2.3 Exercise 3

Implement, using Magma, the cipher $\texttt{TOY}_{16}$ specified below.

2.3.1 $\texttt{TOY}_{16}$ specifications

The cipher $\texttt{TOY}_{16}$ is a 12-round 16-bit block cipher derived from the framework of Substituition-Permutation Networks, i.e. it is a set of 2^{16} encryption bijections of $(\mathbb{F}_2)^{16}$ where each round function is defined as the composition of a non-linear confusion layer, a linear diffusion layer and a key addition. In order to describe its components, let us fix the notation used throughout the remainder of this document. Let us denote by $V = (\mathbb{F}_2)^{16}$ the message space, i.e. the vector space of all the strings of 16 bits. The set V denotes also the key space of the cipher, i.e. each key of $\texttt{TOY}_{16}$ is a sequence of 16 bits. Each message (or key) $x \in V$ is here represented by its coordinates in the following way: $x = (x_1, x_2, \dots, x_{16})$. The message space V is divided into four *bricks* of size four, i.e. $V = V_1 \oplus V_2 \oplus V_3 \oplus V_4$, where $V_i = (\mathbb{F}_2)^4$ for each $1 \le i \le 4$. For each key $k \in V$ the encryption function E_k is defined as

$$E_k \stackrel{\text{def}}{=} \prod_{i=1}^{12} (\sigma_{k_i} \circ \lambda \circ \gamma) = (\sigma_{k_{12}} \circ \lambda \circ \gamma) \circ \dots \circ (\sigma_{k_1} \circ \lambda \circ \gamma)\,,$$

where $\sigma_{k_i} \circ \lambda \circ \gamma(x) = \lambda(\gamma(x)) + k_i$ and where the confusion layer γ is defined as specified in Sec. 2.3.2, the diffusion layer λ is specified in Sec. 2.3.3 and σ_{k_i} denotes the sum with the round key k_i which is derived from k by the key-scheduling algorithm defined in Sec. 2.3.4. An example of 1-round encryption of an SPN cipher is depicted in Fig. 1.

2.3.2 The confusion layer

The confusion layer γ is a permutation of $(\mathbb{F}_2)^{16}$ which applies on each brick the Sbox $\gamma' : (\mathbb{F}_2)^4 \to (\mathbb{F}_2)^4$ in a parallel way, i.e.

$$\gamma(x) = \gamma(x_1, x_2, \dots, x_{16}) \stackrel{\text{def}}{=} (\gamma'(x_1, x_2, x_3, x_4), \dots, \gamma'(x_{13}, x_{14}, x_{15}, x_{16}))\,.$$

The Sbox γ' is defined from the inversion in the finite field $\mathbb{F}_2[t]/\langle g \rangle$, where g is the polynomial of Sec. 2.1. Given $x \in (\mathbb{F}_2)^4$, compute $\gamma'(x)$ proceeding as follows:

- if $x = 0$, then $\gamma'(x) \stackrel{\text{def}}{=} 0$;

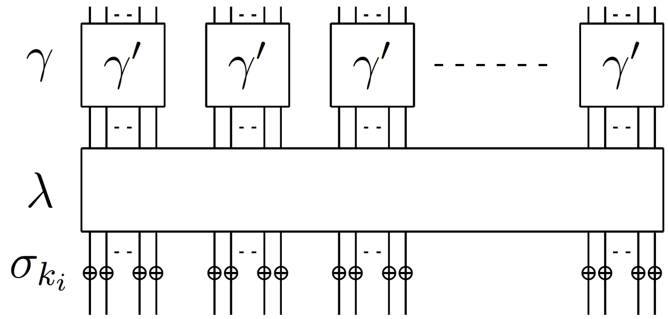

Figure 1: Example of 1-round encryption

- if $x \neq 0$, interpret the 4-bit vector $x = (x_1, x_2, x_3, x_4) \in (\mathbb{F}_2)^4$ as the polynomial $p = x_1 t^3 + x_2 t^2 + x_3 t + x_4 \in \mathbb{F}_2[t]$;
- determine the inverse of p as an element of the field $\mathbb{F}_2[t]/\langle\, g\,\rangle$, i.e. the polynomial $q \in \mathbb{F}_2[t]$ such that $p * q = 1 \mod g$;
- denote the coefficients of q as y_1, y_2, y_3, y_4, i.e. $q = y_1 t^3 + y_2 t^2 + y_3 t + y_4$;
- define $\gamma'(x) \overset{\text{def}}{=} (y_1, y_2, y_3, y_4) \in (\mathbb{F}_2)^4$.

2.3.3 The diffusion layer

The diffusion layer λ is a linear permutation acting on the bits of each message according to the following rule:

$$
\begin{aligned}
\lambda(x) &= \lambda(x_1, x_2, \dots, x_{16}) \\
&\overset{\text{def}}{=} (x_1, x_5, x_9, x_{13}, x_2, x_6, x_{10}, x_{14}, x_3, x_7, x_{11}, x_{15}, x_4, x_8, x_{12}, x_{16}).
\end{aligned}
$$

2.3.4 The key-schedule

The key-schedule of $\texttt{TOY}_{16}$ is an algorithm which takes in input any key $k \in V$ and produces the sequence of the 12 round keys $(k_1, \dots, k_{12}) \in V^{12}$ to be used during the encryption process. The procedure uses a subroutine called Rot which operates on the vector by right shifting its coordinates of $s \overset{\text{def}}{=} 8 - a$ positions, where a is the integer corresponding to the solution of Exercise 2.2. As an example, if $s = 1$, then $\mathrm{Rot}(1, 0, 0, \dots, 0, 1) = (1, 1, 0, 0, \dots, 0)$. The round keys are then derived as follows:

- $k_1 \overset{\text{def}}{=} k$;
- for each $2 \le i \le 12$, if $k_{i-1} = (l_1, \dots, l_{16})$, then

  $$k_i \overset{\text{def}}{=} \text{Rot}\left(\gamma'(l_1, l_2, l_3, l_4), l_5, l_6, l_7, l_8, \gamma'(l_9, l_{10}, l_{11}, l_{12}), l_{13}, l_{14}, l_{15}, l_{16}\right),$$

  i.e. k_i is obtained from k_{i-1} by updating the values of the first and third brick using the Sbox and then applying a rightward rotation of s bits.

2.3.5 Test vectors

The correctness of your implementation of $\texttt{TOY}_{16}$ may be checked using the following test vectors.

x	k	$E_k(x)$
(0000000000000000)	(1111111111111111)	(0000110010110110)
(1111111111111111)	(1111111111111111)	(1001011001000000)
(1111100011100110)	(1100001010101010)	(1001100101101100)

Please provide your Magma code in a separate *txt* file, written in such a way we can copy and paste it into our terminal and check its correctness. It is recommended to implement the encryption process as function which takes in input the message and the chosen key and produces as output the encrypted message, whatever way you decided to represent it.

2.4 Exercise 4 *

1. Let $E_k(x) = (0,1,1,0,0,1,1,0,0,1,0,1,0,0,0,0)$ be the encryption of the message $x \overset{\text{def}}{=} (1,0,1,0,1,1,1,1,0,1,1,0,0,1,0,1)$ with the unknown key k. Find the key.
2. Is $\texttt{TOY}_{16}$ a perfect cipher?

Hints

- Exercise 2: think before using Magma.
- $\gamma'((1,1,0,1)) = (1,1,0,0)$.