Exercise sheet 4, 03 December 2020

This exercise sheet is mostly about the block cipher DES. A block cipher takes a block of b bits of data and encrypts it to b bits of data in an invertible manner, using a key k of ℓ bits:

$$\mathrm{Enc} : \{0,1\}^b \times \{0,1\}^\ell \to \{0,1\}^b; \quad (M,k) \mapsto \mathrm{Enc}_k(M) = C.$$

You can find the authoritative description of DES at http://csrc.nist.gov/publications/fips/fips46-3/fips46-3.pdf Part of the learning target for these exercises is that you learn how to read a crypto standard, so you should use this document for the following exercises. For context I give some explanation.

DES is a Feistel cipher, that means that the message block of 64 bits is split into a left half of 32 bits and a right half of 32 bits. In every round the right half is used to encrypt the left half and the sides flip; more formally to compute (L_i, R_i) from (L_{i-1}, R_{i-1}) put $L_i = R_{i-1}$ and $R_i = L_{i-1} \oplus f_i(R_{i-1})$. The f_is are functions that depend on the round key k_i.

At the beginning and end of DES the input bits are permuted using permutation IP and its inverse $FP = IP^{-1}$.

The key schedule takes the 64 bit key and expands it into 16 subkeys $k_1, k_2, \ldots, k_{16}$ of 48 bits each, one per round of DES.

The function f_i takes the 32-bit input, expands it into 48 bits, XORs those with the round key k_i and splits the resulting 48 bits into 8 blocks of 6 bits. Each of those blocks then gets fed through some substitution box (S-box) $S_1, S_2, \ldots, S_8$ which maps 6 bits to 4 bits. Finally those 32 bits get combined and permuted and then XORed into L_i.

After the last round the bits ae permuted using FP, and output.

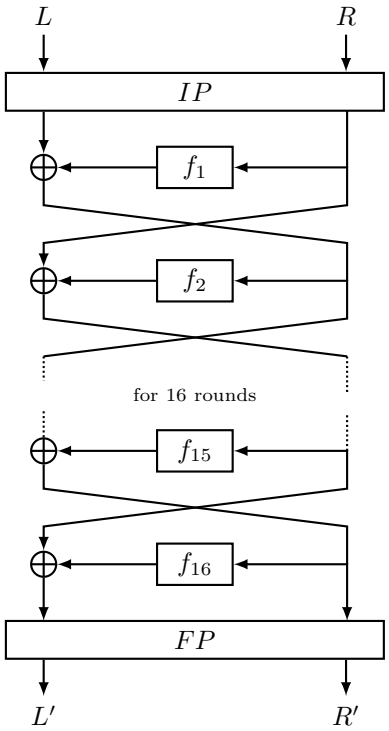


Image: Jérémy Jean at https://www.iacr.org/authors/tikz/

1. Explain how decryption works (note that the f_i are not invertible). State how to recover (L_{i-1}, R_{i-1}) from (L_i, R_i).

2. Take S_5 and compute $S_5(x_1) \oplus S_5(x_2)$ and compare the result with $S_5(x_1 \oplus x_2)$ for the following values:

   (a) $x_1 = (000000), x_2 = (100000)$

   (b) $x_1 = (111111), x_2 = (000001)$

   (c) $x_1 = (000000), x_2 = (101010)$

3. Compute the first subkey if the 56-bit key consists of 56 zeros.

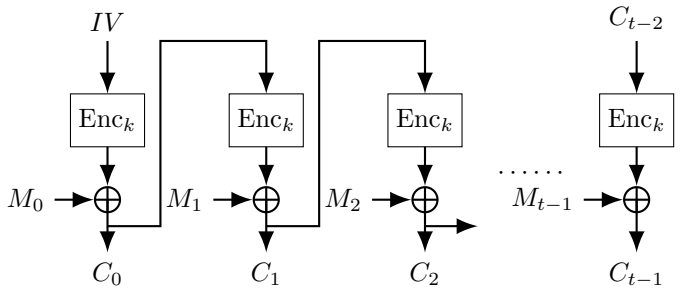
4. Compute the output of the first round (i.e. include the initial permutation, the split into left and right, the function f, the xor and the swap) when the input is the all-zero string and the key is 56 zeros.

5. Compute the output of the first round (i.e. include the initial permutation, the split into left and right, the function f, the xor and the swap) when the input is the all-zero string with the rightmost bit replaced by 1 and the key is 56 zeros. Do not forget the initial permutation.

6. To encrypt a long message m of more than b bits, split it into blocks of length b and append some padding (specified by the system) to reach a multiple of the block length, turning m into $M_0\, M_1\, M_2 \ldots M_{t-1}$.

   The Electronic-Code-Book (ECB) mode encrypts each block separately: $C_i = \mathrm{Enc}_k(M_i)$. This is easy to understand and easy to decrypt as $\mathrm{Dec}_k(C_i) = M_i$ but has the same problems as any substitution cipher, namely that it breaks under frequency analysis. Look at the famous ECB penguin to see the problem.

   The Cipher Feedback Mode (CFB) avoids this issue by chaining the encryptions, starting with an IV, which gets sent as part of the ciphertext.



   Show how to encrypt $(IV, M_0, M_1, M_2, \ldots)$ and to decrypt $(IV, C_0, C_1, C_2, \ldots)$.

7. Check out AngeCryption and a nice example. If you don't want to run the code you can see what it does here. Full version in chapter 10.

8. Read up on the POODLE attack (e.g `https://en.wikipedia.org/wiki/POODLE#POODLE_attack_against_TLS` and references). This uses a weakness is a different mode, namely CBC.