# Cryptography – Project 2

*Code source available on* [*my GitHub*](https://github.com/pival13/ITUProject/blob/master/BLG520_Cryptography/HW2/diffAnalysis.py)*.*

For the second project, we had to build 3 different Substitution-Permutation Network (SPN), with the following substitution tables (4-bits) and permutation tables (16-bits):

The first SPN use as a substation table , and the below permutation table.

The second SPN use the below substitution and permutation tables as in.

The third SPN use only the substation table, and no permutation.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| x | 0000 | 0001 | 0010 | 0011 | 0100 | 0101 | 0110 | 0111 | 1000 | 1001 | 1010 | 1011 | 1100 | 1101 | 1110 | 1111 |
| S(x) | 0000 | 0001 | 1001 | 0110 | 1101 | 0111 | 0011 | 0101 | 1111 | 0010 | 1100 | 1110 | 1010 | 0100 | 1011 | 1000 |

The differential tables for the two substitution tables used can be found at the end of the document. From them, we can see that the initial substation table is quite balanced, with all inputs having from 6 to 8 possible outputs (except 0), while the altered version () is more biased, with a few cases having only 4 different outputs, and some of them appearing half of the time.

Finding the key require several iterations. During each iteration, we try to find some bits of it. Considering two inputs and , differing by , we can find the probability of this . By exemple, on the first SPN, with , we can have with a probability of 3/128. This is achieved by several steps of substitution and permutation:

Then using a thousand different pairs of outputs, whose inputs differ by , we can xor their 4th to 11th bits with every possible subkey, and find the inverse of the substitution. For every subkey giving , we increment a counter. Eventually, the subkey appearing 3 times over 128 will be part of the key. Repeating this operation gives us the while key. Then reverting the key scheduler gives us the master, or initial, key.

I made some tests using the secret key .

On the first SPN, the following have been used:

with a probability of 3/128, giving the subkey

with a probability of 3/128, giving the subkey

with a probability of 3/256, giving the subkey

with a probability of 3/512, giving the subkey 4 and 9.

Obviously, the second and fourth operation have only been performed to check whether the results are coherent. Concatenating the subkeys give us the key . By reverting the key scheduler, we can successfully retrieve our original key .

On the second SPN, the following deltas have been used:

with a probability of 2/128, giving the subkey

with a probability of 9/256, giving the subkey

with a probability of 2/128, giving the subkey

with a probability of 1/128, giving the subkey 4 and 9.

Once again, the second and fourth operations’ goals are to double-check the results. As we can see, the probabilities used are quite similar to the previous case, despite the differences on their differential tables. That aside, the subkeys found are the same.

For the third SPN, the following deltas have been observed:

with a probability of 3/86, giving 12 different possible keys

with a probability of 3/86, giving 12 different possible keys.

with a probability of 3/86, giving 8 different possible keys.

with a probability of 3/86, giving 8 different possible keys.

with a probability of 27/512, giving 2 different possible keys.

with a probability of 27/512, giving 6 different possible keys.

with a probability of 27/512, giving 2 different possible keys.

with a probability of 27/512, giving 4 different possible keys.

This time, unlike previously, there is no clear subkey appearing, even when trying several different cases. Of course, considering there is *only* 96 possible combinations using the subkeys, it may be possible to try each of them until the correct one is found. However this is not, in my opinion, a viable solution.

Of course, the highest probabilities have been used in every of the previous cases. So, the only conclusion possible here is that the differential analysis is not adequate to analyze this third SPN.

Differential table of SPN 2 and 3

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0000 | 0001 | 0010 | 0011 | 0100 | 0101 | 0110 | 0111 | 1000 | 1001 | 1010 | 1011 | 1100 | 1101 | 1110 | 1111 |
| 0000 | 16 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 0001 |  | 2 | 2 | 2 |  |  | 2 |  |  |  | 2 |  |  | 2 | 2 | 2 |
| 0010 |  | 2 | 2 | 2 |  |  |  | 2 |  | 2 |  |  | 4 |  | 2 |  |
| 0011 |  | 2 | 2 |  | 2 |  | 2 |  | 4 |  |  |  |  |  | 2 | 2 |
| 0100 |  |  |  | 2 |  | 2 | 6 | 2 |  |  | 2 |  |  | 2 |  |  |
| 0101 |  |  |  |  | 2 | 4 |  | 2 | 2 |  |  | 2 | 4 |  |  |  |
| 0110 |  | 2 |  | 2 | 6 |  | 2 |  |  |  | 4 |  |  |  |  |  |
| 0111 |  |  | 2 |  | 2 | 2 |  | 2 | 2 | 2 |  | 2 |  |  | 2 |  |
| 1000 |  |  |  | 3 |  | 2 |  | 2 | 2 |  |  |  |  | 2 |  | 2 |
| 1001 |  |  | 2 |  |  |  |  | 2 |  | 2 | 2 | 2 |  | 2 | 4 |  |
| 1010 |  | 2 |  |  | 2 |  | 4 |  |  | 2 |  |  | 2 |  |  | 4 |
| 1011 |  |  |  |  |  | 2 |  | 2 |  | 2 |  | 2 | 2 | 2 | 2 | 2 |
| 1100 |  |  | 4 |  |  | 4 |  |  |  |  | 2 | 2 |  |  | 2 | 2 |
| 1101 |  | 2 |  |  | 2 |  |  |  | 2 | 2 |  | 2 |  | 4 |  | 2 |
| 1110 |  | 2 | 2 | 2 |  |  |  | 2 |  | 4 |  | 2 | 2 |  |  |  |
| 1111 |  | 2 |  | 2 |  |  |  |  | 2 |  | 4 | 2 | 2 | 2 |  |  |

Differential table of SPN 1

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0000 | 0001 | 0010 | 0011 | 0100 | 0101 | 0110 | 0111 | 1000 | 1001 | 1010 | 1011 | 1100 | 1101 | 1110 | 1111 |
| 0000 | 16 |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| 0001 |  |  |  |  | 4 |  |  |  | 4 |  | 2 | 2 |  |  | 2 | 2 |
| 0010 |  |  | 2 |  | 2 |  | 2 | 2 |  |  |  | 2 | 6 |  |  |  |
| 0011 |  |  | 2 | 2 | 2 |  | 2 |  | 4 |  |  |  | 2 |  |  | 2 |
| 0100 |  |  | 2 | 2 |  |  | 6 | 2 |  |  | 4 |  |  |  |  |  |
| 0101 |  |  | 4 |  |  |  |  |  | 4 | 4 |  |  |  |  | 4 |  |
| 0110 |  | 4 |  |  | 2 |  | 2 |  |  |  | 4 |  | 2 |  | 2 |  |
| 0111 |  |  | 2 |  | 2 | 4 |  |  |  |  | 2 |  | 2 |  | 4 |  |
| 1000 |  |  |  | 4 | 2 |  | 2 | 4 |  |  |  |  |  | 2 |  | 2 |
| 1001 |  |  |  | 2 |  | 2 |  | 4 |  |  |  | 2 | 4 | 2 |  |  |
| 1010 |  | 4 |  | 4 | 4 |  |  |  |  |  |  |  |  |  |  | 4 |
| 1011 |  |  |  |  |  |  |  | 4 |  |  | 2 | 6 |  |  | 2 | 2 |
| 1100 |  |  | 2 |  |  | 6 |  |  |  |  |  | 2 |  | 2 | 2 | 2 |
| 1101 |  | 4 |  |  |  | 2 | 2 |  |  |  | 2 | 2 |  | 2 |  | 2 |
| 1110 |  |  | 2 | 2 |  |  | 2 | 2 |  | 8 |  |  |  |  |  |  |
| 1111 |  | 4 |  |  |  |  |  |  |  | 4 |  |  |  | 8 |  |  |