Randomized Turbo Codes for the Wiretap Channel

Alireza Nooraiepour and Tolga M. Duman

***Abstract*—We study application of parallel and serially concatenated convolutional codes known as turbo codes to the randomized encoding scheme introduced by Wyner for physical layer security. For this purpose, we first study how randomized convolutional codes can be constructed. Then, we use them as building blocks for developing randomized turbo codes. We also develop iterative low-complexity decoders corresponding to the randomized schemes introduced and evaluate the code performance. We demonstrate via several examples that the newly designed schemes can outperform other existing coding methods in the literature (e.g., punctured low density parity check (LDPC) and scrambled BCH codes) in terms of the resulting security gap.**

I. Introduction

The wiretap channel is introduced in [1] by Wyner as a model for studying secure communications. It consists of a transmitter (Alice), a legitimate receiver (Bob) and an eavesdropper (Eve) connected to the transmitter through the main and the eavesdropper's channels, respectively. Alice's purpose is to transmit a message $\mathbf{M}$ while ensuring that 1) the probability of decoding error for Bob goes to zero (as the reliability constraint), 2) the normalized mutual information $\frac{1}{n}I(\mathbf{M};\mathbf{Z}^n)$ goes to zero (as the security constraint) where $\mathbf{Z}^n$ denotes the eavesdropper's observation of length n. Wyner defined the notion of secrecy capacity as the highest transmission rate for which the security constraint is satisfied and proved that there exists a coding scheme which achieves the secrecy capacity using randomized encoding in the form of *coset-coding* at the transmitter as the main source of confusion at the eavesdropper.

Construction of practical codes for the wiretap channel has enjoyed an increasing attention in the recent years. For the case where the main channel is noiseless and the eavesdropper's channel is a binary erasure channel, the authors in [2] prove that using LDPC codes along with their duals can achieve the secrecy capacity. In [3], it is proved that polar codes can achieve the secrecy capacity when both the main and the eavesdropper's channels are modeled as binary symmetric channels. In addition, application of lattice codes to the wiretap channel is studied in [4]. We emphasize that all the aforementioned schemes use a form of coset-coding for the encoding process.

For the case of additive white Gaussian noise (AWGN) channels, the secrecy capacity equals to the difference between the capacities of the main and the eavesdropper's channels, and for it to be greater than zero, the signal to noise ratio (SNR) of the main channel must be larger than that of the eavesdropper's channel. The difference between the qualities of the main and eavesdropper's channels needed for achieving physical layer security is the *security gap*. The security gap is a valuable metric for secrecy which can be directly computed from the bit error rate (BER) performance of a code over a noisy channel. By denoting the BERs calculated through the main and eavesdropper's channels with P_{main} and P_{eve}, respectively, one can use an alternative set of reliability and security constraints as follows: $P_{main} \leq P_{main}^{max}$ (≈ 0) and $P_{eve} \geq P_{eve}^{min}$ (≈ 0.5) where P_{main}^{max} and P_{eve}^{min} represent the maximum and minimum desired BERs for Bob and Eve, respectively. Denoting the lowest SNR (in dB) which satisfies the reliability constraint by SNR_{main} and the largest SNR (in dB) which satisfies the security constraint by SNR_{eve}, the security gap is defined as $\text{SNR}_{\text{main}} - \text{SNR}_{\text{eve}}$.

Some practical coding schemes aiming at reducing the security gap have been proposed in [5–8]. Punctured LDPC codes are utilized in [5], while the authors in [6] demonstrate that using non-systematic codes obtained from scrambling information bits of a systematic code are quite effective to reduce the security gap. In [7], the authors apply different techniques including scrambling, concatenation, and hybrid automatic repeat-request (HARQ) to LDPC codes in order to further reduce the security gap. In addition, code concatenation based on polar and LDPC codes for the wiretap channel is studied in [8].

In this paper, we consider application of the turbo codes to the randomized encoding scheme. Turbo codes in the form of parallel and serial concatenation of convolutional codes have a near Shannon limit performance and consequently a very sharp slope in their BER versus SNR curves over a variety of channels, including AWGN channels [11, 12]. Hence, we consider them as potential candidates for the wiretap channel from a security gap perspective. With this motivation, in this paper, we first describe how randomized convolutional codes are constructed. Then, we obtain dual of turbo codes and explain how they enable us to construct an effective randomized coding scheme. We also propose low-complexity iterative decoders for the resulting randomized codes, and evaluate their performance in terms of the resulting security gaps.

The rest of the paper is organized as follows. The channel model is introduced in Section II. We present the main idea of the randomized encoding scheme in Section III. Then, we study how convolutional codes can be applied to this scheme in Section IV, and describe the corresponding decoder. Randomized turbo codes along with their corresponding iterative decoders are studied in Section V. Numerical examples are provided in Section VI, and finally, the paper is concluded in Section VII.

II. Channel Model

We consider an AWGN wiretap channel where both the main and eavesdropper's channels are expressed as

$$y = x + N \tag{1}$$

A. Nooraiepour was a graduate student with the Dep. of Elect. Eng., Bilkent University. He is now a PhD student at Rutgers university, NJ, US. T. M. Duman is with the Dep. of Elect. Eng., Bilkent University, TR-06800, Ankara, Turkey. emails: {nooraiepour, duman}@ee.bilkent.edu.tr. This work was supported by the Scientific and Technical Research Council of Turkey (TUBITAK) under the grant 113E223.

978-1-5090-5019-2/17/$31.00 ©2017 IEEE

where $x = (-1)^c$ is the binary phase-shift keying (BPSK) modulated version of the transmitted bit $c \in \{0, 1\}$. N represents the Gaussian noise with zero mean and variance $N_0/2$. We assume that the noise samples are independent and identically distributed (i.i.d.) across different uses of the channel. We have $E_b = 1/R$ where E_b is the energy per bit and R is the transmission rate. We define the SNR as E_b/N_0. We emphasize that the model in (1) is used for both the main and eavesdropper's channels (with different noise power levels).

III. Randomized Coding Scheme for Secrecy

A. Encoding

In conventional encoding used to provide reliability over a noisy channel, each message is mapped to a unique codeword. On the other hand, a randomized encoding scheme aims at confusing the eavesdropper by introducing enough randomness in the encoding process. Specifically, in the general scheme of Wyner [1], each message gets mapped to a unique coset of a certain code and codewords from this coset are transmitted uniformly randomly. Therefore, to transmit a k-bit message, 2^k cosets are needed. Assuming that each coset consists of 2^r codewords, in order to cover all the codewords in this setup, we need a linear code of dimension at least $k+r$ which we call the *big code*. We also refer to the coset corresponding to the all-zero message as the *small code* denoted by $\mathcal{C}$ with generators $\mathbf{g}_1, \mathbf{g}_2, ..., \mathbf{g}_r$.

A message denoted by *data bits* $\mathbf{s} = [s_1, s_2, ..., s_k]$ is mapped to the coset obtained by $s_1\mathbf{h}_1 + s_2\mathbf{h}_2 + ... + s_k\mathbf{h}_k + \mathcal{C}$ where $\mathbf{h}_i$'s are linearly independent n-tuples outside $\mathcal{C}$. Then the transmitted codeword through the channel is obtained by choosing a random codeword uniformly randomly in $\mathcal{C}$. As pointed out in [2], this can be accomplished by a random bit vector $\mathbf{v} = [v_1, v_2, ..., v_r]$ using

$$\mathbf{c} = s_1\mathbf{h}_1 + s_2\mathbf{h}_2 + ... + s_k\mathbf{h}_k + v_1\mathbf{g}_1 + v_2\mathbf{g}_2 + ... + v_r\mathbf{g}_r. \quad (2)$$

The randomized encoding scheme needs two sets of generators, one for the random bits and the other for the data bits. The vectors $\mathbf{g}_i$'s are determined by choosing a linear code as the small code $\mathcal{C}$. Then, determination of $\mathbf{h}_i$'s requires an exhaustive search which may not be practical. In [10], it is proved that one can use generators of the dual of the small code $\mathcal{C}^\perp$ as $\mathbf{h}_i$'s if $\mathcal{C}^\perp$ is not *pseudo-self-dual*[1].

B. Optimal Decoding Rule

Given a received noisy vector $\mathbf{y}$, the maximum a posteriori probability (MAP) decoder picks a coset index which maximizes the probability $P(C^i|\mathbf{y})$ [2] where C^i denotes the ith coset. Assuming that there are M cosets which represent the messages, and in each of them there are N codewords, the output of the optimal MAP decoder is

$$\hat{i} = \underset{i=1,2,...,M}{\operatorname{argmax}} P(C^i|\mathbf{y}). \quad (3)$$

[1]A linear code $\mathcal{C}(n, r)$ with generator matrix $\mathbf{G}$ is defined as *pseudo-self-dual* if $\mathbf{G}\mathbf{G}^T$ is rank-deficient [9].

[2]We denote probability with P and probability density with p throughout this paper.

Using Bayes' rule and the total probability theorem (assuming that the codewords in each coset have equal probabilities to be transmitted through the channel), we can write

$$P(C^i|\mathbf{y}) = \frac{p(\mathbf{y}|C^i)P(C^i)}{p(\mathbf{y})},$$
$$p(\mathbf{y}|C^i) = \sum_{j=1}^{N} p(\mathbf{y}|\mathbf{c}_{ji}, \mathcal{C}^i)P(\mathbf{c}_{ji}|\mathcal{C}^i) = \frac{1}{N}\sum_{j=1}^{N} p(\mathbf{y}|\mathbf{c}_{ji}, \mathcal{C}^i), \quad (4)$$

where $\mathbf{c}_{ji}$ denotes the jth codeword in the ith coset. Finally, for an AWGN channel and equiprobable cosets, the optimal MAP decoder has the form

$$\hat{i} = \underset{i=1,2,...,M}{\operatorname{argmax}} \sum_{j=1}^{N} e^{\frac{-\|\mathbf{y}-\mathbf{c}_{ji}\|^2}{N_0}}. \quad (5)$$

Note that for the main and eavesdropper's channel the noise variances are different, hence the resulting optimal decoding rules are different. We also note that for the optimal MAP decoding, one needs to go through all the codewords in all the cosets making implementation of the algorithm prohibitively complex for practical code dimensions.

IV. Randomized Convolutional Codes

We will use randomized convolutional codes as building blocks to construct the randomized turbo codes in Section V, hence we first discuss their generation. In order to build a randomized convolutional coding scheme, one can choose a convolutional code as the small code introduced in Section III-A (see [9] for the details). The dual of a convolutional code is also necessary for this purpose which is obtained in a systematic manner using the following result.

Definition 1: Reverse of a convolutional code $\mathcal{C}$ with polynomial generator $\mathbf{G}(D) = \mathbf{G}_0 + \mathbf{G}_1 D + ... + \mathbf{G}_m D^m$ is defined as the convolutional code $\widetilde{\mathcal{C}}$ with polynomial generator $\widetilde{\mathbf{G}}(D) = \mathbf{G}_m + \mathbf{G}_{m-1}D + ... + \mathbf{G}_0 D^m$ where m denotes the memory size.

Theorem 1 (Taken from [14]): Dual of a convolutional code $\mathcal{C}$ with the polynomial generator $\mathbf{G}(D)$ has a polynomial generator of the form $\widetilde{\mathbf{H}}(D)$ where $\mathbf{G}(D)(\mathbf{H}(D))^T = \mathbf{0}$.

A. Decoding of Randomized Convolutional Codes

When the Euclidean distances among the codewords in each coset are relatively large or when the SNR is sufficiently high, the summation in the optimal decoding rule in (5) is dominated by the terms which correspond to the codewords at the minimum Euclidean distance to the received vector $\mathbf{y}$. Therefore, as an approximate decoding approach, one can find the codeword at the minimum Euclidean distance to the given received noisy vector (referred to as the minimum distance decoder).

As described in Section III-A, the encoding process needs two convolutional codes whose trellises can be combined to form a trellis for the big code governing codewords obtained by (2), i.e., the codewords that are being sent through the channel. This "big" trellis can then enable us to find the minimum distance codeword to the output of the channel $\mathbf{y}$ via an appropriate use of the Viterbi algorithm.

B. Obtaining a Subset of Convolutional Codes

As discussed in Section III-A, the codewords in each coset represent a single message and are aimed at confusing the eavesdropper. If the main channel is noiseless, decoding process at the legitimate receiver is trivial, and we only want to confuse the eavesdropper. In this case, it is desirable to use as many codewords as possible in each coset. If the main channel is also noisy, then one should consider reducing the number of codewords in each coset in order to increase the error correction capabilities of the legitimate receiver. As discussed in Section III-A, the number of codewords in each coset is governed by the small code $\mathcal{C}(n,r)$ and equals 2^r assuming that the random bits are being encoded by the generators of the small code.

Let $\mathcal{C}$ be a convolutional code of rate a/b with the generator matrix $\mathbf{G}(D)$ with a rows. After finding the equivalent generator matrix $\mathbf{G}^{[k]}(D)$ to $\mathbf{G}(D)$ with rate ka/kb for $k = 2,3,\ldots$, one can obtain a subset of $\mathcal{C}$ by choosing different rows from ka available rows of $\mathbf{G}^{[k]}(D)$. Clearly, the resulting convolutional code has a smaller rate than $\mathcal{C}$ and improved error correction capabilities.

We now explain how one can obtain an equivalent generator matrix $\mathbf{G}^{[k]}(D)$ with rate $k/bk,\ k = 2,3,\ldots$ for a convolutional code with a generator matrix $\mathbf{G}(D)$ of rate $1/b$. The extension of the method to the general case (for a rate a/b code) is quite straightforward. $\mathbf{G}^{[k]}(D)$ accepts k input bits in each time slot, so the input bits u_i's are fed to the encoders in the following manner

$$\begin{array}{ccccc} \ldots & u_{i+3k-1} & u_{i+2k-1} & u_{i+k-1} & \rightarrow \quad \mathbf{g}_1 \\ \ldots & u_{i+3k-2} & u_{i+2k-2} & u_{i+k-2} & \rightarrow \quad \mathbf{g}_2 \\ & \vdots & \vdots & \vdots & \vdots \\ \ldots & u_{i+2k+1} & u_{i+k+1} & u_{i+1} & \rightarrow \quad \mathbf{g}_{k-1} \\ \ldots & u_{i+2k} & u_{i+k} & u_i & \rightarrow \quad \mathbf{g}_k \\ \ldots & D^2 & D & 1 & \end{array} \tag{6}$$

where "$\rightarrow\ \mathbf{g}_i$" means that the bits are being fed to a specific generator $\mathbf{g}_i$ (a row of $\mathbf{G}^{[k]}(D)$) and the last row denotes the delay associated with the input bits in each column. We denote the output of the encoder corresponding to $\mathbf{G}(D)$ to the input u_{i+f} by $\mathbf{v}_f$ whose elements are $v_{f,j}$ where $0 \le f \le k-1$ and $1 \le j \le b$. Furthermore, we consider the corresponding output of $\mathbf{G}^{[k]}(D)$ to the input vector $[u_i\ u_{i+1}\ \ldots u_{i+k-1}]$ as $\begin{bmatrix}\mathbf{o}_0 & \mathbf{o}_1 & \ldots & \mathbf{o}_{k-1}\end{bmatrix}$ whose elements, $\mathbf{o}_f$'s, are vectors each of which consists of b sequences, and each sequence is the sum of the delayed u_i's produced through the k generators within the structure in (6). $\mathbf{G}^{[k]}(D)$ and $\mathbf{G}(D)$ are equivalent if

$$\mathbf{v}_f = \mathbf{o}_{k-f-1},\ 0 \le f \le k-1 \tag{7}$$

where $\mathbf{v}_f = u_{i+f}\mathbf{G}(D)$ which is known since $\mathbf{G}(D)$ is given. We note that each element of $\mathbf{o}_i$ is produced by a column of $\mathbf{G}^{[k]}(D)$. Furthermore, (7) consists of bk equations which determine the desired generators, $\mathbf{g}_i$'s, $1 \le i \le k$ needed for the corresponding column of $\mathbf{G}^{[k]}(D)$.

[3]We denote convolutional codes in octal notation throughout the paper.

Example 1: Consider the code[3] $[561\ 753]$ of memory $m = 8$ and rate $1/2$, i.e.,

$$\mathbf{G}(D) = [1 + D^2 + D^3 + D^4 + D^8,\ 1 + D + D^2 + D^3 + D^5 + D^7 + D^8]. \tag{8}$$

Following the steps described above, we can obtain the equivalent generator matrix of $\mathbf{G}(D)$ with rate $4/8$ as

$$\mathbf{G}^{[4]}(D) = \begin{bmatrix} p(D) & 1+D^2 & 0 & 1+D & 1 & 1 & 1 & 1+D \\ D & D+D^2 & p(D) & 1+D^2 & 0 & 1+D & 1 & 1 \\ D & D & D & D+D^2 & p(D) & 1+D^2 & 0 & 1+D \\ 0 & D+D^2 & D & D & D & D+D^2 & p(D) & 1+D^2 \end{bmatrix} \tag{9}$$

where $p(D) = 1 + D + D^2$. To obtain a subset of $\mathcal{C}$, one can use any subset of the rows of $\mathbf{G}^{[4]}(D)$ as the generator matrix. We note that the resulting subset may have a smaller rate than the original code $\mathcal{C}$. For example, if we choose only one of the rows of $\mathbf{G}^{[4]}(D)$ as the generator matrix, the resulting code will have a rate of $1/8$. ■

C. Randomized Convolutional Code Design

Earlier in this section, we discussed how a small code and its dual can be used to form the big code. Since both the small code and its dual are assumed to be convolutional codes, the big code is also a convolutional code. Clearly, the minimum pairwise distance among the codewords in each coset with respect to a specific codeword is larger than (or equal to) that in the big code. For practical cases, the codewords in the big code which are at the minimum distance of c_{ij}, the ith codeword in the jth coset, do not belong to the jth coset. Therefore, assuming that a minimum distance decoder is being used, they are important sources of decoding errors. Hence, a design metric becomes the minimum pairwise distance among the codewords of the big code which controls the error correcting capability of the minimum distance decoder. In practice, one should choose this distance in a way that results in the smallest security gap.

If one uses a convolutional code $\mathcal{C}(n,r)$ (small code) to encode the random bits and its dual $\mathcal{C}^{\perp}(n,n-r)$ to encode the data bits, the big code consists of all the 2^n n-tuples (ignoring trellis termination to the all-zero state for the time being); a fact that results in the lowest possible minimum distance (of 1) for the big code. In this case, performance of the minimum distance decoder is poor from the legitimate receiver's point of view. Alternatively, one can use the approach described in Section IV-B to obtain a subset of $\mathcal{C}(n,r)$ denoted by $\mathcal{C}'(n,r')$ where $r' < r$. Now, using generators of $\mathcal{C}'$ and $\mathcal{C}^{\perp}$ to encode random and data bits, respectively, the big code will have $r' + n - r$ many generators which is less than n; hence, the resulting big code can achieve a larger minimum distance. We note that in either case the data transmission rate is $(n-r)/n$ since the data bits' encoder is the same.

Take the small code $\mathcal{C}$ as a convolutional code of rate $R = b/c$ with the *minimal-basic* generator matrix $\mathbf{G}(D)$ [14]. Equivalent generator matrices to $\mathbf{G}(D)$ which reproduce $\mathcal{C}$ are obtained by

$$\mathbf{G}_{2nd}(D) = \mathbf{T}(D)\mathbf{G}(D) \tag{10}$$

where $\mathbf{T}(D)$ is a $b \times b$ full rank matrix. Then, instead of working with $\mathbf{G}(D)$, one may use $\mathbf{G}_{2nd}(D)$ to obtain new subsets of

$\mathcal{C}$, and consequently new generators for the random bits (see Section IV-B). That is, different choices for $\mathbf{T}(D)$ result in different generators for the random bits, and hence they result in different sets of codewords in each coset and possibly different minimum distances for the big code.

Example 2: Let us choose the small code $\mathcal{C}$ as the convolutional code $[561\ 753]$ which is the same code given earlier in (8). Its dual $\mathcal{C}^\perp$ is the optimal convolutional code (in terms of the minimum distance) of memory 8 and rate $1/2$ with the generator $[657\ 435]$. If one uses the generators of $\mathcal{C}^\perp$ and the entire $\mathcal{C}$ to encode the data and random bits, respectively, the resulting big code will have a minimum distance of 2 (they do not cover all the n-tuples because of the trellis termination to zero state). However, if one uses the generators of $\mathcal{C}^\perp$ for the data bits, and $[D\ D\ D\ D+D^2\ p(D)\ 1+D^2\ 0\ 1+D]$ for the random bits which is a subset of $\mathcal{C}$ as discussed in Example 1, the big code will attain a minimum distance of 6.

We can improve the minimum distance even more by using (10), i.e., with

$$\mathbf{G}^{[4]}_{2nd}(D) = \mathbf{T}(D)\mathbf{G}^{[4]}(D) \tag{11}$$

where $\mathbf{G}^{[4]}(D)$ is the same as (9), and the 4×4 matrix $\mathbf{T}(D)$ is described by its polynomial inverse

$$\mathbf{T}^{-1}(D) = \begin{bmatrix} 1+D & D & D & 1+D \\ D & D^2+1 & 1 & D \\ D & D & 1+D & D \\ 1+D & 1 & D & D \end{bmatrix}. \tag{12}$$

After some straightforward algebra, one can calculate $\mathbf{G}^{[4]}_{2nd}(D)$ (which is 4×8) and obtain one of its rows as the 1×8 vector

$$\begin{aligned}[D^5+D^4+D^3 \quad & D^5+D^3+D^2 \quad D^4+D^3 \quad D^5+D \\ D^5+D^4+D^3+D^2+D+1 \quad & D^5+D^3+D^2+D+1 \\ & D^3+D^2 \quad D^3+D^2+1].\end{aligned} \tag{13}$$

Using $\mathcal{C}^\perp$ and (13), we obtain a big code with minimum distance 10. Here, it is clear that data bits are encoded with rate $1/2$ while the random bits' encoding rate is $1/8$. We note that the code $\mathcal{C}^\perp$ has a minimum distance of 12 which is an upper bound on the minimum distance of the big code. ■

V. Randomized Turbo Codes

A. Randomized Parallel Concatenated Convolutional Codes (RPCCCs)

Berrou *et al.* in [11] proposed a parallel concatenation of two recursive systematic convolutional codes (RSCs) separated by a K-bit interleaver. In parallel concatenated convolutional codes (PCCCs), the first RSC code (denoted as RSC1) encodes the information sequence $\mathbf{u}$ of length K, and produces the parity bits $\mathbf{p}$. The interleaved version of $\mathbf{u}$ denoted by $\mathbf{u}_\Pi$ gets encoded by the second RSC code (denoted as RSC2) and parity bits $\mathbf{q}$ are produced. We denote the generator of each RSC code by $G(D) = [1\ \frac{g_2(D)}{g_1(D)}]$.

Dual of PCCCs can be obtained by substituting each component RSC code with its dual. Using Theorem 1, a generator for the dual of $G(D)$ is obtained as $G^\perp(D) = [\frac{\widetilde{g}_2(D)}{\widetilde{g}_1(D)}\ \ 1]$. We note that $\mathbf{u}_\Pi$ is not transmitted in the original construction of PCCCs [11], however, here we transmit $\mathbf{u}_\Pi$ as well in order to be able to obtain the dual of RSC2. We denote the input sequence corresponding to the dual of a PCCC by $\mathbf{u}'$, and the resulting parity bits by $\mathbf{p}'$ and $\mathbf{q}'$.

We use a PCCC for encoding the random bits, and its dual to encode the data bits. Clearly, in the proposed scheme, the number of data and random bits, which equal to the number of cosets and the number of codewords in each coset, respectively, is $n/4$ where n is the codeword length (ignoring trellis termination). We denote the resulting codeword of the PCCC by $\mathbf{c}_1 = [c_1, c_2, \ldots, c_K]$ where $c_k \triangleq [u_k, p_k, u_{\Pi,k}, q_k]$, and the resulting codeword of its dual with $\mathbf{c}_2 = [c'_1, c'_2, \ldots, c'_K]$ where $c'_k \triangleq [p'_k, u'_k, q'_k, u'_{\Pi,k}]$. The codeword transmitted through the channel is the modulo-2 sum of $\mathbf{c}_1$ and $\mathbf{c}_2$, i.e., $\mathbf{c} = \mathbf{c}_1 + \mathbf{c}_2$.

B. Randomized Serially Concatenated Convolutional Codes (RSCCCs)

A serially concatenated convolutional code (SCCC) consists of two RSC codes as proposed in [12] as illustrated in Fig. 1 where the RSC codes are taken to be the same (though this is not necessary). The outer RSC code encodes the information sequence $\mathbf{u}$, and the resulting codeword gets permuted and fed to the inner code to generate the final codeword.

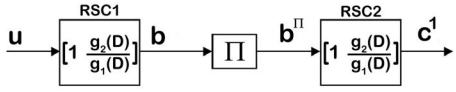

Fig. 1: The encoder for an SCCC.

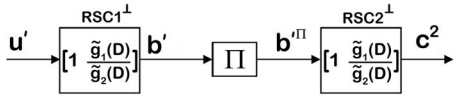

Fig. 2: The encoder for the dual of the SCCC in Fig. 1.

Similar to the PCCC case, (a subset of) the dual of an SCCC is obtained by replacing each constituent RSC encoder with its dual as in Fig. 2. We note that $G^\perp(D) = [1\ \frac{\tilde{g}_1(D)}{\tilde{g}_2(D)}]$ is also a dual for $G(D) = [1\ \frac{g_2(D)}{g_1(D)}]$. Then, we are able to use one of the encoders in Figs. 1 and 2 to encode the random bits, and the other one to encode the data bits in the randomized encoding scheme. Assuming $\mathbf{c}^1 = [v_1, q_1, v_2, q_2, \ldots, v_K, q_K]$ and $\mathbf{c}^2 = [v'_1, q'_1, v'_2, q'_2, \ldots, v'_K, q'_K]$, the transmitted codeword is the modulo-2 sum of these two codewords, i.e.,

$$\mathbf{c} = \mathbf{c}^1 + \mathbf{c}^2 = [v_1+v'_1, q_1+q'_1, v_2+v'_2, q_2+q'_2, \ldots, v_K+v'_K, q_K+q'_K]. \tag{14}$$

C. Decoding

We now describe the decoding for the proposed randomized turbo coding schemes. First, we emphasize the importance of the *big trellis* in the decoding process by taking RPCCCs as an instance (similar arguments can be made for RSCCCs as well). For this case, the big trellis obtained by combining the trellises corresponding to RSC1 and RSC1$^\perp$ governs the modulo-2 sum $[u_k+p'_k, p_k+u'_k]$, and can be used to provide soft information about the pair of bits (u_k, u'_k). Similarly, the big trellis corresponding to RSC2 and RSC2$^\perp$ controls the modulo-2 sum $[u_{\Pi,k}+q'_k, q_k+u'_{\Pi,k}]$.

The optimal MAP decoding rule for RPCCCs and RSCCCs is the same as (5) which is not practical. Therefore, we propose a sub-optimal decoder which jointly decodes the random and data bits (i.e., u_i's and u_i''s) by generalizing the decoders introduced in [11] and [12] for the case of PCCCs and SCCCs, respectively. For this purpose, the MAP decoding criterion is given by

$$(\hat{u}_l, \hat{u}_l') = \underset{(u_l, u_l')}{\operatorname{argmax}} \; P\big((u_l, u_l')|\mathbf{y}\big) \tag{15}$$

where $\mathbf{y}$ is the received signal and $(u_l, u_l') \in \{00, 01, 10, 11\}$. The posterior probabilities $P\big((u_l, u_l')|\mathbf{y}\big)$ are computed using

$$P\big((u_l = k, u_l' = j)|\mathbf{y}\big) = \sum_{\mathcal{U}_{kj}} p(s_{l-1} = s', s_l = s, \mathbf{y}) \tag{16}$$

where $(kj) \in \{00, 01, 10, 11\}$, and $\mathcal{U}_{kj}$ is set of pairs (s', s) for the state transitions $(s_{l-1} = s') \rightarrow (s_l = s)$ whose corresponding input labels are kj. Such probabilities can be computed efficiently using the BCJR algorithm [11].

Case 1 (Randomized PCCCs): Fig. 3 illustrates the proposed iterative decoder for RPCCCs. M_{12}^e and M_{21}^e denote the logarithmic extrinsic information on the pair of bits (u_l, u_l') which are being exchanged between the two constituent decoders and are of the form $\big[\log(P_{00}^e)\ \log(P_{01}^e)\ \log(P_{10}^e)\ \log(P_{11}^e)\big]$ where P_{kj}^e denotes the extrinsic probability that $(u_l, u_l') = (k, j)$ satisfying $P_{00}^e + P_{01}^e + P_{10}^e + P_{11}^e = 1$. The initial value for either of M_{12}^e and M_{21}^e is taken as $\big[\log(0.25)\ \log(0.25)\ \log(0.25)\ \log(0.25)\big]$.

The logarithmic branch metric $\tilde{\gamma}_l(s, s')$ used in the log-domain BCJR algorithm to compute functions $\tilde{\alpha}_l(s)$ and $\tilde{\beta}_l(s)$ (see [13] for the notation) is obtained in a straightforward manner as

$$\tilde{\gamma}_l(s, s') = \log\big(P_{u_l u_l'}^e\big) - \frac{\|y_l - c_l\|^2}{N_0} \tag{17}$$

for an AWGN channel where c_l denotes the output label for each branch in the trellis at time l and y_l is the corresponding observation. We note that for computing the outgoing extrinsic information at each decoder, the information that is already known at the destination should not be used, in other words, $\log\big(P_{u_l u_l'}^e\big)$ must be removed from (17).

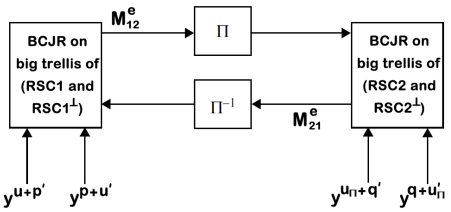

Fig. 3: The iterative decoder for a RPCCC when the PCCC encodes the random bits and its dual encodes the data bits.

Case 2 (Randomized SCCCs): The iterative decoder for an RSCCC works based on the description in Fig. 4. Two constituent decoders exchange their information on the pair of bits (b_i, b_i') introduced in Figs. 1 and 2. Specifically, M_{12} and M_{21} are of the form $\big[\log(P_{00}^e)\ \log(P_{01}^e)\ \log(P_{10}^e)\ \log(P_{11}^e)\big]$ where P_{kj}^e denotes the extrinsic probability that $(b_i, b_i') = (k, j)$. The decoder corresponding to RSC2 and its dual works similar to the ones in the case of RPCCCs described earlier. However, the decoder for RSC1 and its dual only receives extrinsic information (i.e., there is no observation from the channel). The logarithmic branch metric used to obtain $\tilde{\alpha}_l(s)$ and $\tilde{\beta}_l(s)$ is computed as

$$\tilde{\gamma}_i(s, s') = \log\big(P_{b_{2i-1} b_{2i-1}'}^e\big) + \log\big(P_{b_{2i} b_{2i}'}^e\big), \tag{18}$$

and the extrinsic information on bits b_{2i-1}'s can be computed using

$$\tilde{\gamma}_i(s, s') = \log\big(P_{b_{2i} b_{2i}'}^e\big) \tag{19}$$

while the extrinsic information on b_{2i}'s is computed by setting

$$\tilde{\gamma}_i(s, s') = \log\big(P_{b_{2i-1} b_{2i-1}'}^e\big) \tag{20}$$

in the BCJR algorithm.

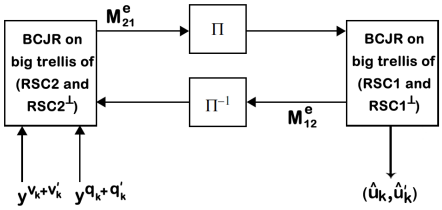

Fig. 4: The iterative decoder for an RSCCC where one of the encoders in Figs. 1 and 2 encodes the random bits and the other encodes the data bits.

VI. Numerical examples

In this section, we evaluate the performance of the newly proposed randomized coding schemes in terms of their security gaps. We denote the length of the codewords, the number of data bits and the number of random bits by n, k and r, respectively (see Section III-A). As a first example, we consider two randomized convolutional codes. The first code is obtained by using $[657\ 435]$ for encoding the data bits, and the generator of rate $1/8$ and memory 4

$$\begin{matrix}\big[D^3+1 & D^4+1 & D^4+D^3+D^2 & D^4+D^3+D+1 \\ D^3+D^2+D & D^3 & D^3+D^2 & D^3+D^2+1\big],\end{matrix} \tag{21}$$

which is a subset of dual of $[657\ 435]$, is used to encode the random bits. The minimum distance of the code is 8. Following the discussion in Section IV-C, we have obtained a second randomized convolutional code (of minimum distance 10) by using the following encoder of rate $1/8$ and memory 5 for the random bits (which is another subset of the dual of $[657\ 435]$)

$$\begin{matrix}\big[D^4+D^3+1 & D^3+1 & D^5+D^4+D^2+D+1 & D^4 \\ D^5+D^4+D^3+D^2 & D^4+D^2+1 & D^5+D^4+D^3+D^2+D+1 & \\ & & D^5+D^4+D^2+D\big]. & \end{matrix} \tag{22}$$

For both code examples we have provided enough zero bits at the end of the input sequences in order to force the trellises to the all-zero state.

The RSC code used in all the randomized turbo code examples is $[1\ 7/5]$ whose dual is either $[1\ 5/7]$ or $[7/5\ 1]$; the latter is used for the case of RPCCCs and the former for the case of RSCCCs. The number of iterations is set to 10 for the decoders in the Figs. 3 and 4. The interleaver which has been used for these examples is the S-random interleaver introduced in [15]. Furthermore, the trellises in the RPCCCs and the inner trellises in RSCCCs are terminated to the all-zero state.

The resulting BERs at the eavesdropper for the randomized convolutional coding scheme are shown in Fig. 5 as a function of security gap, which demonstrates that the randomized convolutional codes can result in lower security gaps in comparison with the punctured LDPC codes for high BERs at Eve [5].

Fig. 6 illustrates the performance of randomized turbo codes. It shows that the RSCCC example results in lower security gaps in comparison to the RPCCC one. By applying scrambling to the RSCCC of length $n = 8004$, a security gap of about 0.9 dB for $P_{eve}^{min} \approx 0.5$ (which is the worst scenario for the eavesdropper) is obtained. The notion of perfect scrambling in [6] ensures that a single (or more) bit error(s) in the decoded word is sufficient to make on average half of the bits in that word erroneous after descrambling. This result also illustrates that for similar code lengths and similar code rates, the scrambled RSCCC results in a 0.1 dB lower security gap in comparison with the scrambled BCH scheme (which, to the best of our knowledge, is the best existing scheme in the literature as far as the security gap is concerned). Another advantage of the newly proposed randomized coding schemes is shown in the Fig. 7 where it is clear that one can achieve a 0.6 dB improvement in the waterfall region by using a scrambled RSCCC compared to a scrambled BCH code.

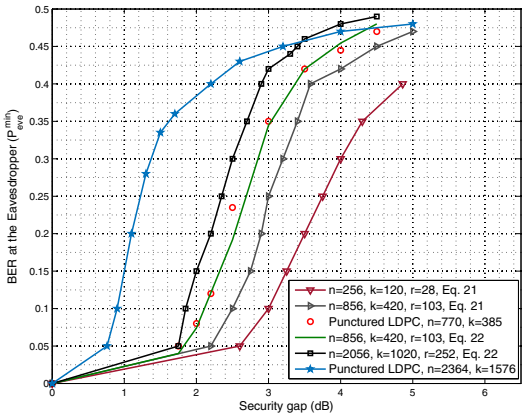

Fig. 5: P_{eve}^{min} versus the security gap (at $P_{main}^{max} \approx 10^{-5}$) for a randomized convolutional code when $[657\ 435]$ encodes the data bits for three different codeword lengths and two different random bit encoders in (21) and (22).

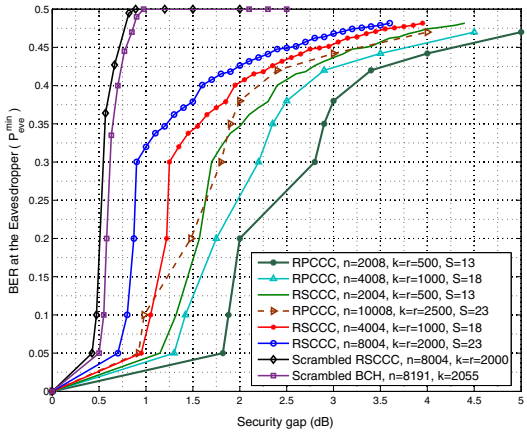

Fig. 6: P_{eve}^{min} versus the security gap (at $P_{main}^{max} \approx 10^{-5}$) for the RSCCC and RPCCC examples of different lengths.

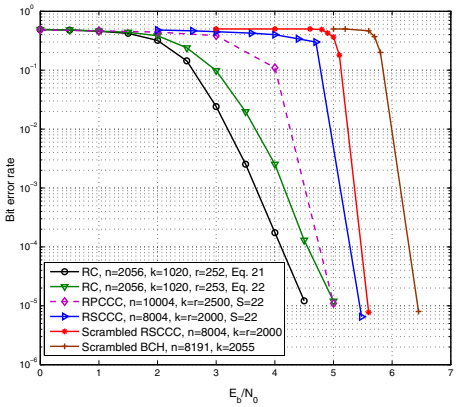

Fig. 7: BER curves corresponding to some of the codes in Figs. 5 and 6. RCCs stands for randomized convolutional codes scheme.

VII. Conclusions

We have proposed ways of applying turbo codes to the coset coding method used to provide secure transmission over wiretap channels. First, we have studied how randomized convolutional codes can be constructed. Armed with this result, we have developed RPCCC and RSCCC schemes and proposed corresponding iterative decoders. We have illustrated our findings via numerical examples, which demonstrate that the randomized convolutional codes can outperform punctured LDPC codes for high BERs at Eve. Finally, we have demonstrated through examples that using scrambling idea along with the RSCCCs can improve upon the security gap obtained from scrambling a BCH code.

References

[1] A. Wyner, "The wire-tap channel," *Bell System Technical Journal*, vol. 54, no. 8, pp. 1355-1387, Oct. 1975.

[2] A. Thangaraj, S. Dihidar, A. Calderbank, S. McLaughlin and J. Merolla, "Applications of LDPC codes to the wiretap channel," *IEEE Trans. Inf. Theory*, vol. 53, no. 8, pp. 2933-2945, Aug. 2007.

[3] H. Mahdavifar and A. Vardy, "Achieving the secrecy capacity of wiretap channels using polar codes," *IEEE Trans. Inf. Theory*, vol. 57, no. 10, pp. 6428-6443, Oct. 2011.

[4] F. Oggier, P. Sole and J. Belfiore, "Lattice codes for the wiretap Gaussian channel: construction and analysis," *IEEE Trans. Inf. Theory*, vol. 62, no. 10, pp. 5690-5707, Oct. 2016.

[5] D. Klinc, J. Ha, S. McLaughlin, J. Barros and B.-J. Kwak, "LDPC codes for the Gaussian wiretap channel," *IEEE Trans. Inf. Forensics Security*, vol. 6, no. 3, pp. 532-540, Sep. 2011.

[6] M. Baldi, M. Bianchi, and F. Chiaraluce, "Non-systematic codes for physical layer security," in *Proc. IEEE Information Theory Workshop (ITW 2010)*, Dublin, Ireland, Aug. 2010.

[7] M. Baldi, F. Bambozzi, and F. Chiaraluce, "Coding with scrambling, concatenation, and HARQ for the AWGN wiretap channel: a security gap analysis," *IEEE Trans. Inf. Forensics Security*, vol. 7, no. 3, pp. 883-894, Jun. 2012.

[8] Y. Zhang, A. Liu, C. Gong, G. Yang, and S. Yang, "Polar-LDPC concatenated coding for the AWGN wiretap channel," *IEEE Commun. Lett.*, vol. 18, no. 10, pp. 1683-1686, Oct. 2014.

[9] A. Nooraiepour and T. M. Duman, "Randomized convolutional codes for the wiretap channel," *IEEE Trans. Commun.*, vol. 65, no. 8, pp. 3442-3452, Aug. 2017.

[10] A. Nooraiepour, "Randomized convolutional and concatenated codes for the wiretap channel," M.S. Thesis, Dept. Elect. Eng., Bilkent Univ., Ankara, Turkey, 2016.

[11] C. Berrou, A. Glavieux, and P. Thitimajshima, "Near shannon limit error-correcting coding and decoding: turbo codes," *Proc. 1993 Int. Conf. on Communications*, pp. 1064-1070, May 1993.

[12] S. Benedetto, D. Divsalar, G. Montorsi, and F. Pollara, "Serial concatenation of interleaved codes: performance analysis, design, and iterative decoding," *IEEE Trans. Inf. Theory*, vol. 44, pp. 909-926, May 1998.

[13] W. Ryan and S. Lin, *Channel Codes, Classical and Modern.* Cambridge University Press, Cambridge, UK, 2009.

[14] R. Johannesson and K. Zigangirov, *Fundamentals of Convolutional Coding.* Piscataway, NJ: IEEE Press, 1999.

[15] D. Divsalar and F. Pollara, "Multiple turbo codes for deep-space communications," *Telecommun. Data Acquisition Rep., Jet Propulsion Lab.*, Pasadena, CA, USA, pp. 66-77, May 1995.