**Exercise sheet 3, 30 November 2023**

This exercise sheet takes you on a trip to investigate a stream cipher. You will explore statistical properties of the cipher and also find some explanations for what you see.
The cipher uses a 256 byte state vector $S$ (array of 256 bytes) which at every moment contains a permutation of the integers 0,1, ...,255. The key $k$ consists of $\ell$ bytes, where $\ell$ is at least 5 and at most 256.
Note: 1 byte = 8 bits, so the numbers $0, 1, \ldots, 255$ are exactly those that can be represented in one byte. All computations are on indices; the swaps cause permutations of $S$. The key positions are taken modulo the key length, so typically each key byte gets used multiple times in the initialization.

### Cipher setup

```
# initialize
for i = 0 to 255:
    S[i] = i # this puts i into S[i]

# feed in the key, key k has length l
j = 0
for i = 0 to 255:
   j = (j + S[i] + k[i mod l]) mod 256   # j is a number in [0,255]
   # swap S[i] and S[j]
   dummy = S[i]
   S[i] = S[j]
   S[j] = dummy

# generate n bytes of output stream
i = 0
j = 0
for k = 0 to n-1:
   i = (i + 1) mod 256   # i is a number in [0,255]
   j = (j + S[i]) mod 256  # j is a number in [0,255]
   # swap S[i] and S[j]
   dummy = S[i]
   S[i] = S[j]
   S[j] = dummy
   output S[(S[i] + S[j]) mod 256]   # this grabs the contents of S
                                     # at index (S[i] + S[j]) mod 256
```

You can find a Python implementation here and a Sage implementation here.
To use them in the experiments you need an outer loop to pick the keys, set the output lengths, and collect the outputs, but this should get you started.
Stan Korzilius, was so nice to contribute an implementation in Java for you to use.

1. Take 16 bytes as keylength. Generate outputs of 2 bytes for many randomly chosen keys,, i.e., each time randomly pick 16 integers in $[0, 255]$ as key and output two values. (In the above variables, that means `l = 16, n = 2`.)

   Plot the distribution of the second output byte.

2. What happens to the output if $S[2] = 0$ at the end of the key-setup stage? Explain your observation in the previous part.

3. Take 16 bytes as keylength; vary the key but keep the first byte of it fixed and plot the first output byte.

4. Take 16 bytes as keylength; vary the first three key bytes and keep the remaining ones constant. Plot the distribution of the third output byte + key[0] + key[1] + key[2] + key[3].

5. Read the specification of WEP (the older protocol to connect to routers). How can you use the knowledge from the first three parts to likely break it?

6. Check out the documentation and explanation of Aircrack-ng.