**Homework sheet 1, due 7 December 2017 at 13:45**

Some of these exercises require more knowledge than you have when they come out. We will cover the missing parts next week but you can already start on the first part.
If you use sage for your computations note that you can compute modulo a polynomial using %. In Pari the function `Mod` also works for polynomials.

Submit your homework by encrypted and signed email to Gustavo and Tanja. Do not forget to attach your public key and the public key of anybody you put in cc.

1. For both of the following sequences

$$s_{k+8} = s_{k+5} + s_{k+2} + s_{k+1} + s_k \qquad s_{k+6} = s_{k+3} + s_k$$

   do the following subexercises:

   (a) Draw the LFSR corresponding this sequence.

   (b) State the associated matrix corresponding to the LFSR state update and compute its order.

   (c) State the characteristic polynomial $f$ and compute its factorization.

   (d) For each of the factors of $f$ compute the order.

   (e) What is the longest period generated by this LFSR? Make sure to justify your answer.

   (f) State the lengths of all subsequences so that each state of $n$ bits appears exactly once.

2. In RC4 we need to swap two states. This is easiest to do using an extra variable, i.e. we copy $S[i]$ to `dummy`, copy $S[j]$ to $S[i]$ and finally copy `dummy` to $S[j]$. To save on storage space one might have the idea to implement the swap in the following three steps:

   (a) $S[i] \leftarrow S[i]$ xor $S[j]$

   (b) $S[j] \leftarrow S[i]$ xor $S[j]$

   (c) $S[i] \leftarrow S[i]$ xor $S[j]$

   Explain first why this usually computes the correct $S[i]$ and $S[j]$. Now assume that this piece of code does the swap in the second part of the code (after the key setup). Explain why this can go wrong and state (with explanation) the expected number of steps until this goes wrong for the first time. Explain what happens long term with this implementation. Note that there are multiple possibilities of what happens. I don't expect a full analysis.

3. This exercise expects you to brute force RC4 at "export-cipher" strength (40 bit keys). Through some side-channel information you learn that this key was set up for 2WF80 and that the first byte `key`$[0] = 80$. Find a key that could have produced the following output sequence:

   130, 189, 254, 192, 238, 132, 216, 132, 82, 173.