Assignment # 2

Homework

Homework problems are a preparation for the quizzes. They are *not* graded. Please use the `mywpi` forum to post questions you have on these problems.

- 2.2, 2.4, 2.5 ($C(x) = 1 + x + x^3$), 2.6, 2.8, 2.10, 3.3, 3.5, 3.7, 3.12

Project

1. Problem 2.11 from the book.
2. Problem 3.6 from the book.
3. One important property that makes DES secure is the non-linearity of the S-boxes. In particular, a single change in the input should affect at least half the bits of the output. To verify this, we will check the *strict avalanche criterion* (SAC). It is computed as:

$$\sum_{x \in \mathbb{Z}_{2^n}} S(x) \oplus S(x \oplus 2^i) \geq (2^{(n-1)}, 2^{(n-1)}, \ldots, 2^{(n-1)})$$

   The S-box is a function that maps a $n-$bit input to an $m-$bit output $S : \mathbb{Z}_{2^n} \rightarrow \mathbb{Z}_{2^m}$, i.e. for DES $n = 6, m = 4$. Furthermore, $(2^{(n-1)}, 2^{(n-1)}, 2^{(n-1)}, 2^{(n-1)})$ is the individual sum over each output bit position. In order for the SAC to be fulfilled, the value of each individual sum must be greater or equal to $2^{(n-1)}$.

   Hint: The SAC is computed for each output bit separately, i.e. you should get 6x4 SAC coefficients: one per s-box output bit is $m = 4$ per row, one row for each input bit position, giving $n = 6$ rows, i.e. you are checking 6x4 values.

   (a) Implement a function that on the input $x \in \mathbb{Z}_{2^6}$ produces an output $y \in \mathbb{Z}_{2^4}$ according to the DES S-box S_4 (i.e. implement the DES S-box S_4).

   (b) Write a program that, based on the above implementation of S_4 computes the SAC for $i = 0 \ldots 5$. Is the SAC fulfilled by S_4? Please turn in both the code and the output.

4. Implement an exhaustive key search for DES. Recover the key for the following plaintext-ciphertext pair (both given in hex notation):

   ```
   pt = 48656c6c6f212121
   ct = 1f6339383e8da6c4
   ```

   Please turn in your working code together with the correct 64-bit representation of the key.

   Note: while the key space of DES is way too small by now, it is still too large to be searched in reasonable time on a simple desktop PC. In order to facilitate the search, the first four bytes of the 64-bit key have been fixed to 0. i.e. the 64-bit key looks like this (in hex representation): `00000000XXXXXXXX`.

   Hint: Do not try to implement DES yourself. Use an existing implementation of DES. Possible choices are:

   - In `sage`, a DES engine is available in `Crypto.Cipher`

     ```
     from Crypto.Cipher import DES
     ...
     ```

     (Note that the server might quickly be overloaded if many people try to run an exhaustive search in parallel).
   - For `Python` there is a versatile crypto library available at: `https://www.dlitz.net/software/pycrypto/`. It needs an additional install, but provides the same functionality
   - Any `openSSL` implementation will do. The mode to be selected should be `des_ede3_ecb`. The key needs to be replicated three times.

Bonus Problem

Recover the DES key for the following plaintext-ciphertext pair (both given in hex notation):

```
pt = 48656c6c6f212121
ct = ddb92846141922b8
```

The key starts with zeros, just as the one before (but less). Please turn in your working code together with the correct 64-bit representation of the key and the runtime needed.

Good Luck and Have Fun!