delta_{m,m'}-1}{2}\rfloor}\binom{N}{j}\alpha^j\left(1-\alpha\right)^{N-j}\right)\end{aligned}\tag{50}$$

Therefore, by noting that the right hand side of (49) is a decreasing function of $\delta_{m,m'}$, and using the definition of $\check{\delta}$ which is given in (46), we conclude that:

$$Pr\left(\mathcal{E}_d\right)\leq M\left(M-1\right)\left(1-\sum_{j=0}^{\lfloor\frac{\check{\delta}-1}{2}\rfloor}\binom{N}{j}\alpha^j\left(1-\alpha\right)^{N-j}\right)\tag{51}$$

Since the bound provided in (51) is a decreasing function of $\check{\delta}$, we propose to search for a set of cosets with large MMPD, $\check{\delta}$. For this search, we take a systematic approach as follows. We select an integer n_F such that $n_F>log_2M$. Then, we form a search space consisting of 2^{n_F} cosets corresponding to the combinations of frozen bits denoted by vectors $\left\{\boldsymbol{f}^0,\ldots,\boldsymbol{f}^{2^{n_F}-1}\right\}$, where for $\boldsymbol{f}^m$ we set the first n_F bits according to the binary representation of the decimal number m, and the rest of the bits to zero. For example, we set $\boldsymbol{f}^5=(1,0,1,0,\ldots,0)$. We generate the $2^{n_F}\times 2^{n_F}$ MPD matrix $\Delta=\left[\delta_{m,m'}\right]$ for these cosets. Then, we run a sequential algorithm given in Algorithm 1 to find a set of M cosets with the maximum achievable MMPD in this search space.

In Algorithm 1, $\mathcal{Y}\left(m;\check{\delta}\right)$ identifies the set of all cosets $\mathcal{C}^{m'}$ with an MPD greater than or equal to $\check{\delta}$ from the coset $\mathcal{C}^m$; i.e.,

$$\mathcal{Y}\left(m;\check{\delta}\right)=\left\{m'\in\left\{0,\ldots,2^{N_F-1}\right\}\mid\delta_{m,m'}\geq\check{\delta}\right\}\tag{52}$$

The algorithm initiates $\check{\delta}_{opt}$ as the maximum achievable MMPD; then follows a sequential process which iteratively extends a search tree. The variable τ_j denotes the number of remaining nodes at stage j to be examined for a possible extension, which we refer to as *non-examined* nodes. When the tree is extended to stage j, a path from a leaf node to the root of the tree represents a set of $j+1$ cosets with an MMPD greater than or equal to $\check{\delta}_{opt}$. This is clear from Step 1 of the algorithm, where node ν_{j,τ_j-1} generates τ_{j+1} child nodes; where the child nodes are picked according to the set $\bigcap_{j'=0}^{j} \mathcal{Y}\left(\mathbf{val}\left(\nu_{j',\tau_{j'}-1}\right);\check{\delta}_{opt}\right)$. Therefore, the child nodes of ν_{j,τ_j-1} correspond to the cosets that have an MPD greater than or equal to $\check{\delta}_{opt}$ from the coset corresponding to ν_{j,τ_j-1} (that is $\mathcal{C}^{\mathbf{val}\left(\nu_{j,\tau_j-1}\right)}$) as well as the cosets corresponding to the ancestors of ν_{j,τ_j-1} (that are $\mathcal{C}^{\mathbf{val}\left(\nu_{j',\tau_{j'}-1}\right)}$ for $j' \in \{0,\ldots,j-1\}$). This confirms that each child node of ν_{j,τ_j-1} along with its ancestors introduces a set of cosets with an MMPD greater than or equal to $\check{\delta}_{opt}$.

Step 2 of Algorithm 1 indicates that if ν_{j,τ_j-1} has child nodes that have MPDs greater than or equal to $\check{\delta}_{opt}$ from all of their ancestors (including ν_{j,τ_j-1}), then the node ν_{j,τ_j-1} can be extended by running Steps 7 and 8 of the algorithm. After completing this extension, the algorithm moves to Step 9 where it checks whether the tree has reach stage numbered $M-1$ (i.e., whether a set of M cosets with an MMPD greater than or equal to $\check{\delta}_{opt}$ are found). If not, j is incremented and the algorithm attempts to extend the tree to the next stage by returning to Step 1. Otherwise, the algorithm moves to Step 10 where the set of M cosets with an MMPD greater than or equal to $\check{\delta}_{opt}$ is given as the search result.

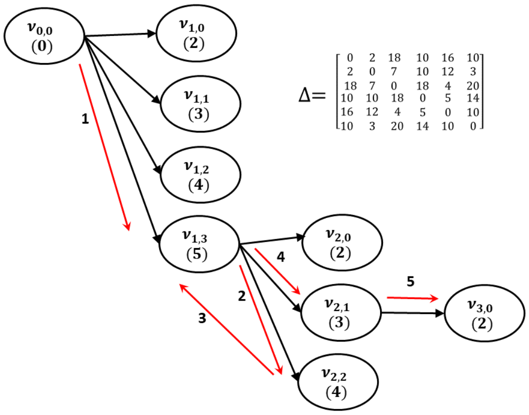
If the algorithm reaches a node ν_{j,τ_j-1} for which the number of child nodes is equal to zero (i.e., $\tau_{j+1}=0$), ν_{j,τ_j-1} cannot be extended. In such a case, the algorithm attempts to extend the next available node at stage numbered j. For this, the algorithm runs its Steps 3 and 4 to decrement the number of non-examined nodes at the j-th stage, and check whether this number is still positive, i.e., if $\tau_j > 0$. If $\tau_j > 0$, then the algorithm goes to Step 1 and considers the next non-examined node at the j-th stage of the tree. For example, if $\tau_j = 5$ and node $\nu_{j,4}$ cannot be extended, running Steps 3, 4, and 1 replaces $\nu_{j,4}$ by $\nu_{j,3}$ in (53), where $v_{j,3}$ is the next non-examined node (note that the algorithm examines the nodes in descending order of their indexes).

Step 5 of Algorithm 1 considers the case where there are no further non-examined nodes at stage j (i.e., $\tau_j = 0$). In this case, the algorithm needs to move back to stage $j-1$ and check whether there are remaining non-examined nodes at that stage. This task is fulfilled at Step 5 by decrementing j and returning to Step 3. Step 6 of the algorithm indicates that if the root node of the tree has no further non-examined child nodes (i.e., if $\tau_1 = 0$), it is no longer possible to extend the tree and the search process for the given value of $\check{\delta}_{opt}$ has failed. In such a case, the algorithm decrements $\check{\delta}_{opt}$ and initiates a new search by going back to Step 0.

1) Examples: Figure 3 demonstrates an example of the sequential search process, where the search space consists of 6 cosets with an MPD matrix Δ shown in the figure. The goal is to find 4 cosets with an MMPD greater than or equal to 10. The value corresponding to each node is denotes in parentheses below the node name. Steps taken by Algorithm 1 are shown by

Algorithm 1 Sequential Search for a Subset of Cosets with largest achievable MMPD.

```
Initialization: Let $\check{\delta}_{opt} \leftarrow \max_{m,m' \in \{0,1,\ldots,2^{n_F}-1\}} \delta_{m,m'}$.
(0) Generate $\nu_{0,0}$ as the root node of a search tree, and assign the value of $\mathbf{val}(\nu_{0,0}) = 0$ to it. Set $j = 0$, and $\tau_0 = 1$.
(1) Let:
        $\tau_{j+1} \leftarrow \left| \bigcap_{j'=0}^{j} \mathcal{Y}\left(\mathbf{val}\left(\nu_{j',\tau_{j'}-1}\right);\check{\delta}_{opt}\right) \right|$    (53)
(2) if $\tau_{j+1} > 0$, go to Step 6.
(3) Let $\tau_j \leftarrow \tau_j - 1$.
(4) If $\tau_j > 0$, go to Step 1.
(5) If $\tau_j = 0$ and $j > 1$, let $j \leftarrow j-1$, then go to Step 3.
(6) If $\tau_j = 0$ and $j = 1$, let $\check{\delta}_{opt} \leftarrow \check{\delta}_{opt} - 1$ then go to Step 0.
(7) Extend $\nu_{j,\tau_j-1}$ by generating its $\tau_{j+1}$ child nodes as $\nu_{j+1,0},\ldots,\nu_{j+1,\tau_{j+1}-1}$.
(8) For $l \in \{0,\ldots,\tau_{j+1}-1\}$, let $\mathbf{val}(\nu_{j+1,l}) = \tilde{y}_l$, where $\tilde{y}_l$ denotes the $l$-th element of the ordered set found by sorting the set $\bigcap_{j'=0}^{j} \mathcal{Y}\left(\mathbf{val}\left(\nu_{j',\tau_{j'}-1}\right);\check{\delta}_{opt}\right)$ in ascending order.
(9) If $j < M$, let $j \leftarrow j+1$ and go to Step 1.
(10) Output the set $\left\{\mathbf{val}\left(\nu_{0,\tau_0-1}\right),\ldots,\mathbf{val}\left(\nu_{M-1,\tau_{M-1}-1}\right)\right\}$ and terminate the algorithm.
```



$$\Delta = \begin{bmatrix} 0 & 2 & 18 & 10 & 16 & 10 \\ 2 & 0 & 7 & 10 & 12 & 3 \\ 18 & 7 & 0 & 18 & 4 & 20 \\ 10 & 10 & 18 & 0 & 5 & 14 \\ 16 & 12 & 4 & 5 & 0 & 10 \\ 10 & 3 & 20 & 14 & 10 & 0 \end{bmatrix}$$

Figure 3. Example of the sequential search process for finding 4 cosets with an MMPD greater than or equal to 10.

red arrows. The algorithm terminates after running for 5 steps. At Step 1, the algorithm intends to extend the tree from Stage numbered $j = 0$ to Stage numbered $j + 1 = 1$. From (52), (53) it is found that $\mathcal{C}^0$ has $\tau_1 = 4$ child nodes represented by the set $\mathcal{Y}(0; 10) = \{2, 3, 4, 5\}$. Therefore, the tree is extended by generating the 4 child nodes $\nu_{1,0}, \nu_{1,1}, \nu_{1,2}, \nu_{1,3}$ and assigning the values $2, 3, 4, 5$ to them. At Step 2, the algorithm attempts to extend node $\nu_{1,3}$ from Stage $j = 1$ to Stage $j + 1 = 2$. For this, using (52) the algorithm finds $\mathcal{Y}(5; 10) = \{0, 2, 3, 4\}$, then using (53) it finds $\tau_2 = |\mathcal{Y}(0; 10) \cap \mathcal{Y}(5; 10)|$ which gives $\tau_2 = 3$. Therefore, $\nu_{1,3}$ is extended by generating its 3 child nodes named as $\nu_{2,0}, \nu_{2,1}, \nu_{2,2}$. The values of these nodes are assigned after sorting the set $\mathcal{Y}(0; 10) \cap \mathcal{Y}(5; 10)$, which gives their corresponding values as $2, 3, 4$. Step 3 in Fig. 3 shows the situation where the algorithm attempts to extend $\nu_{2,2}$ from Stage $j = 2$ to Stage $j + 1 = 3$. However, by noting that $\mathcal{Y}(4; 10) = \{0, 1, 5\}$, from (53), the number of child nodes available for $\nu_{2,2}$ is found as:

$$\tau_3 = |\mathcal{Y}(0; 10) \cap \mathcal{Y}(5; 10) \cap \mathcal{Y}(4; 10)| = 0 \tag{54}$$

i.e., $\nu_{2,2}$ cannot be extended to Stage $j = 3$. Therefore, the algorithm returns to the parent node of $\nu_{2,2}$, i.e., $\nu_{1,3}$, and attempts to extend its next child node which is $\nu_{2,1}$. Note that when $\tau_3 = 0$, the condition at Step (2) of Algorithm 1 is not satisfied; therefore, the algorithm moves to Step (3) and updates τ_2 as $\tau_2 = 2$. Then, since the condition in Step (4) of the algorithm is satisfied, it returns to Step (1) and attempts to find a new value for τ_3, as the number of child nodes for $\nu_{2,1}$. This time, $\mathcal{Y}(4; 10)$ in (54) is replaced by $\mathcal{Y}(3; 10)$ where $\mathcal{Y}(3; 10) = \{0, 1, 2, 5\}$. Therefore, in this case $\tau_3 = 1$ and $\nu_{2,1}$ may be extended to Stage 3 by generating its child node, $\nu_{3,0}$. Since the tree has reach stage numbered 3, four cosets with MMPD greater than or equal to 10 are found by following the path from the root node to the leaf node $\nu_{3,0}$, as cosets numbered $(0, 5, 3, 2)$.

Now, let us demonstrate the potential of the proposed coset design method by giving a toy example in which 4 cosets of $(32, 16)$ polar code are employed to encode data segments of length 16 bits. For coset design purposes, we set $n_F = 5$; hence, the search space consists of 32 cosets denoted by $\mathcal{C}^0$ through $\mathcal{C}^{31}$, out of which 4 cosets with maximum possible MMPD will be selected using Algorithm 1.

Using (43), (48), it is straightforward to show that $\delta_{m,m'}$ is equal to the minimum distance between the codewords of the polar code and the vector $\left(\tilde{\boldsymbol{u}}\left(\boldsymbol{f}^m\right) \oplus \tilde{\boldsymbol{u}}\left(\boldsymbol{f}^{m'}\right)\right) \times G$. After finding all values $\delta_{m,m'}$, $m, m' \in \{0, 1, \ldots, 31\}$, which give the MPD matrix, Δ, we run Algorithm 1 and find 4 cosets with the maximum achievable MMPD. We refer to these 4 cosets by the *designed cosets*.

The probability of detection error, i.e., $p_d = Pr(\mathcal{E}_d)$, for the 4 designed cosets is plotted in Fig. 4. The curve is found by Monte Carlo simulation where 4 randomly generated data segments of length 16 bits are encoded by the 4 cosets, then the encoded vectors are permuted according to a permutation pattern $\boldsymbol{\pi}$ which is drawn from $\mathcal{P}_M$ uniformly at random. Then, BSC noise is added to the permuted vectors to obtain the received vectors $\boldsymbol{r}^{\pi_0}, \ldots, \boldsymbol{r}^{\pi_3}$. Finally, the permutation pattern, $\hat{\boldsymbol{\pi}}$, is detected by the proposed joint decoding method. This process is repeated for several times, and p_d is estimated by dividing the number of tries for which $\hat{\boldsymbol{\pi}} \neq \boldsymbol{\pi}$, by the total number of tries.

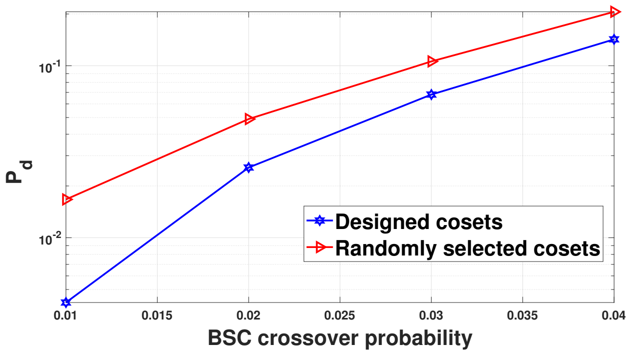


Figure 4. Performance of the proposed coset design method for a toy example with 4 cosets of $(32, 16)$ polar code.

In Fig. 4 we also plot the curve for a benchmark scheme for which the cosets are randomly selected and are re-generated at each try, but their realizations are known at the decoder. It is observed that the designed cosets significantly reduce the probability of detection error compared to this benchmark scheme with randomly selected cosets.

Remark 4. For the toy example considered in this section, it is feasible to generate all 2^{16} codewords and find the exact value of $\delta_{m,m'}$. However, when K is large, generating all 2^K codewords is not feasible. For the case of large K, we generate the artificial reliability vector, $\boldsymbol{L}(\boldsymbol{a}, \epsilon)$, according to (13) by letting $\boldsymbol{a} = \left(\tilde{\boldsymbol{u}}\left(\boldsymbol{f}^m\right) \oplus \tilde{\boldsymbol{u}}\left(\boldsymbol{f}^{m'}\right)\right) \times G$. Then, we give this reliability vector to a list decoder with a list size of L. If $\boldsymbol{x}'$ denotes the codeword on the list with the minimum Hamming distance from $\boldsymbol{a}$, then $\delta_{m,m'}$ can be approximated as $d_H(\boldsymbol{a}, \boldsymbol{x}')$. By increasing the list size L (e.g. $L = 1024$), and adjusting the value of ϵ, one may improve the accuracy of this approximation.

C. Reducing the Complexity of Joint Decoder

Constructing the CSD matrix requires running M^2 decoders, that imposes a large computational complexity. However, when the cosets are designed based on the frozen bits, we may take a lower complexity approach to construct a CSD matrix, albeit with lower accuracy. For this, we note that by making arguments similar to those provided in Remark 3, one may show that the designed cosets are subsets of an $(N, K + n_F)$ polar code, which we refer to as the *supercode*. Hence, each $\boldsymbol{r}^{m'}$ can be regarded as a noisy codeword of this supercode and can be decoded over this supercode. To implement such an approach, at the decoder we *unfreeze* the n_F frozen bits that contribute to the coset design process. Then, we run a list decoder to find L codewords of the supercode, which we refer to by *super-codewords*. We partition the set of L super-codewords into M

disjoint subsets denoted by $\check{\mathcal{X}}^{m,m'}$ such that the j-th codeword in $\check{\mathcal{X}}^{m,m'}$ is expressed as $\check{\boldsymbol{u}}^{m,j} \times \boldsymbol{G}$, where $\check{\boldsymbol{u}}^{m,j}$ has the property that its n_F unfrozen bits are identical to the first n_F bits of $\boldsymbol{f}^m$, i.e.,

$$\left(\check{u}^{m,j}_{\rho_K}, \check{u}^{m,j}_{\rho_{K+1}}, \ldots, \check{u}^{m,j}_{\rho_{K+n_F-1}}\right) = \left(f^m_0, f^m_1, \ldots, f^m_{n_F-1}\right) \tag{55}$$

for $j \in \left\{0, 1, \ldots, |\check{\mathcal{X}}^{m,m'}|\right\}$.

Due to the one-to-one mapping between $\mathcal{C}^m$ and $\boldsymbol{f}^m$, for every $\boldsymbol{x'} \in \check{\mathcal{X}}^{m,m'}$ we have $\boldsymbol{x'} \in \mathcal{C}^m$, i.e., $\check{\mathcal{X}}^{m,m'} \subset \mathcal{C}^m$. Therefore, the vectors $\boldsymbol{x'} \in \check{\mathcal{X}}^{m,m'}$ may be employed as candidates to approximate the CSD between $\mathcal{C}^m$ and $\boldsymbol{r}^{m'}$ by defining:

$$\check{\gamma}_{m,m'} = \begin{cases} \min\limits_{\boldsymbol{x'} \in \check{\mathcal{X}}^{m,m'}(L)} d_H\left(\boldsymbol{x'}, \boldsymbol{r}^{m'}\right); & \check{\mathcal{X}}^{m,m'} \neq \emptyset \\ d_H\left(\boldsymbol{e}^m, \boldsymbol{r}^{m'}\right); & \text{otherwise} \end{cases} \tag{56}$$

The second condition in (56) indicates that when $\check{\mathcal{X}}^{m,m'}$ is an empty set, i.e., when none of the L decoded super-codewords is in $\mathcal{C}^m$, we estimate the CSD as the Hamming distance between $\boldsymbol{r}^{m'}$ and the coset leader of $\mathcal{C}^m$. Note that by definition, the CSD is the minimum distance of the members of $\mathcal{C}^m$ from $\boldsymbol{r}^{m'}$; hence, it may be bounded as $d_H\left(\boldsymbol{e}^m, \boldsymbol{r}^{m'}\right)$[2].

Now, we may generate a CSD matrix $\check{\boldsymbol{\Gamma}} = [\check{\gamma}_{m,m'}]$ and employ it in the joint decoding process. Although this CSD matrix leads to a suboptimal decoding, it can be generated by a significantly lower computational complexity. The reason is that for generating each column of $\check{\boldsymbol{\Gamma}}$, a single list decoder is employed over the $(N, K + n_F)$ super-code, whereas each column of $\boldsymbol{\Gamma}$ is generated by running M list decoders over the (N, K) polar code (see Eq. (14)). Therefore, this proposed approach may be employed to trade-off performance for computational complexity.

Note that the above proposed low complexity decoder is only applicable to designed cosets and not to randomly selected cosets. The reason is that for randomly selected cosets, vectors $\boldsymbol{f}^0$ through $\boldsymbol{f}^{M-1}$ are picked uniformly at random from the set $\mathcal{B}_{N-K}$; therefore, the number of active frozen bits is not limited (unlike the designed cosets for which this number is n_F). In other words, for randomly selected cosets, the supercode will have rate 1 and decoding over it will give no information (unlike designed cosets for which the supercode has a rate $\frac{K+n_F}{N} < 1$).

1) Limiting the Number of Examined Permutations: A further complexity concern is that according to (19), the joint decoder examines all $M!$ permutation patterns in $\mathcal{P}_M$ in order to infer the most likely pattern from the CSD matrix, Γ. When M is large, the complexity of this search is prohibitive. To overcome this drawback, we propose a low-complexity (but suboptimal) iterative search method that produces a permutation pattern, $\hat{\boldsymbol{\pi}}$, by taking only M iterations. The proposed search method works as follows. In each column of the CSD matrix, we find the difference between the minimum valued entry and the entry that has the next smallest value. Then, we select the column that maximizes this difference. Let us assume that the maximum difference occurs for the m'_0-th column in which the m_0-th row contains the minimum valued entry. We assign the m'_0-th received segment to m_0-th coset; then, we exclude the m_0-th row and the m'_0-th column from the matrix, and proceed to the next iteration.

The justification for taking this approach is that if $\mathcal{C}^m$ is the coset with the minimum CSD to $\boldsymbol{r}^{m'}$, with a CSD of $\gamma_{m,m'}$, and if $\mathcal{C}^{m''}$ is the second closest coset to $\boldsymbol{r}^{m'}$ with a CSD of $\gamma_{m'',m'}$, then, roughly speaking, a larger value of $\gamma_{m'',m'} - \gamma_{m,m'}$ implies a larger possibility that $\boldsymbol{r}^{m'}$ belongs to $\mathcal{C}^m$. Hence, it is reasonable to find the segment for which the difference is maximized and make a decision about that segment.

Let us demonstrate the above-proposed process by giving an example. Consider the following CSD matrix corresponding to three cosets and three received vectors:

$$\boldsymbol{\Gamma}_0 = \begin{bmatrix} \gamma_{0,0} & \gamma_{0,1} & \gamma_{0,2} \\ \gamma_{1,0} & \gamma_{1,1} & \gamma_{1,2} \\ \gamma_{2,0} & \gamma_{2,1} & \gamma_{2,2} \end{bmatrix} = \begin{bmatrix} 4 & 0 & 2 \\ 17 & 10 & 9 \\ 14 & 5 & 16 \end{bmatrix} \tag{57}$$

where the index 0 in Γ_0 indicates the initial state, i.e., iteration numbered 0. In $\boldsymbol{\Gamma}_0$, the differences between minimum and next smallest entries for $\boldsymbol{r}^0$, $\boldsymbol{r}^1$, $\boldsymbol{r}^2$ are $10, 5, 7$. Therefore, $\boldsymbol{r}^0$ is selected and since $\mathcal{C}^0$ has the minimum CSD from $\boldsymbol{r}^0$, $\boldsymbol{r}^0$ is mapped to $\mathcal{C}^0$. Then, $\boldsymbol{r}^0$ and $\mathcal{C}^0$ are excluded from the search by excluding their corresponding column and row, and updating the CSD matrix as follows:

$$\boldsymbol{\Gamma}_1 = \begin{bmatrix} \gamma_{1,1} & \gamma_{1,2} \\ \gamma_{2,1} & \gamma_{2,2} \end{bmatrix} = \begin{bmatrix} 10 & 9 \\ 5 & 16 \end{bmatrix} \tag{58}$$

where the index 1 in $\boldsymbol{\Gamma}_1$ indicates that we have moved to iteration numbered 1. For $\boldsymbol{\Gamma}_1$, the differences of entries for the columns corresponding to $\boldsymbol{r}^1$ and $\boldsymbol{r}^2$ are 5 and 7, respectively. Therefore, $\boldsymbol{r}^2$ is selected and since $\mathcal{C}^1$ has the minimum-valued CSD from $\boldsymbol{r}^2$, $\boldsymbol{r}^2$ is assigned to $\mathcal{C}^1$. Then, the CSD matrix is updated by excluding the column and the row corresponding to $\boldsymbol{r}^2$ and $\mathcal{C}^1$, respectively. The updated CSD matrix is expressed as $\boldsymbol{\Gamma}_2 = [\gamma_{2,1}] = [5]$. In the final iteration, the only remaining segment in $\boldsymbol{\Gamma}_2$, which is $\boldsymbol{r}^1$, is mapped to the only remaining coset in $\boldsymbol{\Gamma}_2$, which is $\mathcal{C}^2$. Therefore, the received sequence is sorted as $\left(\boldsymbol{r}^0, \boldsymbol{r}^2, \boldsymbol{r}^1\right)$ which is equivalent to detecting a permutation pattern $\hat{\boldsymbol{\pi}} = (0, 2, 1)$.

VI. NUMERICAL RESULTS

In this section, we provide numerical examples to quantify the performance of the proposed scheme, and compare it with that of the explicit indexing method. We also compare the performance of designed cosets against the benchmark scheme that employs randomly selected cosets. In addition, the performance of the low-complexity joint decoder is compared with that of the optimal joint decoder through these numerical results. The 5G

---

[2] One may tighten this bound by including $\tilde{\boldsymbol{u}}\left(\boldsymbol{f}^m\right) \times G$ which is another member of $\mathcal{C}^m$.

standardization unique channel-reliability sequence is employed for polar coding [24].

We study two setups, denoted by Setup A and Setup B. For Setup A, the $(255, 225)$ RS code is employed as the outer code and 32 cosets of the $(128, 64)$ polar code are employed as the inner code. Also, a list decoder with a list size of $L = 4$ is employed for generating the CSD matrix. For Setup B, the outer code is the $(63, 35)$ RS code while inner encoding and indexing are performed by 9 cosets of the $(64, 42)$ polar code. The CSD matrix is generated by aid of list decoders with a list size of $L = 1$.

Figure 5 shows FER curves for Setup A, when explicit indexing and designed cosets are employed, respectively. For the coset design process, we take $n_F = 8$ frozen bits. The analytical approximation found for random coding is also plotted in the figure. It is observed that the designed cosets outperform explicit indexing and also achieve FERs lower than those predicted by the analytical approximation.

Figure 6 shows the performance of the low-complexity decoding method. The curve labeled as "full decoding" shows the case where the CSD matrix is constructed by running M^2 decoders with a list size of 4; whereas for low-complexity decoding case, the matrix is constructed by running M list decoder with a list size of 128. Although the results suggest a suboptimal performance for the low-complexity decoder, this low-complexity method offers a choice for trading off the performance for the computational complexity. Also, since $M = 32$ cosets are employed in Setup A, searching over all $M!$ permutations is infeasible; therefore, in both cases of full and low-complexity decoding, the method proposed in V-C1 is employed to deduce $\hat{\pi}$ from the generated CSD matrix.

Figure 7 shows the frame error rate and the probability of detection error, p_d, for Setup B, for two cases that correspond to the benchmark scheme with randomly selected cosets, and the designed cosets with $n_F = 12$, respectively. It is observed that the designed cosets outperform random cosets in terms of both FER and p_d; however, their advantage is more significant for the case of p_d.

Figure 8 shows the FER curves for Setup B, when designed cosets with $n_F = 6$ are employed. For the low-complexity decoding, the matrix is generated by running $M = 9$ list decoders of list size L (where both cases of $L = 8$ and $L = 32$ are examined. For the full decoding, $M^2 = 81$ list decoders with a list size of 4 are employed to generate the CSD matrix. It is observed that by increasing the list size, L, the low-complexity decoder is capable of offering a performance which is close to the one offered by the full decoder.

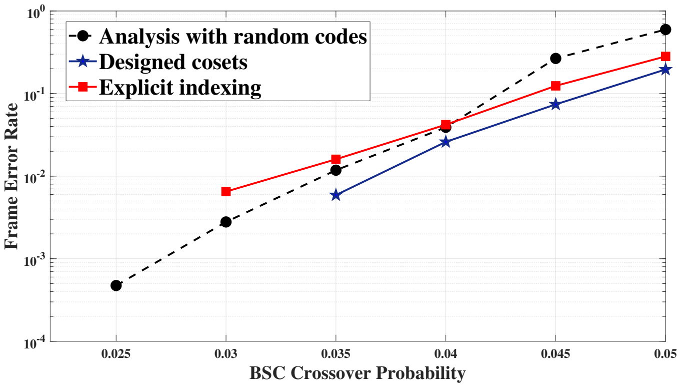


Figure 5. FERs achieved for Setup A by employing explicit indexing and designed cosets, respectively. The analytical approximation found for random coding is also plotted as a reference.

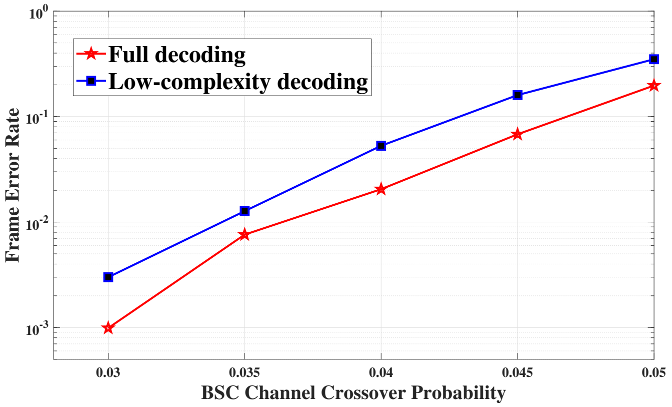


Figure 6. FERs achieved for Setup A when full decoding and low-complexity decoding are applied to generate the CSD matrix.

VII. CONCLUSIONS

We propose an implicit indexing approach for data transmission over a noisy shuffling channel, where data is encoded by an outer RS code, then the RS codeword is sliced into short-length segments, which are encoded by separate cosets of a polar code. We design a joint decoder that detects the permutation pattern and performs polar decoding. The joint decoder generates a coset-segment distance matrix and infers the permutation pattern from this matrix. We establish a one-to-one mapping between the subsets of the frozen bits and the cosets of the polar code; then we employ this mapping to design cosets offering low probabilities of detection error. Furthermore, a low-complexity decoding method is proposed that trades off the performance of the joint decoder for its computational complexity. Through random coding analysis, an analytical approximation is derived for the frame error rate of the proposed scheme. We also show that explicit indexing is equivalent to a special case of the proposed coset-based indexing, and is outperformed by the designed cosets.

Performance analysis of the proposed scheme for noisy shuffling channels with insertion, deletion and substitution errors, and design of suitable inner codes for those channels, are among interesting directions for future research.

REFERENCES

[1] J. Haghighat and T. M. Duman, "A practical concatenated coding scheme for noisy shuffling channels with coset-based indexing," in proceedings IEEE GLOBECOM, December 2023.

[2] G. M. Church, Y. Gao, and S. Kosuri, "Next-generation digital information storage in DNA," Science, vol. 337, no. 6102, pp. 1628–1628, Sep. 2012.

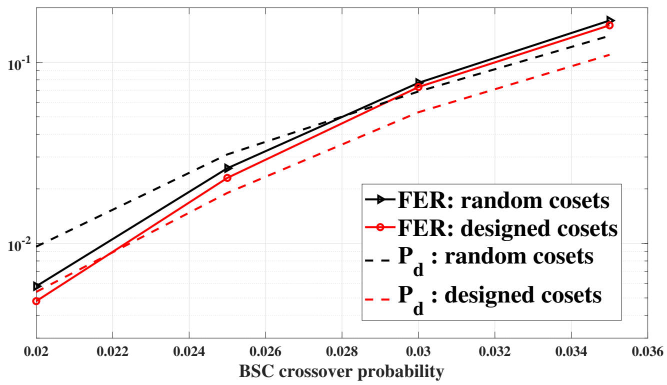


Figure 7. Frame error rates and probabilities of detection error for Setup B when the benchmark scheme with randomly selected cosets, and the designed cosets with $n_F = 12$ are employed.

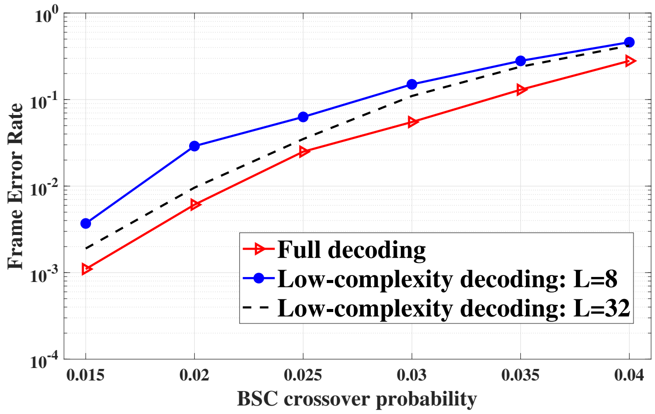


Figure 8. FER curves for Setup B, when designed cosets are employed. The CSD matrix is generated either by full decoding or by the low-complexity decoding method.

[3] N. Goldman, P. Bertone, S. Chen, C. Dessimoz, E. M. LeProust, B. Sipos, and E. Birney, “Towards practical, high-capacity, low- maintenance information storage in synthesized DNA,” Nature, vol. 494, no. 7435, pp. 77–80, 2013.

[4] S. Kosuri and G. Church, “Large-scale de novo DNA synthesis: Technologies and applications”, Nature Methods, vol. 11, no. 5, pp. 499-507, May 2014.

[5] R. N. Grass, R. Heckel, M. Puddu, D. Paunescu, and W. J. Stark, “Robust chemical preservation of digital information on DNA in silica with error correcting codes,” Angew. Chem. Int. Ed., vol. 54, no. 8, pp. 2552–2555, 2015.

[6] L. Organick *et al.*, “Random access in large-scale DNA data storage,” Nature Biotechnology, vol. 36, pp. 242–248, Feb. 2018.

[7] Heckel, R., Mikutis, G., and Grass, R.N. A characterization of the DNA data storage channel. Sci Rep 9, 9663 (2019).

[8] I. Shomorony and R. Heckel, “Capacity results for the noisy shuffling channel,” 2019 IEEE International Symposium on Information Theory (ISIT), 2019, pp. 762-766.

[9] A. Lenz, P. H. Siegel, A. Wachter-Zeh and E. Yaakobi, “Coding over sets for DNA storage,” in IEEE Transactions on Information Theory, vol. 66, no. 4, pp. 2331-2351, April 2020.

[10] I. Shomorony and R. Heckel, “DNA-based storage: Models and fundamental limits,” in IEEE Transactions on Information Theory, vol. 67, no. 6, pp. 3675-3689, June 2021.

[11] M. Kovačević and V. Y. F. Tan, “Codes in the space of multisets – Coding for permutation channels with impairments,” in IEEE Transactions on Information Theory, vol. 64, no. 7, pp. 5156-5169, July 2018.

[12] J. Sima, N. Raviv and J. Bruck, “On coding over sliced information,” in IEEE Transactions on Information Theory, vol. 67, no. 5, pp. 2793-2807, May 2021.

[13] K. Levick, R. Heckel and I. Shomorony, “Achieving the capacity of a DNA storage channel with linear coding schemes,” 2022 56th Annual Conference on Information Sciences and Systems (CISS), Princeton, NJ, USA, 2022, pp. 218-223.

[14] A. Lenz, P. H. Siegel, A. Wachter-Zeh and E. Yaakobi, “Anchor-based correction of substitutions in indexed sets,” 2019 IEEE International Symposium on Information Theory (ISIT), Paris, France, 2019, pp. 757-761.

[15] J. Sima, N. Raviv and J. Bruck, “Robust indexing - Optimal codes for DNA storage,” 2020 IEEE International Symposium on Information Theory (ISIT), Los Angeles, CA, USA, 2020, pp. 717-722.

[16] L. Welter, I. Maarouf, A. Lenz, A. Wachter-Zeh, E. Rosnes and A. G. I. Amat, “Index-based concatenated codes for the multi-draw DNA storage channel,” 2023 IEEE Information Theory Workshop (ITW), Saint-Malo, France, 2023, pp. 383-388.

[17] T. Shinkar, E. Yaakobi, A. Lenz and A. Wachter-Zeh, “Clustering-correcting codes,” in IEEE Transactions on Information Theory, vol. 68, no. 3, pp. 1560-1580, March 2022.

[18] N. Weinberger, “Error probability bounds for coded-index DNA storage systems,” in IEEE Transactions on Information Theory, vol. 68, no. 11, pp. 7005-7022, Nov. 2022.

[19] E. Arikan, “Channel polarization: A method for constructing capacity-achieving codes for symmetric binary-input memoryless channels,” in IEEE Transactions on Information Theory, vol. 55, no. 7, pp. 3051-3073, July 2009.

[20] B. Shuval and I. Tal, “A lower bound on the probability of error of polar codes over BMS channels,” 2017 IEEE International Symposium on Information Theory (ISIT), Aachen, Germany, 2017, pp. 854-858.

[21] G. Ricciutelli, M. Baldi, F. Chiaraluce and G. Liva, “On the error probability of short concatenated polar and cyclic codes with interleaving,” 2017 IEEE International Symposium on Information Theory (ISIT), Aachen, Germany, 2017, pp. 1858-1862.

[22] S. Seyedmasoumian and T. M. Duman, “Approximate weight distribution of polarization-adjusted convolutional (PAC) codes,” 2022 IEEE International Symposium on Information Theory (ISIT), Espoo, Finland, 2022, pp. 2577-2582.

[23] S. J. MacMullan and O. M. Collins, “A comparison of known codes, random codes, and the best codes,” in IEEE Transactions on Information Theory, vol. 44, no. 7, pp. 3009-3022, Nov. 1998.

[24] V. Bioglio, C. Condo and I. Land, “Design of polar codes in 5G new radio,” in IEEE Communications Surveys & Tutorials, vol. 23, no. 1, pp. 29-40, Firstquarter 2021.