

# Tarea 1 (incompleta)

Un elemento del campo  $\mathbb{F}_{256}$  puede representarse con un byte. Esto se obtiene directamente porque los elementos del campo pueden verse como polinomios de grado a lo más 7, con coeficientes en  $\mathbb{F}_2$ . La suma y resta de estos elementos, por ser «bit a bit», también es un polinomio de grado  $< 8$ , pero al multiplicar dos polinomios el resultado puede ser de grado  $\geq 8$ . Para arreglar esto se hace una reducción módulo cierto polinomio  $m(x)$ . En esta tarea, para construir  $\mathbb{F}_{256}$  usaremos el polinomio  $m(x) = x^8 + x^4 + x^3 + x + 1$ .

Por ejemplo, el polinomio  $p(x) = x^4 + x + 1$  puede representarse con el byte `00010011 = 0x13`, mientras que el polinomio  $q(x) = x^7 + x^5 + x$  corresponde al byte `10100010 = 0xA2`. Entonces  $p(x) + q(x) = x^7 + x^5 + x^4 + 1$ , que corresponde al byte `10110001 = 0xB1`; o sea que la suma en  $\mathbb{F}_{256}$  es simplemente el XOR entre bytes. El caso del producto no es tan directo, pues hay que multiplicar los elementos y hacer la reducción módulo  $m(x)$ . Como ejemplo,  $p(x) \cdot q(x) = x^{11} + x^9 + x^8 + x^7 + x^6 + x^2 + x$ , que no es de grado menor a 8, pero al reducirlo módulo  $m(x)$  obtenemos  $x^5 + x^4 + x + 1$ .

## Ejercicios

1. Crea un programa llamado `bytes` que reciba como parámetros dos archivos cualesquiera,  $A$  y  $B$ , y que haga lo siguiente. La ejecución deberá ser algo así:

```
$ bytes nombre_A nombre_B
```

- a) En un archivo llamado `xor.out` se guardará el resultado de  $A \oplus B$ , es decir, el XOR entre los bytes de  $A$  y los bytes

de  $B$ . Si  $|A| \neq |B|$ , el archivo más pequeño hay que rellenarlo con la cadena `beetlejuice` tantas veces como sea necesario para que ambos archivos tengan el mismo tamaño (puede agregarse desde una letra). Por ejemplo, si  $|A| = 20$  y  $|B| = 7$ , entonces al archivo  $B$  hay que agregarle la cadena `beetlejuicebe` antes de hacer el XOR. (2 puntos)

- b) Cada byte de  $A$  será multiplicado por el byte `0xAA`. Esta es la multiplicación de  $\mathbb{F}_{256}$  como se construyó anteriormente, es decir, multiplicando los polinomios correspondientes y reduciendo módulo  $m(x)$ . El resultado será guardado en un archivo llamado `multiplicacion.out`. (4 puntos)
- c) (Extra) Si ahora queremos trabajar con bloques de 4 bytes, estos se pueden representar con polinomios de grado menor a 4 con coeficientes en  $\mathbb{F}_{256}$ . Implementa la aritmética de estos polinomios módulo  $x^4 + 1$ . Cada bloque de 4 bytes de  $A$  será multiplicado por el bloque `0x03010102`, y si  $|A|$  no es múltiplo de 4, se rellena con ceros. El resultado será guardado en un archivo llamado `multiplicacion_poli.out`. (4 puntos)

2. Si  $X$  es un conjunto, la notación  $x \xleftarrow{R} X$  quiere decir que a  $x$  se le asigna un elemento de  $X$  al azar, es decir, se escoge un elemento de  $X$  con la distribución uniforme. Por ejemplo,  $B \xleftarrow{R} \{0, 1\}^8$  dice que se escoge un byte aleatorio y se guarda en  $B$ .

Responde las siguientes preguntas mostrando los cálculos realizados. (2 puntos)

- a) Si  $B_1 = \text{0xAE}$  y  $B_2 \xleftarrow{R} \{0, 1\}^8$ , ¿cuál es la probabilidad de que  $B_1 \oplus B_2 = \text{0x00}$ ?
- b) Si en el inciso anterior  $B_1$  también es escogido al azar, ¿qué probabilidad se obtiene?
- c) En el inciso a), ¿cómo cambia la probabilidad si en vez de `0xAE` y `0x00` usamos otros bytes?

3.