FLEXIBLE DESIGN OF FINITE BLOCKLENGTH WIRETAP CODES BY AUTOENCODERS

Karl-Ludwig Besser, Carsten R. Janda, Pin-Hsun Lin, Eduard A. Jorswieck
{*karl-ludwig.besser, carsten.janda, pin-hsun.lin, eduard.jorswieck*}*@tu-dresden.de*

Communications Theory, Communication Lab, Technische Universität Dresden

ABSTRACT

With an increasing number of wireless devices, the risk of being eavesdropped increases as well. From information theory, it is well known that wiretap codes can asymptotically achieve vanishing decoding error probability at the legitimate receiver while also achieving vanishing leakage to eavesdroppers. However, under finite blocklength, there exists a tradeoff among different parameters of the transmission. In this work, we propose a flexible wiretap code design for Gaussian wiretap channels under finite blocklength by neural network autoencoders. We show that the proposed scheme has higher flexibility in terms of the error rate and leakage tradeoff, compared to the traditional codes.

Index Terms— autoencoder, neural networks, finite blocklength, wiretap code, physical layer security

1. INTRODUCTION

For communication scenarios with a potential eavesdropper, it is known that a communication with simultaneously vanishing error probability and information leakage is asymptotically possible, where the maximum achievable secrecy rate is called the secrecy capacity, [1]–[3]. However, for finite blocklength wiretap codes there exists a tradeoff between error probability at the legitimate receiver, the amount of leaked information to the eavesdropper, and the transmission rate, [4]. In this work, a machine learning (ML) approach based on a feed-forward neural network (FF-NN) autoencoder [5] is proposed, which allows a flexible design for both the wiretap code encoder and decoder. In particular, the tradeoff between the reliability and leakage can be adjusted for fixed code parameters, which may not be easily achieved by traditional error-correcting codes (ECCs), [6]. Usually, for given blocklength n and number of information bits k, there exists a fixed design scheme, e.g. polar wiretap codes [7], which yields one particular codebook with only one operating point in the reliability/leakage space.

The application of ML algorithms in communications has gained increasing attention, recently. ML has already been used to replace various parts in traditional communication systems. In [8], the authors implemented convolutional neural networks for demodulation. Besides the application as a demodulator, neural networks (NNs) have been used to successfully decode channel codes, e.g. polar codes and random codes [9], linear codes [10], convolutional codes [11], and polar wiretap codes [12]. A broader approach of replacing multiple parts in the transmission chain has been taken in [13], where the authors used an autoencoder to learn both the coding and the decoding functions.

In this work, a framework based on NN autoencoders is developed, which allows a flexible design of finite blocklength wiretap codes. First, we can easily change the operating point with respect to the tradeoff between block error rate (BLER) and leakage by varying the weightings in the objective function of the autoencoder. Therefore, we can easily attain a higher flexibility than the traditional ECC, which has only one operating point on the tradeoff plane given the code parameters. Second, we can achieve a performance close to that of the polar wiretap code by only one layer NNs for the encoder and decoder part.

Notation: Random variables (of all dimensions) are denoted in boldface capital letter, e.g. $\boldsymbol{M}$. Sets like codebooks are denoted with calligraphic letters, e.g. $\mathcal{C}$. The finite field containing 2^n elements is denoted as $\mathbb{F}_2^n$ and the real numbers are written as $\mathbb{R}$. The identity matrix of size n is denoted as I_n. A Gaussian distribution with mean μ and covariance matrix Σ is denoted as $\mathcal{N}(\mu, \Sigma)$. The probability of event A is denoted as $\Pr(A)$. The logarithm is with base 2.

2. SYSTEM MODEL

The model is shown in Fig. 1. Alice wants to transmit a secret message $\boldsymbol{M}$ from a set $\mathcal{M}$ of possible messages to the legitimate user Bob. The communication is monitored over a second channel by a passive eavesdropper Eve. The main channel between Alice and Bob and the eavesdropper's channel between Alice and Eve are assumed to be independent additive white Gaussian noise (AWGN) channels with noises $\boldsymbol{N}_\mathrm{B}$ and $\boldsymbol{N}_\mathrm{E}$, respectively, with different noise powers such that the wiretap channel is degraded, i.e. $\sigma_\mathrm{E}^2 > \sigma_\mathrm{B}^2$. Bob and Eve receive the signals $\boldsymbol{Y}$ and $\boldsymbol{Z}$, respectively. Alice applies a wiretap code to map the messages $\boldsymbol{M} \in \mathbb{F}_2^k$ into codewords $\boldsymbol{X} \in \mathbb{R}^n$. The codebook is denoted as $\mathcal{C}$. Note that here the domain of the codewords is different to the one used in common ECCs. The codes found by the autoencoder consist of vectors of length n of real numbers which might be non-uniform and asymmetric, which is different to those from the traditional encoding and modulation schemes as can be seen in Table 1.

A measure to quantify the secrecy of the transmission is the leakage $I(\boldsymbol{M}; \boldsymbol{Z})$, which is the mutual information

This work is supported in part by DFG (JO 801/23-1), DFG (LI 2886/2-1), and BMBF FastCloud 03ZZ0517A and FastSecure 03ZZ0522A.

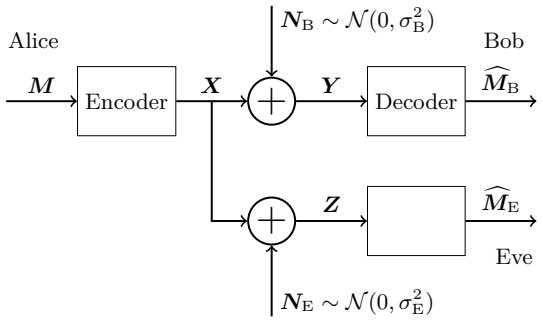

Fig. 1. System model of the wiretap scenario with AWGN channels.

between the secret message $\boldsymbol{M}$ and the received signal $\boldsymbol{Z}$ at the eavesdropper, [14]. It can be calculated as

$$I(\boldsymbol{M}; \boldsymbol{Z}) = h(\boldsymbol{Z}) - h(\boldsymbol{Z}|\boldsymbol{M}), \tag{1}$$

where h denotes the differential entropy. To achieve the strong secrecy, a binning structure [14] is used to construct the wiretap code, i.e. r additional random bits are used to map one secret message to multiple possible codewords. The encoder function g is given as

$$g : \mathcal{M} \times \tilde{\mathcal{M}} \to \mathcal{C} \ , \tag{2}$$

where the messages and random bits are from the sets $\mathcal{M} = \mathbb{F}_2^k$ and $\tilde{\mathcal{M}} = \mathbb{F}_2^r$, respectively. The codewords are set as the channel input without further prefixing.

Since Eve receives a noisy version of an n-dimensional codeword, the distribution of $\boldsymbol{Z}$ is a Gaussian mixture (GM)

$$\boldsymbol{Z} \sim \sum_{\mu_i \in \mathcal{C}} c_i \mathcal{N}(\mu_i, \sigma_{\mathrm{E}}^2 I_n), \tag{3}$$

where the means $\mu_i \in \mathcal{C}$ and the weights $c_i \triangleq \Pr(\mu_i \text{ is selected})$. In the case of uniformly distributed messages, $c_i = |\mathcal{C}|^{-1}$ for all i. The conditional distribution of $\boldsymbol{Z}$ given the transmitted message $\boldsymbol{M} = m$ is also an GM, where the means of the components are all possible codewords for the individual message m,

$$\boldsymbol{Z}|_{\boldsymbol{M}=m} \sim \sum_{\mu_i \in \mathcal{C}_m} \mathcal{N}(\mu_i, \sigma_{\mathrm{E}}^2 I_n). \tag{4}$$

The subset of $\mathcal{C}$ containing only the codewords for message m is denoted as $\mathcal{C}_m$.

Since there is no known closed-form expression for calculating the differential entropy of a GM, upper and lower bounds from [15] are used to bound (1). With a slight abuse of notation, the upper and lower bounds of the differential entropy of an n-dimensional GM with K components are given as

$$\tilde{h}_{\mathrm{UB}}(\mathrm{GM}) = \frac{n}{2} - \sum_{i=1}^{K} c_i \log \sum_{j=1}^{K} c_j p_j(\mu_i) \,, \tag{5}$$

$$\tilde{h}_{\mathrm{LB}}(\mathrm{GM}) = \frac{n}{2} + \frac{n}{2} \log \frac{1}{4} - \sum_{i=1}^{K} c_i \log \sum_{j=1} K c_j \tilde{p}_{j,0.5}(\mu_i) \,, \tag{6}$$

where p_j and $\tilde{p}_{j,\alpha}$ denote the Gaussian probability density functions with mean μ_j and covariance matrices Σ and $\Sigma/\left(\alpha(1-\alpha)\right)$, respectively.

After substituting Eq. (3) and (4) into Eq. (5) and (6), respectively, we can express the leakage upper bound as

$$\begin{aligned} I(\boldsymbol{M}; \boldsymbol{Z}) &\leq \tilde{h}_{\mathrm{UB}}(\boldsymbol{Z}) - \tilde{h}_{\mathrm{LB}}(\boldsymbol{Z}|\boldsymbol{M}) \qquad (7) \\ &= k + n - \frac{1}{2^{k+r}} \sum_{\mu_i \in \mathcal{C}} \log \sum_{\mu_j \in \mathcal{C}} p_{\mu_j}(\mu_i) \\ &\quad + \frac{1}{2^{k+r}} \sum_{m \in \mathcal{M}} \sum_{\mu_i \in \mathcal{C}_m} \log \sum_{\mu_j \in \mathcal{C}_m} \tilde{p}_{\mu_j, 0.5}(\mu_i). \qquad (8) \end{aligned}$$

Note that we use an upper bound of the leakage as the worst-case-scenario for training our system. Therefore, the performance of our scheme can be better in practice.

The reliability at Bob is measured by the BLER which is defined as the probability that the decoded message at Bob $\widehat{\boldsymbol{M}}_{\mathrm{B}}$ is not identical to the message m transmitted by Alice,

$$\mathrm{BLER} = \Pr\left(\widehat{\boldsymbol{M}}_{\mathrm{B}} \neq m | \boldsymbol{M} = m\right). \tag{9}$$

The goal of the code design is to minimize both Eq. (8) and (9) simultaneously. This is a multi-objective programming problem (MOP) which can be solved by the *scalarization approach*, i.e. minimizing the weighted sum [16],

$$\min_{\mathcal{C}} w_1 \mathrm{BLER} + w_2 I(\boldsymbol{M}; \boldsymbol{Z}), \tag{10}$$

with the positive objective weights $w_1, w_2 \in \mathbb{R}_{>0}$.

3. AUTOENCODER IMPLEMENTATION

The wiretap system model shown in Fig. 1 is implemented by an autoencoder shown in Fig. 2. The autoencoder has two vector inputs and two vector outputs. The first input $\boldsymbol{M}$ contains k message bits $m_i \in \mathbb{F}_2^k$. The second input $\tilde{\boldsymbol{M}}$ contains r uniformly random bits which are used to introduce the confusion messages to the wiretap code. The outputs of the network are the estimated message bits $\hat{m}_i$ at Bob and the leakage $I(\boldsymbol{M}; \boldsymbol{Z})$ at Eve. The NN of the autoencoder consists of an encoder part at Alice and a decoder part at Bob. Both parts are deep FF-NNs [5] with the rectified linear unit (ReLU) activation function on all layers but the last one. The final output layer at Bob uses the *sigmoid* activation function to ensure an output between 0 and 1. On top of the last layer of the NN at Alice, a normalization layer is added which centers and scales the data, to be zero mean and unit variance. The two parts are separated by a noise layer which adds Gaussian noise with a variance σ_{B}^2 to model the AWGN channel between Alice and Bob. The second output of the system, which is used for training, is at Eve. At this output, the found codewords from the encoder output at Alice and the noise variance of the wiretap channel σ_{E} are used to estimate the leakage.

For training the network, a multi-objective loss function according to Eq. (10) is considered. The goal of the network is to minimize both the error probability at Bob as well as the leakage at Eve simultaneously to ensure a secure transmission. At Bob's output, the mean squared error (MSE) is used as a loss function L_{B}. As shown in [9], minimizing this loss

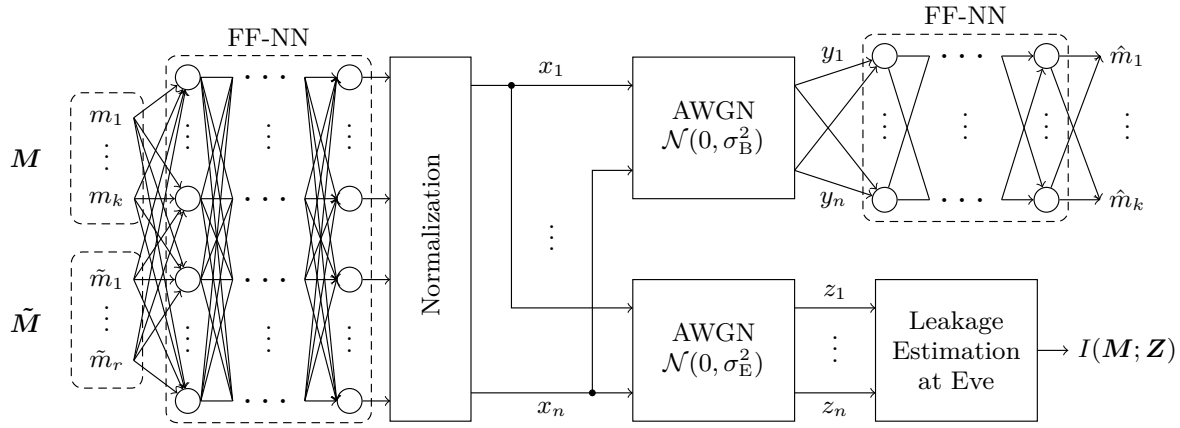

Fig. 2. Implementation structure of the autoencoder system for designing wiretap codes. Both the encoder and decoder at Alice and Bob are implemented by FF-NN between which is a noise layer. As the second loss function, the upper bound of the leakage at the eavesdropper is used.

function is also reducing the decoding errors. During training, the power of the noise which is added in the channel to Bob is fixed as a constant value to reduce the decoding error probability, cf. [9], [12]. At Eve's output, the mutual information $I(\boldsymbol{M}; \boldsymbol{Z})$ between the messages $\boldsymbol{M}$ and the received symbols $\boldsymbol{Z}$ is applied as a loss L_{E}. Since the analytical derivation of the leakage is not tractable, it is approximated as shown in Eq. (7). In this work, the final loss function L to be minimized is a weighted sum of the individual losses,

$$L = w_B L_{\mathrm{B}} + w_E' L_{\mathrm{E}} \tag{11}$$

$$= w_B L_{\mathrm{B}} + \frac{w_E}{k} L_{\mathrm{E}} \, . \tag{12}$$

Since the leakage L_{E} can be as high as k, it is normalized by k to have both losses in L in the same range.

After training the system, the found wiretap codebook (including modulation) is given as the output of the normalization layer at Alice. Due to the random input bits, the code has a binning structure, i.e. there exist multiple codewords for each secure message.

3.1. Energy Calculation

The output of the training process is an encoder and decoder system implicitly including the modulation and demodulation steps. The codewords are modulated, centered, and scaled to unit variance in a normalization layer after the NN at Alice to fulfill the power constraint. The symbol energy E_S is calculated as

$$E_S = \frac{1}{n 2^{k+r}} \sum_{\mu_i \in \mathcal{C}} \|\mu_i\|^2 \, , \tag{13}$$

where $\|\cdot\|$ denotes the Euclidean norm. The signal-to-noise ratio (SNR) in the following is defined as E_S/N_0.

3.2. Simulation Tools

The following tools are used for the simulations. The NNs are implemented in Python3 using Keras [17] with TensorFlow [18] as backend. All calculations were performed on a off-the-shelf computer. The source code of the implementation for all simulations can be found at [19].

4. SIMULATION RESULTS

The code parameters for the simulations are fixed by designing a polar wiretap code with blocklength 16 according to [7] for the AWGN channels to Bob and Eve with an E_b/N_0 of 0 dB and -5 dB, respectively. It yields a $(16, 4, 3)$ code, where the entries denote the blocklength n, number of secure bits k and number of random bits r, respectively. The E_S/N_0 values follow $(k + r)E_b/(nN_0)$. In the following, the SNR values are assumed to indicate E_S/N_0, since the modulation scheme of the codes found by the autoencoders does not follow any classical one and may change between different symbols. Thus, it is not easy to convert the symbol energy to the energy per information bit.

The short blocklength of 16 has been chosen due to the high complexity of the NNs. Usually, the training time complexity is linear in the size of the training set [20], which in this case is the size of the codebook $|\mathcal{C}|$. Since the codebook grows exponentially with the number of secure and random bits, i.e. $|\mathcal{C}| = 2^{k+r}$, the training time complexity also grows exponentially. In contrast, the complexity of a polar decoder is $\mathcal{O}(n \log n)$ [21].

The code parameters are applied to the autoencoder implementation described in Section 3 to generate wiretap codes for the Gaussian wiretap channel. The varied hyperparameters are the structures of the NNs of the autoencoder, the variance of the added training noise, and the weight combination of the different loss functions. The weights are chosen such that $w_B, w_E \geq 0$ and $w_B + w_E = 1$. The detailed settings of the different autoencoders can be found in Table 2. The NN layers are described by a tuple, where each entry represents the number of nodes in the respective layer. The used training algorithm is Adam [22] and the number of training epochs is set to 10^4. After training, the system is tested with 10^6 messages from which the BLER at Bob is determined. Additionally, the leakage is approximated for the found codebook using Monte Carlo simulations based on the implementation from [15], [23]. The results for different autoencoders with 5 dB training SNR and a comparison to a polar wiretap code can be found in Fig. 3. The curves show the individual results for different combinations of the weights, w_B from 0.75

$$c_1 = \begin{pmatrix} 0.709 & 0.877 & -0.775 & -0.775 & 1.463 & 2.173 & -0.775 & -0.554 & -0.121 & -0.775 & -0.775 & -0.775 & -0.775 & 0.129 & -0.775 & 1.522 \\ 0.811 & 0.936 & -0.916 & -0.524 & 1.396 & 2.007 & -0.824 & -0.292 & 0.092 & -1.031 & -0.735 & -0.625 & -0.805 & 0.249 & -1.210 & 1.472 \\ 0.776 & 0.902 & -0.938 & -0.614 & 1.417 & 2.084 & -0.856 & -0.301 & 0.026 & -0.938 & -0.790 & -0.737 & -0.820 & 0.246 & -0.938 & 1.480 \\ 0.806 & 0.909 & -0.883 & -0.462 & 1.364 & 2.003 & -0.757 & -0.248 & 0.080 & -0.957 & -0.690 & -0.580 & -0.736 & 0.237 & -1.533 & 1.447 \\ 0.700 & 0.852 & -0.868 & -0.695 & 1.439 & 2.109 & -0.868 & -0.289 & -0.036 & -0.868 & -0.832 & -0.730 & -0.858 & 0.286 & -0.868 & 1.526 \\ 0.745 & 0.870 & -0.875 & -0.520 & 1.385 & 2.025 & -0.778 & -0.225 & 0.036 & -0.947 & -0.713 & -0.560 & -0.755 & 0.281 & -1.461 & 1.492 \\ 0.716 & 0.840 & -0.987 & -0.582 & 1.406 & 2.092 & -0.779 & -0.209 & -0.014 & -0.979 & -0.738 & -0.638 & -0.740 & 0.294 & -1.182 & 1.500 \\ 0.738 & 0.842 & -0.839 & -0.457 & 1.346 & 2.008 & -0.711 & -0.184 & 0.027 & -0.873 & -0.666 & -0.515 & -0.686 & 0.266 & -1.755 & 1.458 \end{pmatrix}$$

Table 1. Found codewords for message 1 by AE1 with 5 dB training SNR and weights $w_{\mathrm{B}} = 0.9999$ and $w_{\mathrm{E}} = 0.0001$.

to 0.9999 in 20 logarithmically spaced steps. Additionally, the point $w_B = 1$ is evaluated. Since there exists a tradeoff for finite blocklength wiretap codes among blocklength, rate, reliability and secrecy [4], the variation of the weights allows a flexible operating point of the system for fixed code parameters. One could set $w_B = 1$ to design a regular point-to-point code without any secrecy constraints or increase w_E to reduce the leakage, at the cost of reducing the reliability. This tradeoff between leaked information and error rate is clearly visible in Fig. 3. Our ultimate goal is to approach the Pareto boundary of the design space, namely reliability and secrecy. It can be seen that all tested autoencoders perform slightly worse than the polar wiretap code. Note that the performance is better, if the result is closer to the origin, i.e. zero errors at Bob and zero leakage to the eavesdropper. The polar wiretap code achieves a BLER of around 26.4 % and a leakage around 1.33 bits. At a similar BLER of around 26.2 %, the first tested autoencoder (AE1 with 5 dB training SNR) achieves a leakage around 1.46 bits. Note that this is the smallest possible network structure, which has only one layer of size n at Alice and one layer of size k at Bob. In all simulations, the second evaluated autoencoder (AE2) achieves similar results to the first one.

The results for different training SNRs all lie almost on the same line, but in different segments. This means that some hyperparameter settings can achieve a similar operating points but at different weight combinations, while some points are only achievable for particular hyperparameters. The lowest BLER found for the evaluated settings is AE2 with -5 dB training SNR at around 8.4 %. Even though the performance of the code found by the autoencoder is no better than the polar wiretap code for that specific operating point, we have the advantage of the flexibility to tradeoff between the BLER and leakage. Furthermore, since we only use at most two layers for the NN, it may be possible to find a deeper NN to outperform the polar wiretap code, which is our ongoing work.

In Table 1 we show the codewords for message 1 ($\mathcal{C}_1$) found by AE1 with 5 dB training SNR and weights $w_{\mathrm{B}} = 0.9999$ and $w_{\mathrm{E}} = 0.0001$. We can find that the alphabet is over the real numbers which is different to the binary polar wiretap code. More specifically, the alphabets are neither symmetric nor uniform on the quadrature plane.

Name	Encoder Layers	Decoder Layers	Training SNR
AE1	(16)	(4)	-5, 2, 5, and 10 dB
AE2	(128, 64, 16)	(128, 64, 4)	-5, 2, 5, and 10 dB

Table 2. The evaluated autoencoders for a $(16, 4, 3)$ wiretap code.

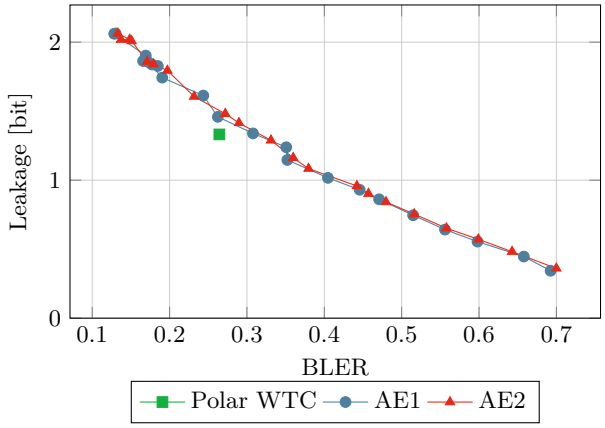

Fig. 3. Comparison of the designed wiretap codes by different autoencoder structures with 5 dB training SNRs to a polar wiretap code. The target code parameters are: $n = 16$, $k = 4$, $r = 3$.

5. CONCLUSION

In this work, we showed that an autoencoder with a minimal structure can provide us the flexibility of varying the operating point on the leakage and BLER plane to design both wiretap encoders and decoders, which cannot be done by traditional ECC. In particular, a multi-objective loss function including the reliability and total leakage has been formulated. Numerical results show the influence of the weighting parameters, the NN structure and the training SNR on the performance of the communication system.

A comparison to a polar wiretap code showed that under finite blocklength the found wiretap codes designed by ML can perform close to the former for a specific operating point. Improved results where the NNs outperform the polar wiretap code can be found in [24]. It should be noted, that there is a major difference between the codes found by the autoencoder and the polar wiretap code. Namely, the used polar wiretap code is binary with a binary phase-shift keying modulation, while the NNs operate on the real numbers. It will be very challenging to implement such a real-valued coding and modulation scheme on a real hardware system. Due to this major difference between the polar code and autoencoder codes, a comparison of the codes designed by the autoencoders with theoretical boundaries would be preferable. Such a framework can be found in [4], where the total variation distance is considered as a metric for the leaked information. This comparison will be part of our future work.

References

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

[2] I. Csiszár and J. Körner, "Broadcast channels with confidential messages," *IEEE Transactions on Information Theory*, vol. 24, no. 3, pp. 339–348, May 1978.

[3] M. Bloch and N. Laneman, "Strong secrecy from channel resolvability," *IEEE Transactions on Information Theory*, vol. 59, no. 12, pp. 8077–8098, Dec. 2013.

[4] W. Yang, R. F. Schaefer, and H. V. Poor, "Finite-blocklength bounds for wiretap channels," in *2016 IEEE International Symposium on Information Theory (ISIT)*, IEEE, Jul. 2016, pp. 3087–3091. arXiv: 1601.06055 [cs.IT].

[5] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[6] S. Lin and D. J. Costello, *Error Control Coding*, 2nd. Prentice Hall, 2004.

[7] H. Mahdavifar and A. Vardy, "Achieving the Secrecy Capacity of Wiretap Channels Using Polar Codes," *IEEE Transactions on Information Theory*, vol. 57, no. 10, pp. 6428–6443, Oct. 2011.

[8] T. J. O'Shea, J. Corgan, and T. C. Clancy, *Convolutional Radio Modulation Recognition Networks*, Feb. 2016. arXiv: 1602.04105 [cs.LG].

[9] T. Gruber, S. Cammerer, J. Hoydis, and S. ten Brink, "On deep learning-based channel decoding," in *2017 51st Annual Conference on Information Sciences and Systems (CISS)*, IEEE, Mar. 2017. arXiv: 1701.07738 [cs.IT].

[10] E. Nachmani, Y. Be'ery, and D. Burshtein, "Learning to decode linear codes using deep learning," in *2016 54th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, IEEE, Sep. 2016, pp. 341–346, ISBN: 978-1-5090-4550-1. arXiv: 1607.04793 [cs.IT].

[11] H. Kim, Y. Jiang, R. B. Rana, S. Kannan, S. Oh, and P. Viswanath, "Communication algorithms via deep learning," in *International Conference on Learning Representations*, 2018.

[12] K.-L. Besser, P.-H. Lin, C. R. Janda, and E. A. Jorswieck, "Machine learning assisted wiretapping," in *2018 Asilomar Conference on Signals, Systems, and Computers*, 2018, will appear.

[13] T. J. O'Shea, K. Karra, and T. C. Clancy, "Learning to communicate: Channel auto-encoders, domain specific regularizers, and attention," in *2016 IEEE International Symposium on Signal Processing and Information Technology (ISSPIT)*, IEEE, Dec. 2016, pp. 223–228. arXiv: 1608.06409 [cs.LG].

[14] M. Bloch and J. Barros, *Physical-Layer Security*. Cambridge: Cambridge University Press, 2011, ISBN: 9780511977985.

[15] A. Kolchinsky and B. Tracey, "Estimating Mixture Entropy with Pairwise Distances," *Entropy*, vol. 19, no. 7, Jul. 2017.

[16] E. Björnson, E. A. Jorswieck, M. Debbah, and B. Ottersten, "Multiobjective Signal Processing Optimization: The way to balance conflicting metrics in 5G systems," *IEEE Signal Processing Magazine*, vol. 31, no. 6, pp. 14–23, Nov. 2014. arXiv: 1406.2871 [cs.IT].

[17] F. Chollet *et al.* (2015). Keras, [Online]. Available: https://github.com/keras-team/keras.

[18] M. Abadi *et al.* (2015). TensorFlow: Large-scale machine learning on heterogeneous systems, [Online]. Available: https://www.tensorflow.org/.

[19] K.-L. Besser. (2018). Autoencoders for flexible wiretap code design in Python, [Online]. Available: https://gitlab.com/klb2/autoencoder-wiretap.

[20] M. Naumov, *Parallel Complexity of Forward and Backward Propagation*, Dec. 2017. arXiv: 1712.06577 [cs.LG].

[21] E. Arıkan, "Channel Polarization: A Method for Constructing Capacity-Achieving Codes for Symmetric Binary-Input Memoryless Channels," *IEEE Transactions on Information Theory*, vol. 55, no. 7, pp. 3051–3073, Jul. 2009.

[22] D. P. Kingma and J. Ba, *Adam: A Method for Stochastic Optimization*, Dec. 2014. arXiv: 1412.6980 [cs.LG].

[23] B. Tracey. (2017). Package for estimating the entropy of a mixture distribution, [Online]. Available: https://github.com/btracey/mixent.

[24] K.-L. Besser, P.-H. Lin, C. R. Janda, and E. A. Jorswieck, "Wiretap code design by neural network autoencoders," *IEEE Transactions on Information Forensics and Security*, 2019, Submitted.