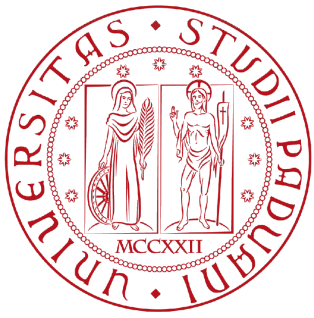

University of Padua
Department of Information Engineering

Information Security Report
Laboratory Session 2

Implementation of random binning encoding and secrecy rate evaluation

Authors:
Luca Badin
Gabriella Margarita Buosi Larez

Teachers:
Nicola Laurenti
Francesco Ardizzon

29th November 2020

Solution

Our solution to Laboratory 2 was written entirely in MATLAB, and it is composed of several source files:

- `wiretap.m` contains the implementation of the uniform error wiretap channel of Task 1, and `task1_simulate_wiretap.m` was used to test and prove conditional independence and uniformity;
- `rbe.m` contains the implementation of the probabilistic binning encoder of Task 2;
- `rbd.m` contains the implementation of the deterministic binning decoder of Task 3;
- Files prefixed with `task4_` were used to carry out the various analyses required by Task 4;
- `wiretap_bsc.m` contains the implementation of the wiretap binary symmetric channel of Task 5, and `task5_connection.m` was used to test it;
- Files prefixed with `task6_` were used to carry out the three analyses required by task 6;
- `marginals.m` contains simple rowwise and columnwise additions required by both Task 4 and 6.

Task 1: Implement the uniform error channel

The objective of Task 1 was to implement a wiretap channel with uniform error probabilities, subject to the following constraints:

- the input and output alphabets are the set of 7-bit binary strings;
- the legitimate channel introduces at most 1 binary error per word;
- the eavesdropper channel introduces at most 3 binary error per word;

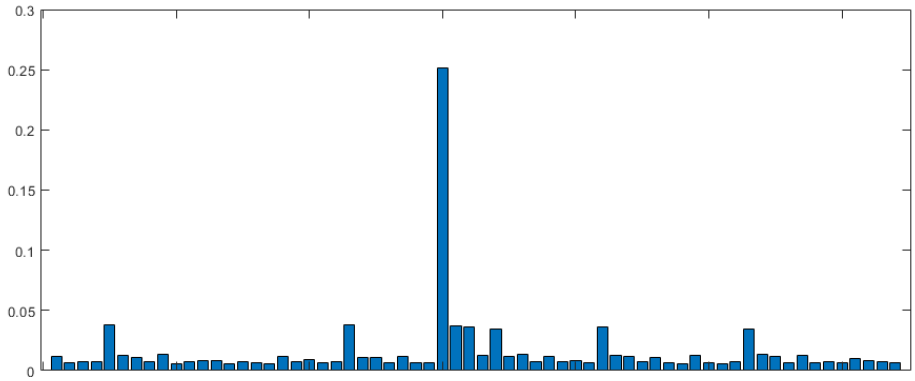

Figure 1: Conditional probability mass distribution

$$p_{z|x}(\cdot|1001000)$$

Observe how all values corresponding to the same number of errors are roughly equiprobable.

- both channel outputs are conditionally uniform and independent of each other, given their common input.

It was achieved through the code in `wiretap.m`; to generate the alteration in either channel, function `wiretap` generates a random "error mask" of the same length as the input strings, whose bits are all zeroes, except for the number of bits specified by the problem constraints. The error masks were then XORed with the input string to obtain the outputs of each channel. By running 10^4 channel realizations with the same binary word input x, it can be observed that outputs are uniform and independent.

Task 2: Implement the random binning encoder

Task 2 required the implementation of a (probabilistic) random binning encoder. The encoder's specifics are:

- the message space is the set of 3-bit binary strings;
- the code is a (7,4) Hamming code (meaning codewords are composed of 7 bits, of which 4 actually carrying data), composed by the following codewords:
  $X' = \{0000000, 1000110, 0100101, 0010011, 0001111, 1100011, 1010101, 1001001, 0110110, 0101010, 0011100, 1110000, 1101100, 1011010, 0111001, 1111111\}$
- the message space is the set of 3-bit words, $M = 0, 1^3$

- the bin $T_{x|u}(d)$ associated to each input $d \in M$ is made of 2 codewords: the one having $[0, d]$ as its 4-bit prefix, and the binary complement of the that codeword (e.g., $T_{x|u}(100) = 0100101, 1011010$)
- the codeword x is chosen randomly and uniformly within the bin associated to the message u

For its implementation, a function `rbe(m)` was built. The function takes as input a binary string m of size 3 and outputs x as the result of the encoding process. The inputted value is converted into a string and used to randomly select the row of the channel matrix to be XORed with.

Task 3: Implement the random binning decoder

Task 3 asks for the implementation of the legitimate decoder, so that it accepts an input $y \in \mathcal{Y}$ and produces the corresponding output $\hat{u} \in \mathcal{M}$.
For this, a deterministic legitimate decoder $D : \mathcal{Y} \mapsto \mathcal{M}$ is to be considered, which:

- looks at the first bit of $\hat{x}$ and identifies the transmitted message $\hat{u}$ as either the bits 2-4 in $\hat{x}$, or their complement
- identifies the transmitted codeword by the minimum Hamming distance criterion $\hat{x} = \arg\min_{a \in \mathcal{X}'} \|a \oplus y\|_H$

For the implementation of the random binning decoder, the function `rbd(m)` was created.It takes as input the message y, output of the legitimate channel, and applies the Hamming distance decoding criterion to get an estimate of the transmitted message. By cascading the functions that implement encoder, legitimate channel, and decoder, it is easy to verify that, because of the Hamming code's property of being systematic, it is able do detect at most two-bit errors and correct one-bit errors. Since the legitimate channel introduces at most a single error per word, the Hamming encoder-decoder pair is able to correctly detect and correct said error every time.

Task 4: Verify perfect secrecy

Task 4 requests to prove the fact that the encoder achieves perfect secrecy by showing that the eavesdropper's channel output z is independent of the secret message u.
In order to this, for each of the 8 possible values the three-bit message u can take, 10^4 realizations of the encoder + eavesdropper channel chain were run. From the graphical results, it can be observed that the output z seen by the eavesdropper is almost identical for every input u, independently of its value. The channel acts according to its input x, not on that of the encoder.

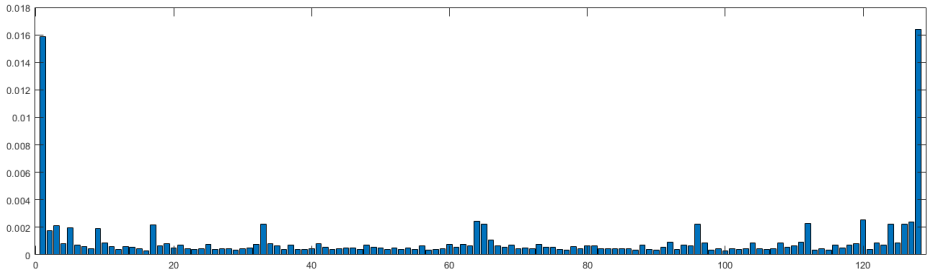

Figure 2: Task 4: Conditional PMD with d = [0 0 0]

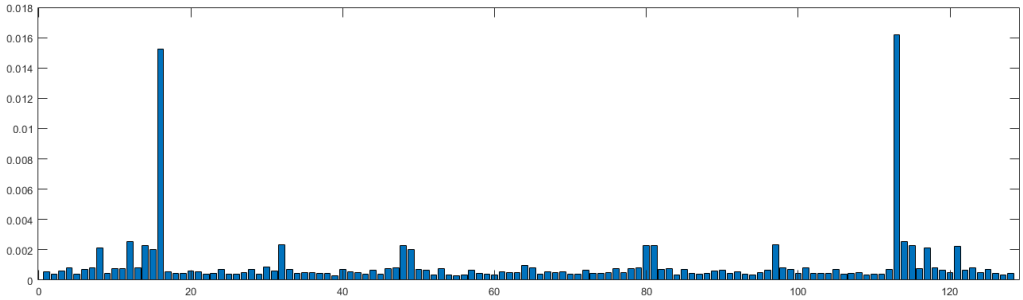

Figure 3: Task 4: Conditional PMD with d = [0 0 1]

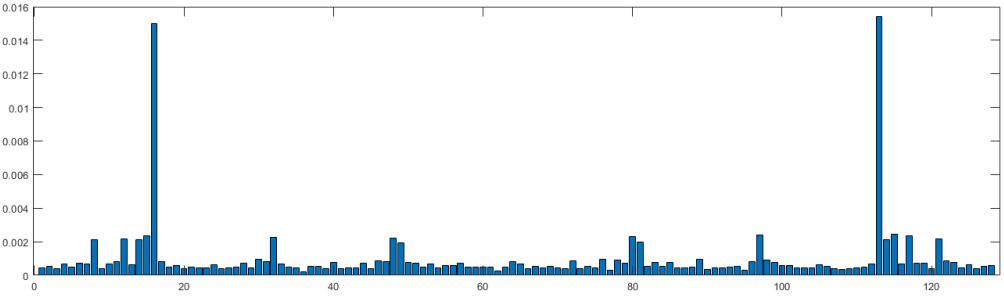

Figure 4: Task 4: Conditional PMD with d = [0 1 0]

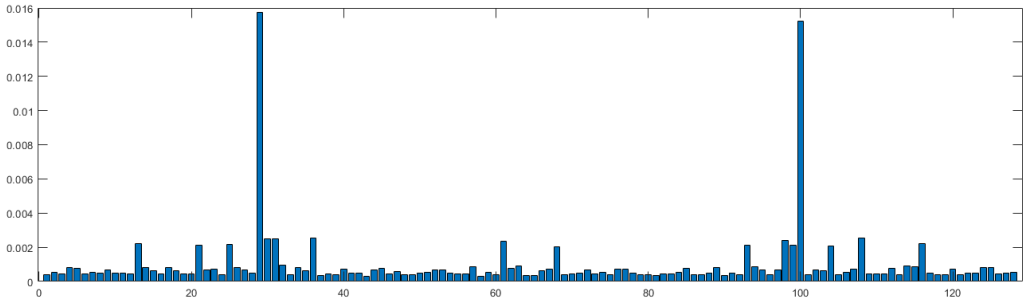

Figure 5: Task 4: Conditional PMD with d = [0 1 1]

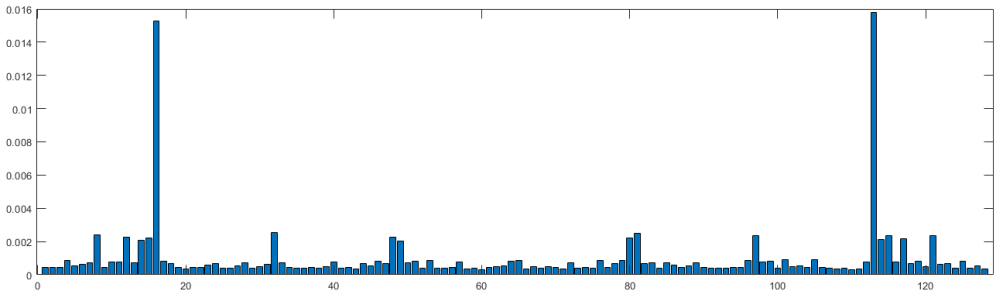

Figure 6: Task 4: Conditional PMD with d = [1 0 0]

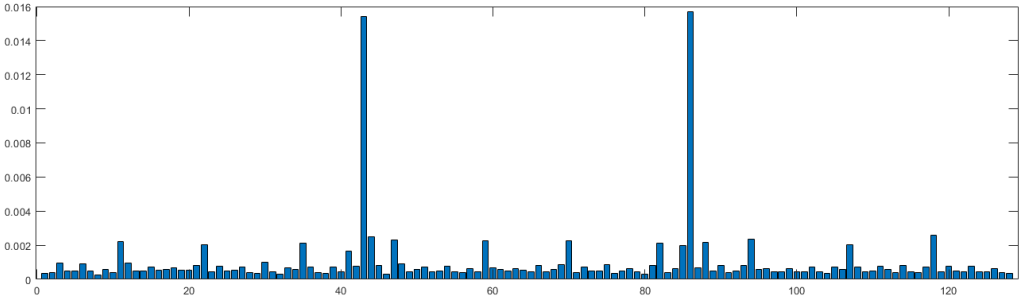

Figure 7: Task 4: Conditional PMD with d = [1 0 1]

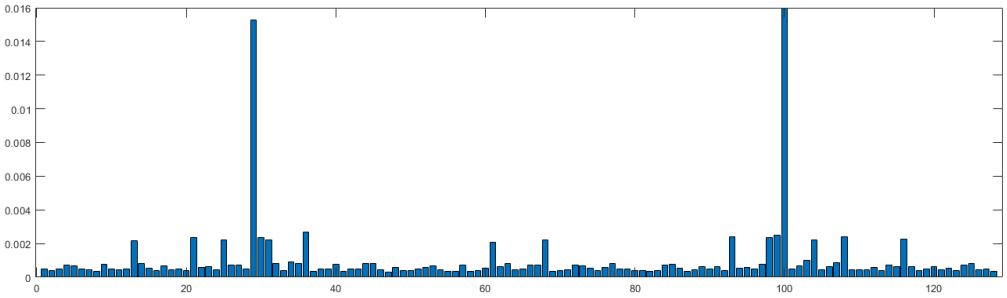

Figure 8: Task 4: Conditional PMD with d = [1 1 0]

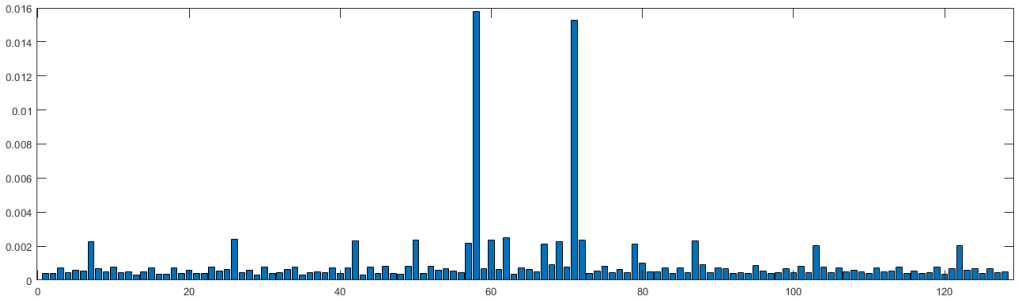

Figure 9: Task 4: Conditional PMD with d = [1 1 1]

Considerations and remarks

By considering the uniform channel as a simpler version of the more general memoryless wiretap channel, the secrecy capacity can be computed and used to analyze the number of secret message bits that can be transmitted.

$$C_s = C_{AB} - C_{AE} = h_2(\delta) - h_2(\varepsilon)$$

For the legitimate channel, the error rate is $\varepsilon = \frac{1}{2}\frac{1}{7} = 0.07$, while for the eavesdropper channel the error rate is $\delta = \frac{1}{4}\frac{3}{7} + \frac{1}{4}\frac{2}{7} + \frac{1}{4}\frac{1}{7} + \frac{1}{4}\frac{0}{7} = 0.21$.
Then, with

$$h_2(\varepsilon) = 0.36$$
$$h_2(\delta) = 0.75$$

the secrecy capacity for a single bit is given by $C_s = 0.4$, while for the whole message it is $C_s = 2.8 \approx 3$.

Finally, the mutual information was evaluated with `task4_mutualinfo.m`, which implements the formula outlined in slide 15; the resulting value is

$$I(u; z) = 0.1048$$

1. How many secret message bits per channel use ("transmitted word") have you obtained with your scheme?How many secret bits per binary digit ("transmitted bit")?

   In order to be able to achieve both perfect reliability and perfect secrecy, the number of bits that can be sent per channel use is equal to $log_2(M)$, where an upper bound on the cardinality of the message space $\mathcal{M}$ is given by:

   $$M = |\mathcal{M}| \leqslant \frac{|\mathcal{Y}|}{N_{y|x}} \frac{N_{z|x}}{|\mathcal{Z}|}$$

   In the scheme just implemented, in which $|\mathcal{M}| = 8$, the upper bound for the number of secret bits obtained per secret word is therefore given by $log_2(8) = 3$. The result coincides with that obtained through the use of the secrecy capacity formula, in which $C_s = 2.8 \approx 3$. The number of secret bits achieved instead per transmitted bit is then 0.4.

2. Is it possible to obtain 4 secret bits per channel use? If so, how should you change your encoder/decoder? If not, why?

   Under the considered scheme's specifications, it is not possible to obtain 4 secret bits per channel use. In order to be able to achieve this, it would be necessary to increase the cardinality of the message space $\mathcal{M}$ since the number of secret bits that can be transmitted depends directly on said parameter.

3. Is it possible to obtain 2 secret bits per channel use? If so, how should you change your encoder/decoder? If not, why?

   It is indeed possible, though not optimal, to obtain 2 secret bits per channel use. Although it is not the maximum secrecy rate, it is an achievable one and can therefore be used. An achievable secrecy rate is such that $|\mathcal{M}_n| \geqslant 2^{nR_s}$. By increasing n, the error probability and mutual information can be steered to 0, at least from some n_0 on.

4. One could consider evaluating the secrecy of this mechanism by cascading the eavesdropper channel with a decoder and measuring the resulting error rates. What do you expect Eve's error would be? Why resort to (more complicated) evaluating the mutual information?

   The random binning decoder implemented in task 3, placed at the output of the legitimate channel, makes use of the hamming distance criterion and is able to detect at most two-bit errors in a single codeword. In the legitimate case, the mechanism works because of the fact that the channel introduces a single-bit error. On the other hand, the eavesdropper channel randomly introduces up to three errors per codeword, in which case the hamming distance decoder will not be able to detect all of them, producing an unreliable measure as a result.
   The statistical distributions for both the legitimate and eavesdropper channels are assumed to be known, and it is specifically true in the case under consideration. From these, the mutual information between the secret message x and that observed by the legitimate receiver y, and the mutual information between x and the message observed by the eavesdropper z, can be easily derived.

Task 5: Simulate transmission over a binary symmetric channel

Task 5 called for the implementation of a wiretap BSC, with parameterised error rates ε, δ over the two channels. This was carried out in `wiretap_bsc.m`: the code iterates over all output bits and flips the current bit depending on a random outcome. Random number generation is done independently for the two channels, in compliance of task specifics.

The final encoder, wiretap channel, decoder chain was implemented in `task5_connection.m`, for the purpose of testing the implementation of the whole system and evaluating performance metrics.

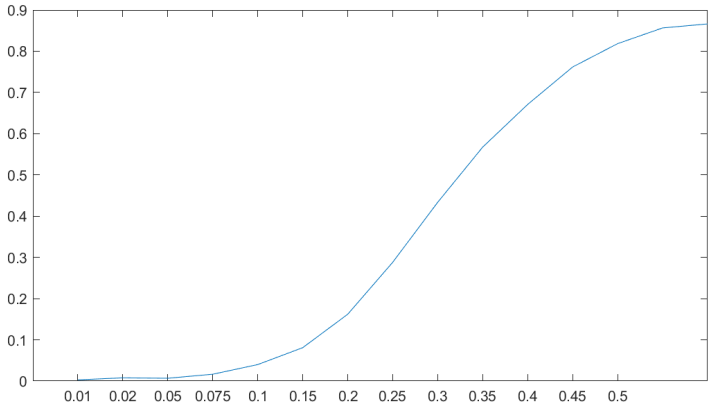

Figure 10: Bob's error rate as a result of channel non-ideality for varying values of ϵ.

Task 6: Evaluate system security over the wiretap BSC

Depending on the values of ε and δ, the legitimate channel will be better or worse than the eavesdropper's channel. It can be stated that the legitimate channel is better than the eavesdropper's ($\varepsilon < \delta$) only if both error probabilities are $< 1/2$. A channel that either always flips the bits or never does so can be considered to be a perfect channel, while one that does so with probability $1/2$ is useless for it is completely unpredictable.
In terms of leaked information, secrecy capacity can be achieved if the channel between the legitimate transmitter and the legitimate receiver is not "more noisy" than the channel between transmitter and eavesdropper. In other words, the mutual information between the message u and the legitimate receiver's observation y must be larger than the mutual information between u and the eavesdropper's observation z.
The mechanism's ideal counterpart corresponds to the case in which the legitimate channel behaves as a perfect channel (with $\varepsilon = 1$ or $\varepsilon = 0$) and the eavesdropper's channel acts as a "dumb" one ($\delta = 0.5$). Graphically, the upper bound can be observed as a point in the upper-left corner of the contour plot in Figure 12.

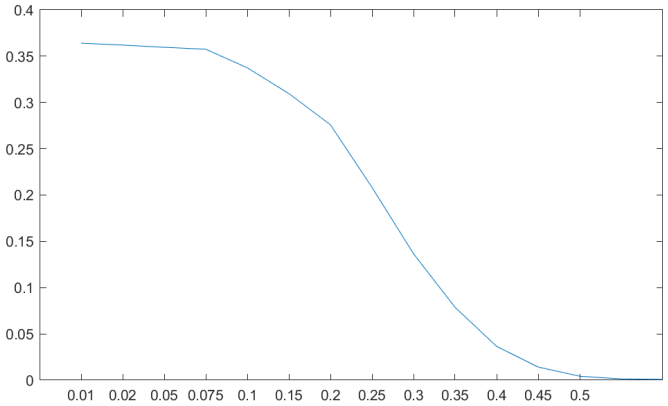

Figure 11: Mutual information $I(u;z)$ as a function of δ.

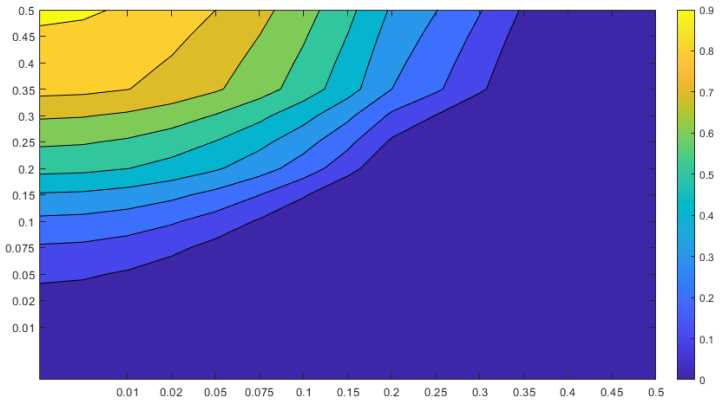

Figure 12: Secrecy capacity as a function of δ and ϵ.