Information Security Lab 2 - Group Flute

Active group members: Simone Dalla Valle, Andrea Manzato, Davide Maria Lazzaro, Filippo Giambartolomei, Nazeer Kawas Mhd

All the tasks have been implemented through the Python programming language. By running the *main.py* it's possible to see the results described in this document. You can run the *main.py* through the command *python3 main.py*. The *numpy* library has to be installed, and a release of **Python** $\geq$ **3.8.0**.

1 Task 1

The first task requires to implement a uniform wiretap channel, that receives as input a 7-bits word and returns the corresponding pair of outputs (y, z). The maximum Hamming distance between the input x and the message collected by the legitimate receiver is 1. For the eavesdropper channel the maximum Hamming distance is 3.

Moreover, y and z are conditionally uniform and independent of each other given x. It means that each configuration has the same probability, regardless of its Hamming distance.

The uniform wiretap channel is implemented by computing XOR operation between the input x and a binary mask. The 1s in the binary mask represent the errors introduced. This mask is calculated as follows: first of all, the number of words with at most k bits equal to 1 is computed as follows:

$$T = \sum_{i=0}^{k} \binom{n}{i} = \binom{n}{0} + ... + \binom{n}{k} \tag{1}$$

Thus, the probality of generating a word with exact i errors is:

$$P[\text{exact } i \text{ errors}] = \frac{\binom{n}{i}}{T} \tag{2}$$

and computational experiments shows that the random generation, that we have implemented, correctly follows this probability distribution (Figure 1 and Figure 2).

Then, a random integer value t in range $[1, T]$ is picked. The generated word will have exactly j errors, where j is the biggest integer such that $\sum_{i=0}^{j} \binom{n}{i} \leq t$. A binary mask with the same lenght of x is computed and it will have j 1s in random distinct positions. The channel output is computed as XOR between x and the mask.

By running 10^4 times the channel realizations and by plotting the frequency of each possible output, you can observe that they follows a uniform distribution. Computing (1) using $k = 1$ and $k = 3$, it turns that, given a fixed x, there are 8 possible outputs for the legitimate channel and 64 for the eavesdropper one. In the figures below a bin reprensents the frequency of a possible channel output. In particular, $P_{z|x}(\cdot|1001000)$ is plotted in Figure 2.

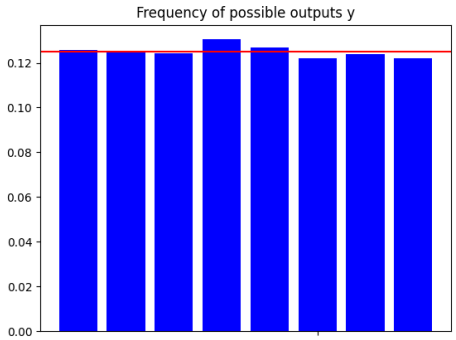

Figure 1: Frequencies of legitimate channel outputs

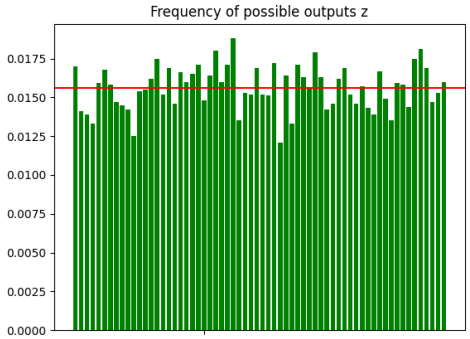

Figure 2: Frequencies of eavesdropper channel outputs

Finally, in order to check conditional independence we have simply checked that:

$$P[y, z|x] = P[y|x] \cdot P[z|x] \tag{3}$$

where $P[y, z|x]$ is the conditional joint probability and $P[y|x]$ and $P[z|x]$ are the probability computed previously.

2 Task 2

The second task requires to implement the uniform binning encoder with a (7,4) Hamming code.
- The generated codewords are:
For message [000]: [0000000] and [1111111];
For message [001]: [0001111] and [1110000];
For message [010]: [0010011] and [1101100];
For message [011]: [0011100] and [1100011];
For message [100]: [0100101] and [1011010];
For message [101]: [0101010] and [1010101];
For message [110]: [0110110] and [1001001];
For message [111]: [0111001] and [1000110].

We can obtain one of the two possible codewords using the function randomBinningEncoder() which randomly returns one of the two.

3 Task 3

The third task requires to implement a decoder for the random binning encoder implemented in task 2. The implemented decoder identifies the most probable sent codeword among all the possible codewords by selecting the one with the minimun Hammming distance with respect to the received codeword.

To verify the correctness of the implementation we checked for that the received message is the correct one without a channel and with a channel introducing at most 1 error per codeword.

- Without a channel the results for 3 messages are:
For the message [011] the decodified message is [011], with [0] corrected errors;
For the message [101] the decodified message is [101], with [0] corrected errors;
For the message [111] the decodified message is [111], with [0] corrected errors.

- With a channel the results are:
For the message [011] the decodified message is [011], with [1] corrected errors;
For the message [101] the decodified message is [101], with [1] corrected errors;
For the message [111] the decodified message is [111], with [1] corrected errors.

4 Task 4

Task 4 requires to implement a chain of encoder+eavesdropper channel. We have run $100 \cdot |Z|$ simulations for every possible value of u. The results are described in Fig. 3. For every message u the codewords are always almost uniformly distributed.

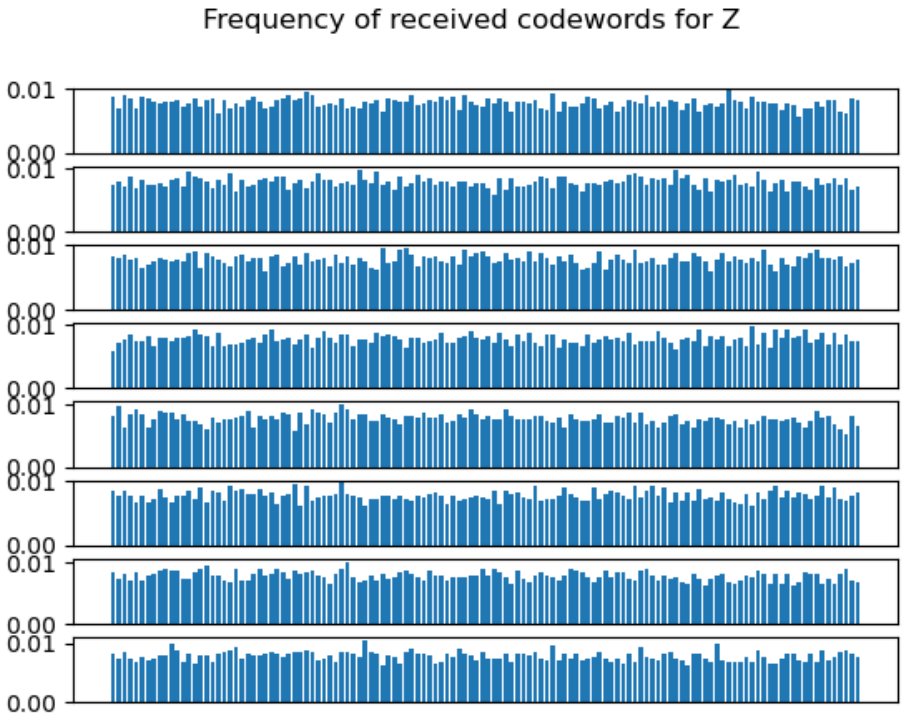

Figure 3: Frequencies of codewords for all the possible messages

We tabulated the joint distribution $p_u, \hat{z(d,c)}$ and computed the marginal distribution, shown in Fig. 4 and computed the mutual information

$$I(u,z) = 0.006609$$

The results suggest that u and z are empirically independent since every codeword is almost equally probable and $I(u,z) \approx 0$.

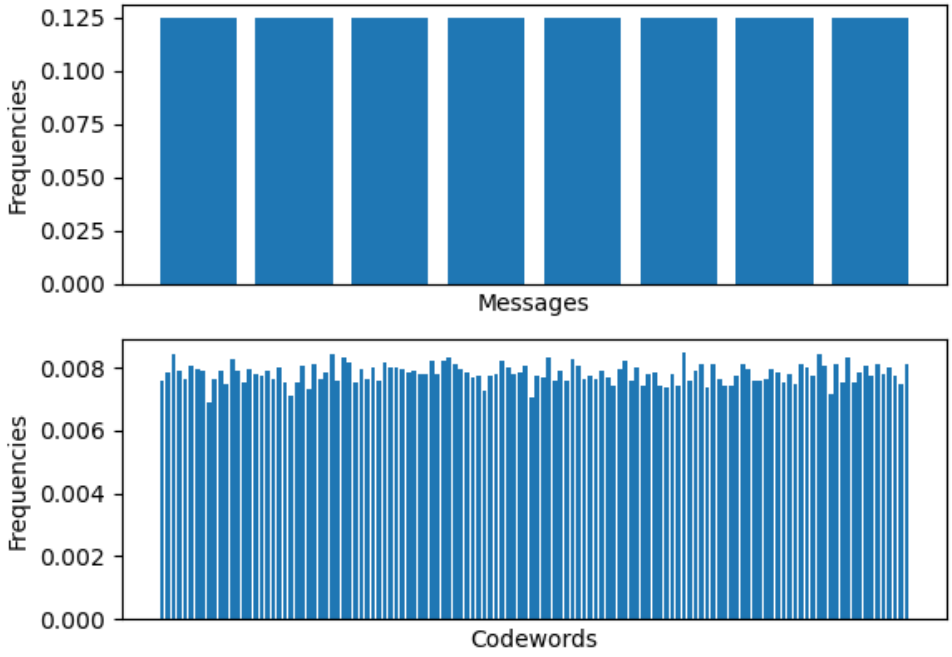

Figure 4: Marginal distribution of u and z

5 Considerations and remarks

1. With the implemented scheme we achieved 8 secret message bits per channel use and 3 secret bits per binary digit.

2. The maximum number of bits that can be trasmitted is described by the following equation:

$$N_{bits} = \log_2 \frac{|Y| N_{z,x}}{N_{y,x} |Z|} \tag{4}$$

   To increase N_{bits} we need to increase $\frac{|Y|}{N_{y|x}}$ or decrease $\frac{|Z|}{N_{z|x}}$.
   To increase $\frac{|Y|}{N_{y|x}}$ we need to modify the encoder to create more codeword for each message. To decrease $\frac{|Z|}{N_{z|x}}$ we need to create increase the size of generated codewords in order to increase $N_{z|x}$. To do so we can use codewords of size ≥ 10 . That, for size $= 10$, would make $N_{z|x} = 176$ and $N_{y|x} = 11$ and if $|Z| = |Y|$ then $N_{bits} = 4$. To avoid changing the number of possible corrected bits by the decoder we can use a (15,11)-Hamming code.

3. To have 2 secret bits we can still use a (7,4)-Hamming code and, at the encoder, generate 4 possible codewords for every message and use only the last 2 bits as input messages. Two of the 4 bits are used in the encoding process and the remaining 2 are the secret bits.

4. In our implementation the probability for the eavesdropper to retrieve the correct word is:

$$P[\hat{u} = u] = \frac{\binom{7}{0} + \binom{7}{1}}{\binom{7}{0} + ... + \binom{7}{3}} = \frac{1}{8}$$

This probability of success is high, however, for this implementation of the system, this is the best performance and it can be shown through the mutual information, that, when it's close to 0 indicates that there is no correlation between the input message and the received codeword at the eavesdropper.

6 Task 5

The task 5 requires to implement a wiretap binary simmetric channel that could be simulated with arbitrary values ε and δ and connected between the random binning encoder and decode. To achieve the randomness of the BSC I considered it as a binomial distribution that describe the number of flipped bit in the message. So I computed the number of error using the function *numpy.random.binomial()* and choose randomly the position of the bit to flip in the message with the *numpy.random.choice()* function.
This random number rapresent the number of error that I will introduce in the message, their position must be randomly picked too.
The function returns an object Results with the message and the number of error that were introduced.
To test the randomness of the introduction of error i tested the function with a message of 5000 bits: **With probabilities $\varepsilon = 0.1$ and $\delta = 0.3$**

For legitimate channel y: Expected errors =[500], Generated errors =[500]
For eavesdropper channel z: Expected errors =[1500], Generated errors =[1501]
For legitimate channel y: Expected errors =[500], Generated errors =[493]
For eavesdropper channel z: Expected errors =[1500], Generated errors =[1504]
For legitimate channel y: Expected errors =[500], Generated errors =[466]
For eavesdropper channel z: Expected errors =[1500], Generated errors =[1462]
For legitimate channel y: Expected errors =[500], Generated errors =[487]
For eavesdropper channel z: Expected errors =[1500], Generated errors =[1526]

As I expected the number of errors are close to the mean of the normal distribution but not always the same

For the concatenation with the random binning encoder and decoder I concatenated all the outputs of the encoder to create the message to fed into the function BSC. Than I splitted the message with errors in 7-bits long messages and fed all to the decoder to obtain a 3-bits word.
I compared the 3-bits strings I generated and fed into the encoder and the decoded ones.
Here are the results for 30 messages:

Total length of the message in bits =[210], Number of errors generated by the channel =[20], Expected number of errors =[21.0]
ERROR codeword =[[1 0 0]], decodedword =[[1 1 0]]
ERROR codeword =[[1 1 1]], decodedword =[[1 0 1]]
ERROR codeword =[[0 0 0]], decodedword =[[0 0 1]]
ERROR codeword =[[0 0 0]], decodedword =[[0 0 1]]
ERROR codeword =[[0 0 0]], decodedword =[[0 0 1]]
ERROR codeword =[[0 0 1]], decodedword =[[0 1 0]]
ERROR codeword =[[0 0 0]], decodedword =[[0 0 1]]

7 Task 6

To evaluate the system security over the wiretap BSC, simulations usinf different values of δ and ϵ are run. We have limited the study of the performances to ranges $\epsilon \in [0, \frac{1}{2}]$ and $\delta \in [0, \frac{1}{2}]$ because dual result will have been obtained if we would have considered the all interval $[0, 1]$.

First, using 200 different values of ϵ in the range $[0, \frac{1}{2}]$, the error decoding probability is estimated repeating the trasmission over the BSC several times. The plot below shows what intuition conjectures. The highest the ϵ value is, the highest the probability will be. $P[\hat{u} \neq u]$ is plotted in the Fig. 5.

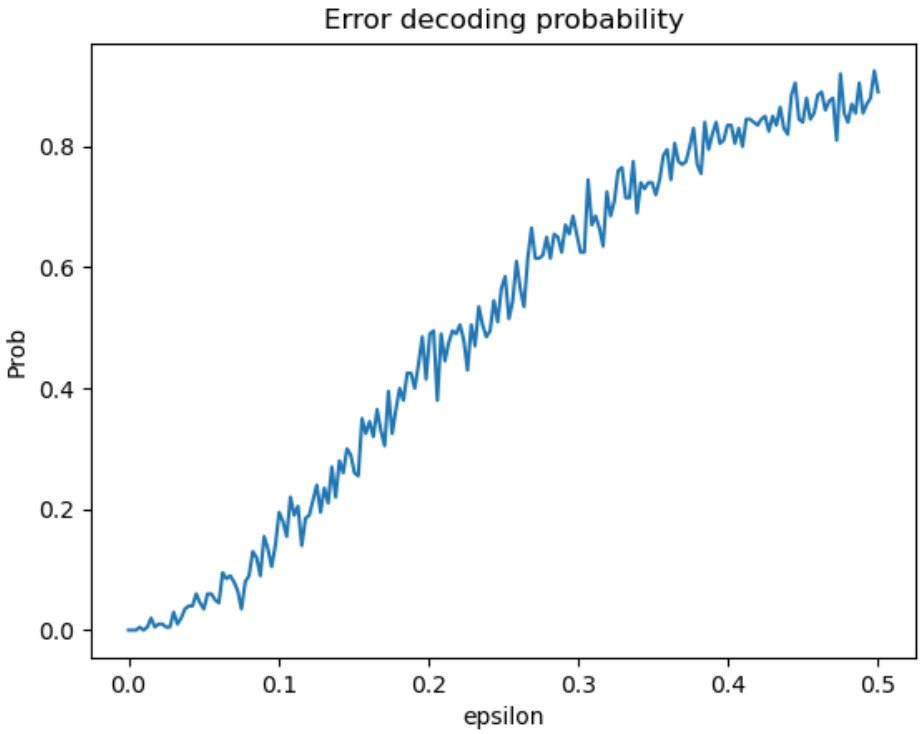

Figure 5: $P[\hat{u} \neq u]$

In order to evaluate the resulting secrecy, we have followed the same steps already done in Task 6. We have computed the empirical joint probability and the marginal distributions w.r.t. y and z for different values of δ and then the mutual information $I(u;z)$ is estimated using these intermediate results. $I(u;z)$ is plotted in Fig. 6 below by varying the value of δ.

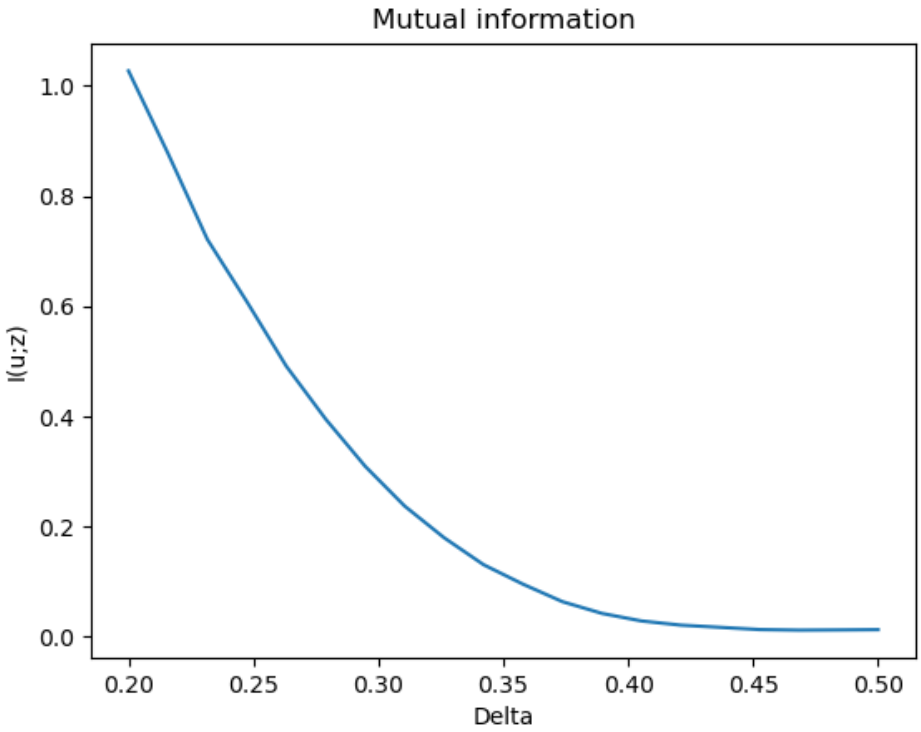
Mutual information
I(u;z)
1.0
0.8
0.6
0.4
0.2
0.0
0.20
0.25
0.30
0.35
0.40
0.45
0.50
Delta

Figure 6: $I(u; z)$