Homework sheet 2, due 05 December 2019 at 13:30

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 all three TAs. Make sure to test early whether your encryption program can handle Jonathan's key and if not, contact him for an alternative. Do not forget to attach your public key and the public key of anybody you put in cc.

1. This exercise is about LFSRs. You know that A and B use an LFSR with state size 6. You observe ciphertext 10100100100010111010 and know that start of the message was tue and that the following encoding was used:

```
a -> 00000
b -> 00001
c -> 00010
...
z -> 11001
0 -> 11010
...
5 -> 11111
```

   The ciphertext is the xor of the message with the output stream of the LFSR.

   (a) Compute the first 15 bits of the LFSR output — 2 point

   (b) Compute the initialization vector and the feedback coefficients of the LFSR. — 6 point

   (c) Compute the next character after tue. — 2 point

   Note that the encoding used above is the binary representation of the integer associated to the letter

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Q	R	S	T	U	V	W	X	Y	Z	0	1	2	3	4	5
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

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. 5 points

3. This exercise expects you to brute force RC4 at "export-cipher" strength (40 bit = 5 byte 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.

   Note: this should be a feasible computation on a notebook. If it feels like more computation chances are that something is wrong in your approach. 5 points