Mitigating Blind Detection Through Protograph Based Interleaving for Turbo Codes

Stefan Weithoffer, Rami Klaimi, Charbel Abdel Nour

Mitigating Blind Detection Through Protograph Based Interleaving for Turbo Codes

Stefan Weithoffer, Rami Klaimi, Charbel Abdel Nour
Email: {stefan.weithoffer, rami.klaimi, charbel.abdelnour}@imt-atlantique.fr
IMT Atlantique, Lab-STICC, UMR CNRS 6285, F-29238 Brest, France

***Abstract*—The complexity involved to blindly detect the channel code parameters in the case of their imperfect knowledge is generally measured in terms of the minimum number of frames that an eavesdropper needs to observe for successful detection, adding an additional layer of privacy. In this work, starting from a defined almost regular interleaver for Turbo codes, we propose methods to construct a larger set of distinct interleavers that increases the minimum number of observations by a factor equal to the size of the constructed set. Furthermore, the generated sets of interleavers can be described by defining only a small number of parameters and are shown to achieve a comparable error correcting performance to base interleavers. To validate the proposed implementation-friendly method, an application example for information frame sizes K=128 bits and K=512 bits is provided for the construction of two sets of 8192 interleavers, prohibitively increasing detection complexity by state-of-the-art methods.**

***Keywords*—Turbo Code, Physical Layer Security, Interleaver**

I. Introduction

In the era of *Internet of Things* (IoT) and massive *Machine-to-Machine* (M2M) communications with tens of billions of devices, sensors and actuators wirelessly connected via the internet [1], authentication, confidentiality, and privacy are an increasingly important concern. The possibility of eavesdropping on and/or tampering with the communication of distributed sensors and actuators in medical, industrial or vehicular environments constitutes an obvious and critical threat. To this end, classical wiretap models assume perfect knowledge of the used communication protocol on the side of Eve, who is passively observing the signal transmission between the two legal users Alice and Bob [2], [3]. In a wiretap model, Alice's and Bob's objective is to encode their transmission, so that the statistical dependence between Eve's observation is minimized, while still allowing reliable communcation between Alice and Bob. Since perfect knowledge of system parameters on the side of Eve is assumed, the resulting keyless communication makes this an attractive avenue for achieving physical layer security [4].

On the other hand, in many scenarios, Eve may have an imperfect knowledge of the communication protocol. The communication parameters in use by Alice and Bob, such as the used channel code and modulation parameters, may be designed to represent an implicit secret key fully or partially unknown to Eve. Consequently, measures to strengthen security by exploiting the knowledge related to the characteristics of existing communication systems on all OSI-layers can be seen as complementary to keyless techniques and becomes of particular interest. For practical cases where the communication protocol follows or builds on a well known communication standard, Eve's goal of guessing the secret key reduces to the *recognition/detection* of predefined parameters.

In the case of the channel code, methods of recognition assume access to the digital bitstream of the transmission and make use of the code's linear property [5]–[7]. Methods for reconstructing *Low Density Parity Check* (LDPC) and *Turbo Codes* (TC) often require additional knowledge of the dimensions of the parity check matrix for LDPC codes [8], [9] or the constituent convolutional codes for TC [10]–[12]. For the recovery of the Turbo Code interleaver, which is in the focus of this paper, [12] assumes knowledge about the code dimensions and rate, as well as the positions of the systematic and parity data in the received data stream. Their proposed algorithm has the same complexity ($O(K^2 N_{min})$, K denoting the information frame size and N_{min} being the number of observed transmissions) as prior works from [10], [11]. However, the number of necessary observations N_{min} for keeping the probability of error for the interleaver recovery $p_f < 1\%$ is reduced by a factor of 2.5 to 16 compared to [11].

In order to counter the eavesdropper's efforts and mitigate the detection of their communication parameters, communication systems deployed between Alice and Bob can now aim at lowering the confidence $C = 1 - p_f$ for a given number of observations and/or increase the number of necessary observations denoted by N_{min}. The use of encrypted interleaver tables as proposed in [13] relies on them being shared and decrypted by the receiver side ahead of the data transmission. Provided that the interleaver tables are not used for more than N_{min} frames, the eavesdroppers confidence can be kept below C. However, the necessary encryption/decryption poses a significant overhead which is prohibitive for most embedded applications. Other works aim at increasing N_{min} through the introduction of a pseudo random puncturing scheme [14]–[16]. Based on the assumptions that the pseudo random number sequence is unknown to the eavesdropper and the Turbo Code frame size is sufficiently large, N_{min} becomes prohibitively large, especially for puncturing to high code rates.

In this work, we address weaknesses of both approaches by proposing new methods for defining protograph *Almost Regular Permutation* (ARP) interleavers [17]. By constructively increasing the size of the *shift vector* S of the ARP, we specify a much larger set starting from a small set of base interleavers for a given Turbo Code frame size K. On one side, designed interleavers maintain excellent error correcting performance, outperforming by far random or uniform interleavers. On the other side, the cardinality of the constructed set of interleavers is increased by a large factor that diminishes the probability of the eavesdropper to be able to detect interleaver parameters.

The remainder of this paper is structured as follows. First, section II recalls ARP interleavers and protograph-based construction methods. Then, different transformation operations on ARP parameters are introduced in section III. Based on the proposed operations, we present as a case study an interleaver set for frame size $K = 128$ and $K = 512$ along with simulation results for all interleavers in the set for different code rates. Section V discusses the implications of using our method of increasing N_{min} in relation with state of the art before section VI concludes the paper.

II. Protograph-based ARP interleaving

The most widely-used interleaver families for Turbo decoding are *Quadratic Permutation Polynomial* (QPP) interleaver [18], the *Dithered Relative Prime* (DRP) interleaver [19] and the ARP interleaver [20]. It was shown in [21] that the ARP interleaver can provide the same interleaving properties as QPP and DRP interleavers with the same or higher minimum Hamming distance values. Furthermore, a protograph-based construction method for parity puncture constrained ARP interleavers [22] was presented in [17], allowing the construction of interleavers with low error floors in the presence of puncturing. We briefly recall this method here in order to provide necessary background. For a detailed discussion, the interested reader is referred to [17].

An ARP interleaver is defined by a value P, which is relative prime to the frame size K, a shift vector $\mathbf{S}$ and a disorder degree Q [20]. The interleaving function, defining connections between the bits of the frames at the input of the first and second decoders, is then given by (1), where K denotes the frame size and $\mod$ the modulo operator:

$$\Pi_{\text{ARP}}(i) = \left(P \cdot i + S_{(i \bmod Q)}\right) \bmod K. \tag{1}$$

For a layered construction [17], the interleaver addresses $\Pi(i)$ are divided into Q groups such that

$$\Pi_{\text{ARP}}(i+Q) \mod Q = \Pi_{\text{ARP}}(i) \mod Q. \tag{2}$$

Each consisting of K/Q bits, these Q groups are called *layers* and apply a regular interleaver structure. The layer index l for the bit i in the linear sequence **d** and the layer index l' for bit $\Pi_{ARP}(i)$ in the interleaved sequence **d'** are defined by:

$$l = i \mod Q \tag{3}$$
$$l' = \Pi_{\text{ARP}}(i) \mod Q. \tag{4}$$

Thus, each layer l in **d** is linked to a layer l' in **d'**. For the bits of layer l at index i, a shift value $S(l)$ is then selected. It is composed of an *inter-layer* shift $T_l \in \{0, ..., Q-1\}$ and an *intra-layer* shift $A_l \in \{0, ..., K/Q-1\}$ such that

$$S(l) = T_l + A_l \cdot Q. \tag{5}$$

As illustrated in Figure 1, the inter-layer shift T_l defines the layer l' of the interleaved sequence **d'** that is connected to layer l (i.e. the position within a period of Q) of the non-interleaved sequence **d**.

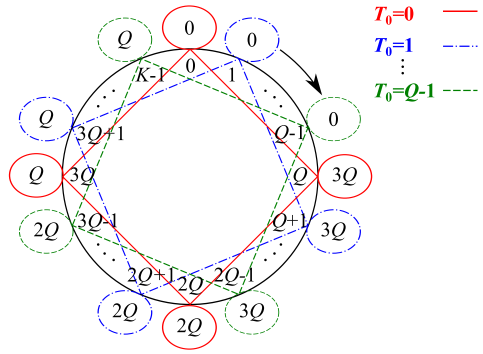

Figure 1. Example of inter-layer shifts for layer $l = 0$, with K = 4Q [17].

The position within layer l' (i.e. which period of Q) that is connected to index i is given by the intra-layer shift A_l (see Figure 2).

Given P, the shift values $S(l)$ can be incrementally selected, establishing a periodic connection pattern with a period of Q. Furthermore, with Q being set equal to the puncturing period M, the validation of additional puncturing constraints can be obtained through the validation of a single puncturing period [17].

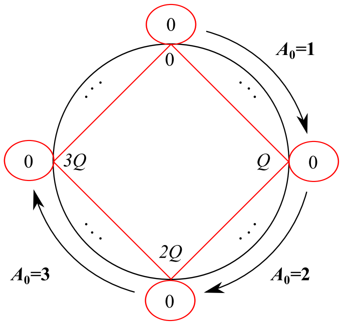

Figure 2. Example of intra-layer shifts for layer $l = 0$, with K = 4Q [17].

The overall construction method then involves the following steps for given code parameters (i.e. frame size K, rate R, constituent code polynomials and puncturing mask) and certain design targets:

1) Select a set $\mathcal{P}$ of candidate values P_c
2) Select the set $\mathcal{S}$ of Q shift values for each $P_c \in \mathcal{P}$
3) Select the best ARP interleaver candidate based on the Turbo Code Hamming distance spectrum

For interleaver design with puncturing constraints, this process is preceded by defining the puncturing constraints after selecting the best puncturing mask for the constituent codes of the Turbo Code. These constraints can be illustrated by a connection graph, the *protograph* (Figure 3), where data positions in the puncturing period M are sorted with respect to their reliability and connected via the interleaver so that highly reliable positions in one component code are connected to unreliable positions in the other component code.

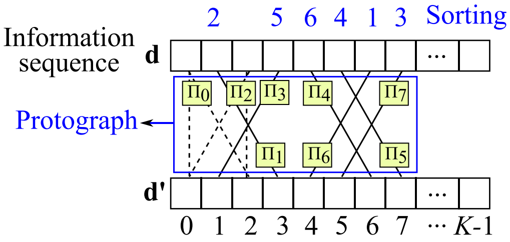

Figure 3. Protograph for $M = 8$ according to the sorting of unpunctured data positions [17].

III. Increasing the number of observations N_{min} for parameter detection

In [10]–[12], knowledge about the code size K and the positions of the systematic and parity data within the received stream is assumed. Moreover, the implicit assumption that the same code is used for all transmissions is also made.

The method discussed in section II progressively (layer by layer) constructs interleaver candidates Π_{ARP} and selects the best ones based on the Turbo Code Hamming distance spectrum. Selected candidates follow (1) and can be supported by the same hardware without overhead. In the following, we propose several implementation-friendly methods to increase the number of admissible candidates while minimizing the design time and maintaining excellent error correcting performance.

A. Base interleaver set

The use of a set $\mathcal{C}$ of interleavers unknown to the attacker, in opposition to only one, increases the number of observations N_{min} required until all interleavers are identified with a confidence $1 - p_f$ by a factor equal to the cardinality $\#\mathcal{C}$.

Any set $\mathcal{C}$ of ARP interleavers can be described jointly by a set $\mathcal{P}$ of values for P, a set $\mathcal{Q}$ of disorder degrees and an associated set $\mathcal{S}$ of shift vectors S (each vector matching corresponding P and Q values). The number of interleavers in the set is then given by

$$\#\mathcal{C} = \left(\sum_{Q_i \in \mathcal{Q}, P_i \in \mathcal{P}} \#\mathcal{S}_{(Q_i, P_i)} \right) \cdot \binom{M}{r} \qquad (6)$$

where $\binom{M}{r}$ gives the number of puncturing patterns of length M for r punctured positions. For the special case $r = 0$ and $\#\mathcal{P} = \#\mathcal{S} = \#\mathcal{Q} = 1$, we get the result from [14].

B. Variable intra-layer shift

With tail-biting termination of the corresponding component code trellises, the sequences **d** and **d'** can be represented as circles [17], [23]. Consequently, the whole information sequence is equally protected by the recursive convolutional code and decoding can start at any arbitrary position within the sequence. Hence to increase N_{min}, we propose to shift the entire interleaver structure in relation to the "start" of the sequences **d** and **d'**. This can be achieved by introducing an additional inter-layer shift $s_T = 0, ..., K - 1$ into (5):

$$S(l) = T_l + s_T + A_l \cdot Q. \qquad (7)$$

Figure 4 illustrates the behavior of the resulting ARP interleavers after shifting by $s_T \in 0, ..., K - 1$. In particular the interleavers shown in Figure 4 are given by

$$\begin{aligned} K =& 128, Q = 4, P = 49 \\ S =& \{3 + s_T, 113 + s_T, 111 + s_T, 93 + s_T\}. \end{aligned} \qquad (8)$$

The *Frame Error Rate* (FER) is given for transmission over AWGN channel and max-Log-MAP decoding with 8 turbo iterations. Since we keep the intra-layer connection structure of the interleaver, the 127 derived interleavers keep roughly the same error correcting performance of the base interleaver.

Hence, we can increase the number of available interleavers for a given $Q_i \in \mathcal{Q}$ by a factor of K.

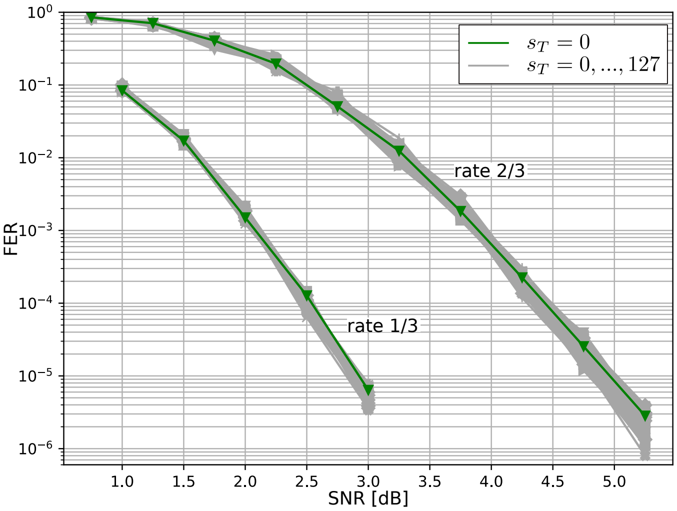

Figure 4. FER perf. of interleavers following (8) for code rates 1/3 and 2/3.

C. Switching the layer starting positions

The starting positions for each layer in the interleaved order, i.e. the first Q addresses $\Pi(0)...\Pi(Q - 1)$ are determined by S and the period P. Due to the regular permutation structure within each layer, subsequent addresses can be determined by adding the same factor (multiple of $Q \times P$) to each starting layer position. Therefore, interleaver design constraints such as the minimum spread and correlation cycle length [17] depend solely on the difference between the starting layer positions.

Having a limited impact on this difference, we propose to construct additional interleaver candidates by permuting the values of these positions between layers, i.e. permuting the A_l values, by modifying the shift vector S:

$$S_i^{'} = (S_i + iP) \bmod Q + \left\lfloor \frac{S_{\delta_{LS}(i)} + \delta_{LS}(i) \cdot P}{Q} \right\rfloor \cdot Q. \quad (9)$$

In (9), $\delta_{LS}(i)$ gives the position in a permutation vector of length Q. Assuming a set $\mathcal{D}$ with *layer start permutations* δ_{LS} for each of the base interleavers, we can add another multiplicative factor $\#\mathcal{D}$ to (6).

D. Switching the protograph connections

Motivated by the same type of arguments for the permutation of starting layer addresses, we can change the protograph connections of a given interleaver by modifying the shift vector:

$$S_i^{'} = S_i + \big(S_{\delta_{PG}(i)} + \delta_{PG}(i) \cdot P - S_i + i \cdot P\big) \bmod Q. \quad (10)$$

Assuming a set $\mathcal{E}$ with *protograph permutations* δ_{PG} for each member of $\mathcal{C}$, we can multiply (6) by $\#\mathcal{E}$:

$$\#\mathcal{C} = K\#\mathcal{D}\#\mathcal{E}\left(\sum_{Q_i \in \mathcal{Q}, P_i \in \mathcal{P}} \#\mathcal{S}_{(Q_i,P_i)}\right) \cdot \binom{M}{r}. \quad (11)$$

For high enough Q values, switching the starting positions or changing the protograph connections of two layers is expected to have a limited impact on the base interleaver properties. This is due to the nature of these operations and to the limited number of positions (K/Q) affected by the operations. After re-evaluation, generated interleavers could increase the set of candidates by $Q!$, where ! denotes the factorial operator.

E. Lifting the disorder degree Q

To extend the possible permutations for (9) and (10), the size of the disorder degree vector can be increased. To do so, we define a *lifting factor* l respecting $Q' = l \cdot Q$ and $K \bmod l = 0$ associated with a lifting permutation as follows:

$$S_i' = S_{i \bmod Q} + P \cdot Q \cdot \left(\delta_L(i \bmod Q) - \left\lfloor \frac{i}{Q} \right\rfloor\right). \quad (12)$$

In (12) δ_L defines a connection in the *lifting permutation*. Figure 5 shows the lifting by a factor of $l = 4$ of a base interleaver with period $Q = 4$ using a lifting permutation $\delta_L = (3, 0, 1, 2)$. Note, that a lifting with $\delta_{Base} = \{0, 1, 2, ..., l-2, l-1\}$ preserves the connections of the base interleaver. This is particularly useful if the disorder degree Q_0 of the base interleaver is small, but a large number of protograph switchings or layer-start switchings are desired. Lifting the base interleaver to a larger disorder degree $Q_1 > Q_0$ increases the number of candidates by $Q_1! - Q_0!$. However, lifting with a permutation other than δ_{Base} implies also permuting the protograph as well as the layer starting positions. Therefore, we will in the following only consider lifting with δ_{Base} and not count interleavers with only a lifted Q in (11).

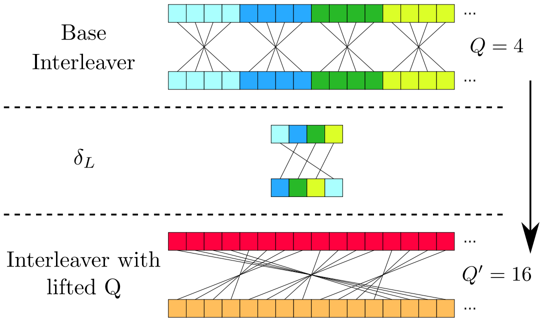

Figure 5. Lifting a base interleaver from $Q = 4$ to $Q' = 16$ via δ_L.

IV. Case Study

Based on the proposals of the previous section, large interleaver sets can be described by a limited set of parameters, as opposed to storing entire tables. Only the set of periods $\mathcal{P}$, the base interleaver shift vectors $\mathcal{S}$, the set of permutations of the starting layer addresses $\mathcal{D}$, the set of protograph permutations $\mathcal{E}$, and the set of lifting graphs $\mathcal{L}$ are needed for the proposal.

As a case study, we constructed $\mathcal{C}_{128}$ from a set of four base interleavers $\{I_0, I_1, I_2, I_3\}$ and $\mathcal{C}_{512}$ from $\{I_4\}$ for which the parameters are listed in Table I.

Table I
Set of base interleavers for $K = 128$ and $K = 512$ bits.

	K	P	Q	S
IL$_0$	128	15	4	$[3, 87, 51, 3]$
IL$_1$	128	49	4	$[3, 113, 111, 93]$
IL$_2$	128	93	4	$[3, 57, 15, 69]$
IL$_3$	128	113	4	$[3, 65, 123, 65]$
IL$_4$	512	61	16	$[8, 50, 107, 192, 258, 289, 454, 360,$ $376, 7, 316, 494, 173, 434, 292, 398]$

For constructing the final interleavers, the base interleavers $\{I_0, I_1, I_2, I_3\}$ were first lifted to a disorder degree $Q' = 8$ via (12) with the base permutation δ_{Base} (I_4 was not further lifted). Then, (9) and (10) are applied using the tuples listed in Table II, which refer to the permutation vectors δ_{LS} and δ_{PG}

Table II
Permutations for the construction of the set $\mathcal{C}$ in the case study given by their lexicographic index. The number i in the left column is given to identify the interleavers in Figures 6,7

i	IL$_0$ $(\delta_{LS}, \delta_{PG})$	IL$_1$ $(\delta_{LS}, \delta_{PG})$	IL$_2$ $(\delta_{LS}, \delta_{PG})$	IL$_3$ $(\delta_{LS}, \delta_{PG})$	IL$_4$ $(\delta_{LS}, \delta_{PG})$
0	0, 0[‡]	0, 0[‡]	0, 0[‡]	0, 0[‡]	0,0[‡]
1	0, 11170	1, 16703	2, 35093	13, 10321	0,14
2	0, 28119	2, 16703	10, 703	122, 20004	6,0
3	0, 28143	3, 16703	11, 36805	122, 23838	6,14
4	1, 39255	4, 16703	28, 714	126, 16490	6,122
5	11, 35478	5, 16703	28, 31628	843, 26649	8,126
6	48, 4477	6, 16703	121, 4598	843, 28126	8,132
7	48, 35481	7, 16703	122, 37016	843, 33489	48,50
8	48, 35791	8, 16703	124, 35772	1112, 722	86,176
9	50, 30438	9, 16703	148, 12475	1255, 33127	120,122
10	53, 4833	10, 16703	1459, 25906	1255, 33133	128,122
11	53, 4835	11, 16703	1465, 1184	1255, 33193	128,126
12	53, 4856	12, 16703	1465, 1244	1256, 1250	128,726
13	3385, 11041	13, 16703	1465, 1304	49, 16227	134,14
14	3385, 14810	14, 16703	1465, 2048	60, 9866	176,176
15	3385, 15066	15, 16703	1465, 2050	72, 3345	846,362

[‡] Identical to the base interleaver.

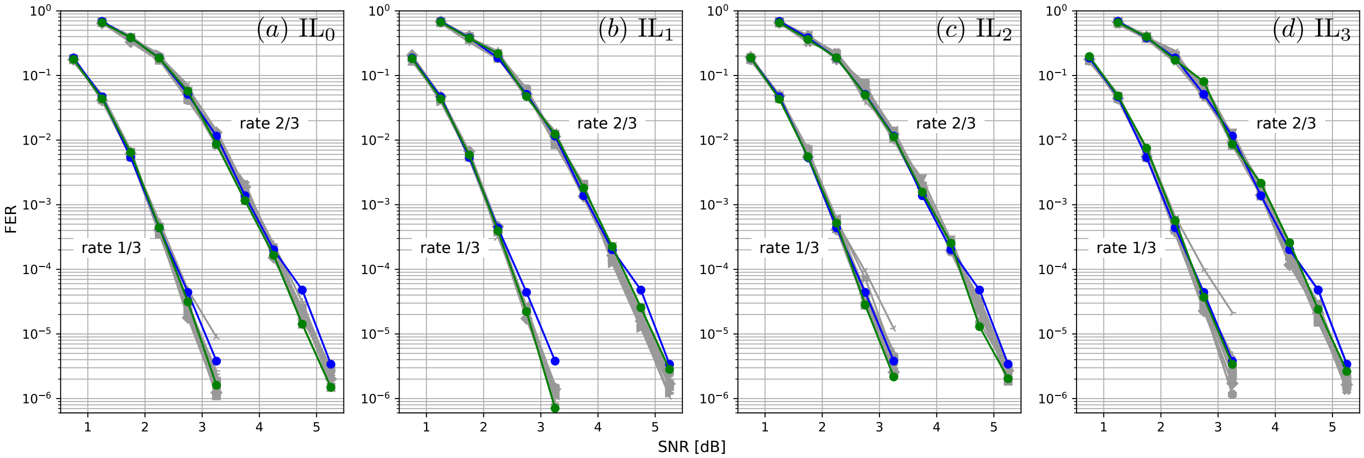

Figure 6. FER performance for $R = 1/3$, $R = 2/3$ of the interleavers with $K = 128$ from Table II (8 iterations, max-Log-MAP) in comparison with the corresponding LTE interleaver (blue). The interleavers corresponding to $i = 0$ are shown in green, those with $i = 1, \ldots, 15$ are shown in grey.

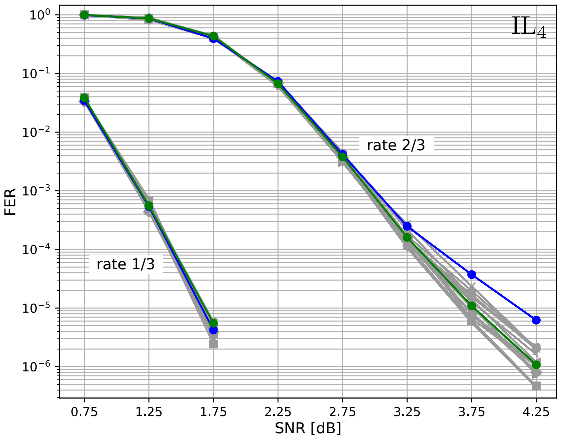

Figure 7. FER performance for $R = 1/3$, $R = 2/3$ of the interleavers with $K = 512$ from Table II (8 iterations, max-Log-MAP) in comparison with the corresponding LTE interleaver (blue). The interleavers corresponding to $i = 0$ are shown in green, those with $i = 1, \ldots, 15$ are shown in grey.

according to their *lexicographic index* [24] respective to the permutation length Q. The permutations $(\delta_{LS}, \delta_{PG}) = (0, 0)$ consequently preserve the base interleavers. Note again, that Q is different for $\{I_0, I_1, I_2, I_3\}$ and $\{I_5\}$ in Table II and therefore the lexicographic index 48, for example refers to $\delta_{LS} = (0, 1, 2, 5, 3, 4, 6, 7)$ for I_0, while it refers to $\delta_{LS} = (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 13, 11, 12, 14, 15)$ for I_5.

The FER simulation results corresponding to the final interleavers for rate 1/3 and 2/3 are shown in Figure 6 for the case where $K = 128$ bits and in Figure 7 for the case where $K = 512$ bits. For additional comparison results for the respective LTE QPP interleaver have been added.

In both cases, the obtained interleaver sets show close to, or better error correcting performance compared to the respective base interleavers as well as the LTE QPP interleavers.

For all resulting interleavers obtained through Tables I and II, a variable intra-layer shift (see (7)) can be applied. Then, for the 4·16 interleavers with $K = 128$, and the 16 interleavers with $K = 512$ it follows

$$\begin{aligned} \#\mathcal{C}_{128} &= 128 \cdot 4 \cdot 16 = 8192 \text{ , and} \\ \#\mathcal{C}_{512} &= 512 \cdot 16 = 8192. \end{aligned}$$

An eavesdropper, that uses the techniques discussed in [12] to recognize/detect the used Turbo Code, can therefore be expected to have a significant increase in N_{min}.

Table III illustrates the effect of the increased N_{min} for the case of $K = 512$ bits and one single code rate. In this table, which is partially reproduced from [12], values for N_{min} as well as runtimes of the algorithms for the blind detection of Turbo Code interleavers with $K = 512$ are given. Note that, the runtimes are to be seen as qualitative numbers, since [12] does not give details on the underlying compute platform.

Table III
N_{min} AND DETECTION TIME t FOR $K = 512$ (REPRODUCED FROM [12])

K	σ	N_{min}			runtime t in [s]		
		This Work‡	[12]	[11]	**This Work**†	[12]	[11]
512	0.6	376832	46	170	15319	1.87	11
512	0.8	909312	111	600	57344	7	37
512	1	$2.8 \cdot 10^6$	346	2800	139264	17	173
512	1.1	$5.4 \cdot 10^6$	660	3837	163840	20	357
512	1.3	$14.9 \cdot 10^6$	1820	29500	$36.6 \cdot 10^6$	64	4477

‡ $N = 8192 \cdot N_{[12]}$, † $t = 8192 \cdot t_{[12]}$.

Nonetheless, through $\#\mathcal{C}_{512} = 8192$, the runtime t for detecting all the interleavers is increases by several orders of magnitude and becomes impractical for high noise levels (large variance σ^2 values of the AWGN). Moreover, scaling this approach to larger frame size in the order of several thousands of bits and multiple different code rates or puncturing patterns also renders detection at lower noise levels impractical.

V. Discussion

It is worth noting, that the additional security obtained through employing larger sets of interleavers is indeed based on the assumption of imperfect knowledge of the Turbo Code parameters on the side of Eve. As such, it is not to be confused with the notion of information theoretic secrecy [4].

However, an additional random number generator can be employed to select a different interleaver from $\mathcal{C}$ for each transmitted block and thus making a recognition/detection necessary for each transmitted block. Even if the complete set of interleavers were known, a suitable random number generation scheme will, in this case, mandate a minimum number of observations to determine the internal state of the random number generator and/or its state transition function.

Furthermore, the permutations for generating $\mathcal{C}$ as well as the base interleavers can be periodically updated with new values, i.e. a completely new set $\mathcal{C}'$, in order to avoid detection of the interleaver parameters.

Note also, that in our case study, only a single puncturing pattern was used to achieve rate $2/3$. Permuting the protograph via (10) can be used to match optimally different puncturing patterns for the same code rate. Thus, our scheme can be efficiently extended to multiple puncturing patterns as in [14]–[16] and can therefore be seen as complementary. At the same time, puncturing the mother Turbo Code with a pseudo random pattern as in [14]–[16], can significantly penalize the error correcting performance of the decoder, since it can lead to long sequences without parities [25]. Moreover, the uniform and pseudo random interleavers used in [14] and [16] are inferior in terms of error correcting performance and implementation complexity in comparison to using ARP interleavers making our proposed scheme superior.

Last, since the proposed detection mitigation scheme is based on the well known ARP interleaver family, efficient integration into existing communication systems is possible with little overhead. This makes our proposal an attractive solution to increase the security of systems already in place.

VI. Conclusion

In this work, we presented permutation transformations on parameter sets defining ARP interleavers. With these, we were able to generate a large set of interleavers with similar error correcting performance starting from a small subset of base interleavers. We further demonstrated the feasibility with a case study on two sets of 8192 distinct interleavers each derived from only 4 and 1 base interleavers for information frame sizes $K = 128$ bits and $K = 512$ bits respectively.

It should be noted, that the proposed permutations for the protograph connections and layer starting positions as well as the lifting of the disorder degree are expected to be extremely useful in the construction of interleaver sets for other applications, for example to support flexible Turbo decoder hardware architectures with minimal overhead [26], since they allow to generate a large number of candidate interleavers with minimum effort and overhead.

Acknowledgment

This work was partially funded by the French National Research Agency TurboLEAP project (ANR-20-CE25-0007).

References

[1] Z. Dawy, W. Saad, A. Ghosh, J. G. Andrews, and E. Yaacoub. Toward massive machine type cellular communications. *IEEE Wireless Commun.*, 24(1):120–128, 2017.

[2] A. D. Wyner. The wire-tap channel. *The Bell System Technical Journal*, 54(8):1355–1387, 1975.

[3] L. H. Ozarow and A. D. Wyner. Wire-tap channel ii. *AT&T Bell Laboratories Technical Journal*, 63(10):2135–2157, 1984.

[4] Matthieu Bloch, Masahito Hayashi, and Andrew Thangaraj. Error-control coding for physical-layer secrecy. *Proceedings of the IEEE*, 103(10):1725–1746, 2015.

[5] Antoine Valembois. Detection and recognition of a binary linear code. *Discrete Applied Mathematics*, 111(1-2):199–218, 2001.

[6] J. Barbier, S. Guillaume, and S. Houcke. Algebraic approach for the reconstruction of linear and convolutional error correcting codes. *Int. Journ. of Applied Math. and Comp. Science*, 2(3):113–118, 2006.

[7] J. Barbier and E. Filiol. Overview of turbo-code reconstruction techniques. *IACR Cryptol. ePrint Arch.*, 2009:68, 2009.

[8] M. Cluzeau. Block code reconstruction using iterative decoding techniques. In *IEEE Int. Symp. on Inf. Theory*, pages 2269–2273, 2006.

[9] M. Cluzeau and J. Tillich. On the code reverse engineering problem. In *2008 IEEE Int. Symp. on Inf. Theory*, pages 634–638, 2008.

[10] M. Côte and N. Sendrier. Reconstruction of a turbo-code interleaver from noisy observation. In *IEEE Int. Symp. on Inf. Theory*, pages 2003–2007, 2010.

[11] M. Cluzeau, M. Finiasz, and J. Tillich. Methods for the reconstruction of parallel turbo codes. In *IEEE Int. Symp. on Inf. Theory*, pages 2008–2012, 2010.

[12] J. Tillich, A. Tixier, and N. Sendrier. Recovering the interleaver of an unknown turbo-code. In *IEEE Int. Symp. on Inf. Theory*, pages 2784–2788, 2014.

[13] A. Motamedi, M. Najafi, and N. Erami. Parallel secure turbo code for security enhancement in physical layer. In *2015 Signal Proc. and Intelligent Systems Conf. (SPIS)*, pages 179–184, 2015.

[14] A. Payandeh, M. Ahmadian, and M. R. Aref. Adaptive secure channel coding based on punctured turbo codes. *IEE Proceedings - Commun.*, 153(2):313–316, 2006.

[15] M. S. Daghighi, A. Payandeh, and M. R. Aref. Adaptive random puncturing based secure block turbo coding. In *5th Intern. Symp. on Telecommun.*, pages 216–220, 2010.

[16] Deyuan Chen and Can Zhang. Joint channel-security coding based on interleaver and puncturer in turbo code. In *2011 3rd Symp. on Web Society*, pages 153–157, 2011.

[17] R. Garzón-Bohórquez, C. Abdel Nour, and C. Douillard. Protograph-based interleavers for punctured turbo codes. *IEEE Trans. on Commun.*, 66(5):1833–1844, May 2018.

[18] J. Sun and O. Y. Takeshita. Interleavers for turbo codes using permutation polynomials over integer rings. *IEEE Trans. on Inf. Theory*, 51(1):101–119, January 2005.

[19] S. Crozier and P. Guinand. Distance Upper Bounds and True Minimum Distance Results for Turbo-Codes Designed with DRP Interleavers. In *Proc. 3rd Int. Symp. on Turbo Codes & Rel. Topics*, Brest, France, 2003.

[20] C. Berrou, Y. Saouter, C. Douillard, S. Kerouedan, and M. Jézéquel. Designing good permutations for turbo codes: towards a single model. In *Proc. IEEE Int. Conf. on Commun.*, pages 341–345, June 2004.

[21] R. Garzón Bohórquez, C. A. Nour, and C. Douillard. On the Equivalence of Interleavers for Turbo Codes. *IEEE Wireless Commun. Letters*, 4(1):58–61, Feb 2015.

[22] R. Garzón-Bohórquez, C. Abdel Nour, and C. Douillard. Improving Turbo Codes for 5G with parity puncture-constrained interleavers. In *9th Int. Symp. on Turbo Codes & Rel. Topics*, pages 151–155, 2016.

[23] C. Weiss, C. Bettstetter, and S. Riedel. Code construction and decoding of parallel concatenated tail-biting codes. *IEEE Trans. on Inf. Theory*, 47(1):366–386, 2001.

[24] S Skiena. Lexicographically ordered permutations. In *Implementing Discrete Mathematics, Combinatorics and Graph Theory with Mathematica*, pages 3–5. Addison-Wesley, 1990.

[25] E. Boutillon, J. Sánchez-Rojas, and C. Marchand. Compression of redundancy free trellis stages in turbo-decoder. *Electronics Letters*, 49(7):460–462, 2013.

[26] S. Weithoffer, O. Griebel, R. Klaimi, C. A. Nour, and N. Wehn. Advanced hardware architectures for turbo code decoding beyond 100 Gb/s. In *2020 IEEE Wireless Commun. and Netw. Conf. (WCNC)*, pages 1–6, 2020.